Convergence based Evaluation Strategies for Learning Agent of Hyper-heuristic Framework for Test Case Prioritization

Jinjin Han  
College of Information Science and Technology  
Beijing University of Chemical Technology  
Beijing, P.R.China  
iehanjinjin@163.com

Junxia Guo  
College of Information Science and Technology  
Beijing University of Chemical Technology  
Beijing, P.R.China  
gjxia@mail.buct.edu.cn

Zheng Li*  
College of Information Science and Technology  
Beijing University of Chemical Technology  
Beijing, P.R.China  
lzheng@mail.buct.edu.cn

Ruilian Zhao  
College of Information Science and Technology  
Beijing University of Chemical Technology  
Beijing, P.R.China  
rlzhao@mail.buct.edu.cn

Abstract—Learning agent plays significant role in the hyper-heuristic framework for test case prioritization, where an evaluation strategy is applied to evaluate the execution results produced by the current heuristic algorithm and select the most appropriate heuristic algorithm for the next generation. Hierarchical Distribution (HD) is used as evaluation strategy based on the dominance relationship between the individuals from the present and last generations. In addition to the distribution of the solution set, a good convergence towards the optimal Pareto front is often desired. In this paper, the convergence ability of the individuals is further considered in the design of the evaluation strategy for the learning agent, in which Pareto Dominance and Convergence Information are adopted. Three evaluation strategies are proposed and empirically studied, and the experimental results show that the hyper-heuristic algorithms with the proposed evaluation strategies are more effective and efficient for test case prioritization.

Index Terms—Test Case Prioritization, Hyper-heuristic Algorithm, Learning Agent, Evaluation Strategy

I. INTRODUCTION

Software engineers rerun all test cases once their software is modified, which is referred to as regression testing. Regression testing is frequently executed throughout the entire process of software development and thus requires a large amount of time and effort. In order to reduce the cost of regression testing, researchers have proposed a variety of test suite optimization techniques, such as test suite minimization (TSM), test case selection (TCS) and test case prioritization (TCP). TSM reduces the test cost by removing the obsolete or redundant test cases from the original test suite to obtain a minimal subset which can satisfy the specific test requirements. TCS selects the test cases associated with the changed codes from the original test suite to reduce the cost of regression testing. TCP aims to find the optimal execution sequences of test cases to achieve testing objectives as soon as possible. Unlike TSM and TCS, TCP only executes all test cases in a certain order, thus TCP does not remove test cases. It is impossible to omit important test cases which can detect some hidden faults in the TCP process. As a result, TCP is regarded as the most critical test suite optimization technique. A variety of techniques have been proposed to prioritize test cases, such as many total/additional prioritization techniques based on coverage criterion [1]. In addition, some search based algorithms have been applied to TCP, including Greedy, Additional Greedy, Hill Climbing and Genetic Algorithms [2]. The Epistatic Test Case Segment (ETS) based genetic algorithms are also used to optimize the test case execution order [3]. However, TCP always involves multiple optimization objectives in real-world industrial applications, such as code coverage effectiveness and execution efficiency, so Multi-objective TCP (MoTCP) has attracted more attention recently.

Multi-objective evolutionary algorithms (MOEAs) have shown to be powerful in dealing with multi-objective problems, in which convergence and diversity are two main goals. Consequently, in the design of MOEAs, two ultimate goals should be considered: (1) minimizing the gap between candidate individuals and the true Pareto optimal set (convergence) and (2) maximizing the distribution of candidate individuals (diversity). In practice, most existing MOEAs achieve fast convergence by prior Pareto-based sorting of the evolving population and high diversity by additional calculation of individuals density information, such as NSGA-II [4] and SPEA2 [5]. Such Pareto dominance based MOEAs actually select individuals first based on dominance relationship and then on density in both the mating and environmental selection. The dominance based selection favors non-dominated individuals to dominated ones so that the individuals in better fronts will be priorly preserved, which is helpful for a fast convergence
speed. However, the dominance based selection may undermine population diversity if well-converged individuals are not diversified or dominated individuals contribute considerably to population diversity.

Many Pareto-based MOEAs have been used to solve MoTCP problems, such as NSGA-II, SPEA2 and TAEA [6]. In addition to these Pareto dominance based algorithms, some indicator based MOEAs are also applied to MoTCP, such as the Hypervolume-based Genetic Algorithm (HGA) [7]. However, the performance of multi-objective algorithms may vary throughout the entire optimization process. The comparison among different multi-objective algorithms may vary according to different multi-objective algorithms [8], [9] indicates that there is no general applicable algorithm for different testing scenarios. In general, a multi-objective algorithm only performs well regarding both effectiveness and efficiency for specific testing scenarios.

Hyper-heuristics are proposed to generate or select heuristics automatically for computational search problems, including heuristic generation and heuristic selection [10]. Heuristic generation involves automatically generating the most appropriate heuristic for a given situation by combining the components of the existing heuristics [11]. Heuristic selection aims to select the suitable heuristic execution sequence for different situations rather than solve the problem directly [12]. Cowling et al. summarized several hyper-heuristic selection frameworks [13], [14], [15], [16] which are divided into two levels. The low level includes a set of heuristics focused on the certain kind of problems. The high level provides a strategy that can select the appropriate heuristics dynamically for different situations.

The idea of hyper-heuristics may solve the algorithm selection problem well for MoTCP. For this motivation, Yi Bian et al. proposed a concrete hyper-heuristic framework for MoTCP [17] with a solution repository in the low level and a learning agent in the high level. The low-level solution repository is composed of 18 multi-objective optimization algorithms. In the high level, the learning agent evaluates and selects the appropriate solutions dynamically for different testing scenarios based on Hierarchical Distribution (HD) strategy. More specifically, in each iteration of the execution process of the hyper-heuristic algorithm, the selected solution is applied to MoTCP and the execution results are transmitted to the high level. Then the learning agent uses a non-dominated sorting algorithm to divide the individuals from the present and last generations into different hierarchical sets, and the HD strategy evaluates the current solution based on the present individual distribution rates in different hierarchies. Finally, the most appropriate solution is selected for the next generation.

As mentioned above, all individuals are divided into several hierarchical sets in each iteration, in which the hierarchy of an individual represents its degree of convergence to the Pareto front. Since the individuals in the Pareto front are not dominated by any other individuals, it is general to consider that the further position away from the Pareto front an individual is, the worse its convergence is. However, although the individuals in each front are all non-dominated, their impacts on accelerating the convergence of population are still different. Various convergence-related metrics have been proposed in the literature, which can distinguish individuals more precisely, such as Pareto Dominance (PD) and Convergence Information (CI) [18]. Thus it’s not accurate enough to compare the convergence ability of the individuals based on their hierarchies, which makes that the HD strategy cannot evaluate solutions accurately. In this paper, the PD and CI metrics are adopted to measure the convergence ability of the individuals, and three convergence based evaluation strategies are proposed, denoted as ESPD, ESCI and ESPC, respectively. Experiments are conducted to prove the effectiveness of the proposed evaluation strategies, and the results show that the hyper-heuristic algorithms with the proposed evaluation strategies perform better regarding both effectiveness and efficiency for different subjects, which verifies that the novel evaluation strategies can improve the accuracy of evaluation results. In particular, ESPC strategy is most effective for the learning agent followed by ESCI and ESPD strategies.

The remainder of this paper is organized as follows: Section II introduces the research background. Section III describes the proposed evaluation strategies in detail. Section IV presents the experimental setup, optimization objectives for MoTCP and empirical results. Finally, conclusions and future work are given in Section V.

II. BACKGROUND

This section describes the research background in detail, including the definition of MoTCP, the selection criterion for MOEAs and the learning agent of the hyper-heuristic framework for MoTCP.

A. Multi-objective TCP

Single-objective TCP cannot satisfy testing requirements in general, which results in more focus on MoTCP recently. The definition of MoTCP [17] is as follows:

**Definition 1 (MoTCP)**

Given: a test suite, T; the set of permutations of T, PT; a vector of n objective functions, \( f_i, i = 1, 2, ..., n \).

Problem: to find \( PT' \subset PT \) such that \( PT' \) is a Pareto-optimal permutation set with respect to the objective functions, \( f_i, i = 1, 2, ..., m \).

In the context of a maximization problem, let two individuals \( P_x, P_y \in PT \). \( P_x \) is said to dominate \( P_y \) if and only if \( f_i(P_x) \geq f_i(P_y) \) for every \( i \) and \( f_j(P_x) > f_j(P_y) \) for at least one \( j \), \( 1 \leq i, j \leq m \). If neither \( P_x \) dominates \( P_y \) nor \( P_y \) dominates \( P_x \), the two individuals are non-dominated. If no individual can dominate \( P^* \), \( P^* \) is Pareto optimal. Since different optimization objectives often contradict each other, there is no single individual that can satisfy all optimization objectives simultaneously. Therefore a set of Pareto optimal individuals is expected which represents the best trade-off among different optimization objectives.
B. Selection Criterion for MOEAs

During the past two decades, researchers have proposed a variety of MOEAs, such as NSGA-II, SPEA2, IBEA [19] and MOEA/D [20]. A large number of selection criteria have been developed for the mating and environmental selection of MOEAs. For example, Pareto dominance based MOEAs compare individuals based on the dominance relationship to accelerate convergence toward the Pareto front, which have been shown to effectively solve multi-objective problems. Many dominance based selection criteria have been proposed, such as Pareto Dominance and Pareto Rank [21]. In addition, Pareto-based MOEAs also adopt a diversity-related metric as the secondary criterion to achieve high diversity. For many-objective problems, most of the individuals in a population are non-dominated in the early stage of the evolutionary process of such Pareto-based MOEAs. As a result, the dominance based criterion fails to distinguish individuals. Then a metric for population diversity will be used as the primary criterion to select individuals, which makes that the performance of Pareto-based MOEAs significantly deteriorates.

In order to enhance the convergence ability of MOEAs in many-objective optimization, various methods have been proposed, which can be divided into three categories. The first category is to introduce the novel concepts of Pareto dominance, such as \( L \)-optimality [22], fuzzy dominance [23] and grid dominance [24]. In the second category, one metric based on the traditional dominance relationship is adopted as the primary criterion, and the other for population convergence is used as the secondary criterion, such as substitute distance [25]. The third category is the indicator based MOEAs, which adopt a single performance indicator to measure the quality of the non-dominated sets regarding both convergence and diversity. For example, as the most popular performance indicator, the hypervolume indicator [26] is widely used to compare the performance of different MOEAs. What’s more, some other performance indicators have been proposed, such as GD [27], IGD [28] and R2 [29]. Apart from the above methods, there are many other convergence enhancement strategies. For example, many MOEAs solve many-objective problems by combining user preferences or reducing the number of objectives. Some reference set based MOEAs have been developed, which use the predefined reference set to guide the search process, such as NSGA-III [30]. In [18], Convergence Information (CI) and Distribution Information (DI) are adopted to obtain an even-distributed Pareto optimal set.

C. Learning Agent of Hyper-heuristic Framework for MoTCP

Despite the variety of multi-objective algorithms for MoTCP, they have been demonstrated to be only effective in specific testing scenarios. In fact, there are a wide range of testing scenarios for MoTCP. To select the appropriate algorithms for different testing scenarios, a hyper-heuristic framework for MoTCP has been proposed with two levels of structure.

The solution repository in the low level includes \( 18 \) MOEAs, which are constructed by combining three Pareto dominance based evolutionary algorithms with six crossover operators. In the high level, the HD strategy is designed for the learning agent to evaluate the low-level solution for a current generation and select the most appropriate solution for the next generation. As shown in Figure 1(a), the execution process of the high level includes the evaluation and solution decision. The domain barrier is an abstract interface used to transmit information between the low and high levels.

![Fig. 1. High level construction of the hyper-heuristic framework for MoTCP](image)

Solution evaluation evaluates the execution results of the solution applied in the current generation, i.e., the objective results of the individuals. In the evaluation process, by using a non-dominated sorting algorithm, all individuals are divided into several hierarchical sets. For example, in Figure 1(b), \( f_1 \) and \( f_2 \) are two maximization functions. Suppose \( H \) fronts are obtained after applying the non-dominated sorting algorithm, and \( \text{Front}_H \) is the Pareto front. The black and white points represent the individuals from the present and last generations, respectively.

An evaluation function for the HD strategy is given as follows:

\[
\text{HD} = \sum_{i=1}^{H} \frac{n_i}{N_i} \times i
\]

where \( n_i \) and \( N_i \) are the number of present and total individuals in the \( i \)th hierarchy, respectively. The evaluation function...
calculates the product of the present individual distribution rate and the hierarchy weight $i$, $i = 1, 2, ..., H$.

A higher HD value indicates that more present individuals are in higher fronts, i.e., more individuals from the present generation dominate the individuals from the last generation.

Solution decision determines the solution that will be used in the next generation. In the process of solution decision, due to the fluctuation of the HD value of each solution, the average HD is adopted to measure the quality of solutions. The solution with the largest average HD is selected and applied in the next generation.

III. THE PROPOSED EVALUATION STRATEGIES

In this paper, Pareto Dominance (PD) and Convergence Information (CI) are considered in the design of the evaluation strategy and three evaluation strategies are proposed for the learning agent of the hyper-heuristic framework for MoTCP, which are ESPD, ESCI and ESPC, respectively. This section describes the details of the three novel evaluation strategies.

A. ESPD

In the high level, a non-dominated sorting approach is used to divide individuals into different hierarchical sets. That is, the HD strategy compares individuals based on their hierarchies. Although the individuals in the same set are non-dominated and incomparable, there may be big difference among the convergence ability of these non-dominated individuals. This makes that the HD strategy hardly guarantees the validity of the comparison result between the present and last generations and thus cannot evaluate the quality of the low-level solutions accurately. In fact, some other metrics based on the dominance relationship can distinguish individuals more precisely such as Pareto Dominance (PD). Let $P$ denotes an individual, the number of individuals that dominate $P$ is the PD value of $P$. The lower the PD value is, the better the individual is. In general, more individuals than one have the same PD value in a population, and besides that, different non-dominated individuals have different PD values. For example, in Figure 1(b), all individuals in the $H$th hierarchy have a PD value of 0. In the $(H - 1)$th hierarchy, only one individual has a PD value of 1, and the others have a PD value of 2.

In this paper, we propose a PD based evaluation strategy ESPD. The ESPD strategy executes the procedure named Eliter_selection to obtain an elite population for each generation. The execution process of the Eliter_selection procedure is divided into two steps: (1) the PD values are calculated for the individuals from the present and last generations; (2) all individuals are sorted in ascending order according to their PD values. The former $N$ individuals compose the elite population, $N$ is the population size. The ESPD strategy evaluates the quality of solutions based on the present individual distribution for different PD values in the obtained elite population.

To present the ESPD strategy in more detail, an example is shown in Figure 2, in which $f_1$ and $f_2$ are two objective functions that need to be maximized. By executing the Eliter_selection procedure, the individuals with lower PD values are selected to form the elite population, including the present and last generations. The black points represent the individuals from the present generation and the white ones represent the individuals from the last generation. The number next to each individual represents its PD value.

We expect that more individuals from the present generation have lower PD values in the elite population, which indicates that the present generation is better than the last generation. As a result, the present individual distribution can be used to evaluate solutions. In addition, the number of different PD values represents the diversity of the individuals. More different PD values are demand to achieve higher diversity.

In ESPD strategy, an evaluation function is given as follows:

$$ESPD = \sum_{i=0}^{H} n_i \times \frac{H - i}{H}$$

where $H$ is the maximum PD value, and $n_i$ is the number of the elite individuals with a PD value of $i$ from the present generation. The evaluation function calculates the product of $n_i$ and the corresponding weight $(H - i)/H$, $i = 0, 1, ..., H$. The sum of $(H+1)$ products is the ESPD value of the solution for the present generation. Therefore, a higher ESPD value indicates that more elite individuals with lower PD values are from the present generation.

B. ESCI

In ESPD strategy, the individuals are distinguished by their PD values. A lower PD value indicates that the individual is closer to the Pareto front. The individuals with a PD value of 0 are Pareto optimal. The ESPD strategy is not able to distinguish the elite individuals with the same PD value. Therefore we can consider combining the PD metric with additional convergence-related metrics. So far, a variety of metrics have been proposed to obtain well-converged individuals. For example, Convergence Information (CI) measures the degree of convergence of an individual to the ideal point based on the sum of its objective values and thus all individuals are probably different from each other regarding the CI metric.
For this motivation, an evaluation strategy based on both PD and CI metrics is proposed in this paper, denoted as ESCI.

For MoTCP problems, the value ranges of different optimization objectives may differ significantly. To measure the convergence of an individual accurately, the normalized objective values are used in the calculation of CI. In this paper, the CI value of an individual $P$ is calculated by summing its normalized objective values:

$$CI(P) = \sum_{i=1}^{m} f_i(P)^\prime$$

where $m$ is the number of objectives, and $f_i(P)^\prime$ represents the $i$th normalized objective value of $P$. A higher CI value is expected for maximization problems. However, some objectives need to be maximized while others need to be minimized in real MoTCP problems. To be consistent with the conclusion for maximization problems, the normalized value of a minimization objective is obtained by subtracting its value from 1.

In each iteration, the ESCI strategy first obtains an elite population by executing the Eliter_selection procedure and then compares the present and last generations based on the CI metric. We expect that more elite individuals from the present generation have higher CI values, which indicates that the present generation outperforms the last generation.

In ESCI strategy, an evaluation function is given as follows:

$$ESCI = \sum CI_{new}$$

where $CI_{new}$ represents the CI value of an elite individual from the present generation. The ESCI value is calculated by summing the CI values of all elite individuals from the present generation. Therefore, a higher ESCI value indicates that more elite individuals with higher CI values are from the present generation.

C. ESPC

The ESPD and ESCI strategies distinguish the elite individuals by the PD and CI metrics, respectively. Therefore we can consider that the elite individuals with the same PD value are further distinguished by their CI values. For this motivation, we propose a variant of the ESPD and ESCI strategies, denoted as ESPC.

In each iteration, the ESPC strategy also executes the Eliter_selection procedure to obtain an elite population. The ESPC strategy evaluates solutions based on the CI values and distribution of the elite individuals in the present generation for different PD values.

If more elite individuals from the present generation have both higher CI values and lower PD values, the present generation is better than the last generation. Furthermore, the larger the number of different PD values is, the higher the diversity of the individuals is.

In ESPC strategy, an evaluation function is given as follows:

$$ESPC = \sum_{i=0}^{H} CI_i \cdot \frac{H - i}{H}$$

where $H$ is the maximum PD value, and $CI_i$ represents the sum of the CI values for the elite individuals with a PD value of $i$ from the present generation. The evaluation function calculates $CI_i$ multiplied by the corresponding weight $(H - i)/H$. The sum of $(H + 1)$ products is the ESPC value of the solution used in the present generation. Therefore, a higher ESPC value indicates that more elite individuals with both lower PD values and higher CI values are from the present generation.

IV. EMPIRICAL STUDIES AND ANALYSIS

To evaluate the effectiveness of the proposed evaluation strategies, we implement four hyper-heuristic algorithms for MoTCP, denoted as HH-HD, HH-PD, HH-CI and HH-PC, respectively. The low level of each hyper-heuristic framework contains 18 solutions, which are formed by the combinations of evolutionary algorithms and crossover operators. In the high level of HH-HD, the HD strategy is applied by the learning agent. Similarly, HH-PD, HH-CI and HH-PC apply the proposed ESPD, ESCI and ESPC strategies for the learning agent in the high level, respectively. Five Java subjects and three C++ subjects are used in the experiments.

A. Experimental Setup

In the experiments, 18 solutions in the low level of each hyper-heuristic framework are constructed by the combinations of three MOEAs (NSGA-II, TAEA and SPEA2) with six crossover operators (single-point crossover, anti-single-point crossover, scan crossover, anti-scan crossover, order crossover and uniform crossover).

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>SUBJECTS IN EXPERIMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>KLOC</td>
</tr>
<tr>
<td>Java</td>
<td></td>
</tr>
<tr>
<td>commons-io</td>
<td>24.4</td>
</tr>
<tr>
<td>commons-lang</td>
<td>63.3</td>
</tr>
<tr>
<td>joda-time</td>
<td>70.3</td>
</tr>
<tr>
<td>commons-math</td>
<td>202.8</td>
</tr>
<tr>
<td>camel-core</td>
<td>215.4</td>
</tr>
<tr>
<td>C++</td>
<td></td>
</tr>
<tr>
<td>flex</td>
<td>3.0</td>
</tr>
<tr>
<td>bash</td>
<td>6.2</td>
</tr>
<tr>
<td>V8</td>
<td>680.4</td>
</tr>
</tbody>
</table>

Table I presents the details of the subjects used in the experiments, including five Java programs from Github and three C++ programs from a benchmark repository. Each Java program has more than 20 kilo lines of code (KLOC). Except commons-io, all Java programs have more than two thousand associated test cases. The largest Java program Camel has many sub-projects and thus the main part of the source code is selected in the experiments. For the three C++ programs, flex and bash from the Software-artifact
Infrastructure Repository (SIR) have more than 1 KLOC of effective code and thousands of associated test cases. Google Chrome, Safari and other projects use V8 with more than 600 KLOC of source code as a JavaScript engine.

In all solutions, the population size, crossover rate and mutation rate are set to 200, 0.01 and 0.001, respectively. The termination condition of each hyper-heuristic algorithm is that the difference of the average values for all optimization objectives is less than $10^{-4}$ in 10 continuous generations or the iteration reaches 10,000 times. Furthermore, due to the nature of commons-lang, commons-math and camel-core, the hyper-heuristic algorithms require very high time cost to find the optimal test sequences. Therefore, the number of generations is set to 100 for these three programs.

B. Optimization Objectives for MoTCP

In our study, Average Percentage Statement Coverage (APSC) and Effective Execution Time (EET) are selected as the optimization objectives in MoTCP to guide the evolutionary process of the hyper-heuristic algorithms.

APSC measures the rate at which a test sequence covers the statements [2], the definition of which is given as follows:

$$APSC = 1 - \frac{TS_1 + TS_2 + \ldots + TS_M}{NM} + \frac{1}{2N} \tag{6}$$

where $N$ is the number of test cases, $M$ is the number of statements, and $TS_i$ represents the index of the test case that is applied to statement $i$ for the first time in the test sequence.

EET measures the total execution cost of a test sequence, which is defined as follows:

$$EET = \sum_{i=0}^{N_{\text{length}}} ET_i \tag{7}$$

where $ET_i$ is the execution time of the $i$th test case, and $N_{\text{length}}$ is the number of the executed test cases when 100% of effective statement coverage is achieved.

C. Results and Discussion

To evaluate the effectiveness and efficiency of HH-HD, HH-PD, HH-Cl and HH-PC, we design the following three research questions:

**RQ1:** In terms of APSC and EET, is the hyper-heuristic algorithm with the proposed evaluation strategies more effective than the original HH-HD for MoTCP?

**RQ2:** In terms of hypervolume and IGD, is the hyper-heuristic algorithm with the proposed evaluation strategies more effective than the original HH-HD for MoTCP?

**RQ3:** Is the hyper-heuristic algorithm with the proposed evaluation strategies more efficient than the original HH-HD for MoTCP?

To answer the above questions, HH-HD, HH-PD, HH-Cl and HH-PC are independently executed 30 times to guarantee the validity of the results.

1) Effectiveness of Hyper-heuristic Algorithms regarding APSC and EET:

Table II presents the average APSC and EET for HH-HD, HH-PD, HH-Cl and HH-PC. In Table II, for all subjects, HH-PC achieves the highest APSC results, followed by HH-Cl, HH-PD and HH-HD except for V8. The APSC result of HH-Cl is highest followed by HH-PD, HH-PC and HH-HD in V8. When considering EET, HH-PC performs best in flex, commons-lang, commons-math and V8, followed by HH-Cl, HH-PD and HH-HD. HH-PC achieves the shortest average EET value followed by HH-PD, HH-Cl and HH-HD in joda-time. The EET results for commons-lang, camel-core and bash show that HH-Cl and HH-PC are better than HH-HD while HH-HD is better than HH-PD. Furthermore, the EET result of HH-Cl is shorter than HH-PC in commons-lang, but not in camel-core and bash. In conclusion, the hyper-heuristic algorithms with the proposed evaluation strategies perform better than HH-HD, as they can obtain better test execution sequences with higher APSC values and shorter EET values. Among our hyper-heuristic algorithms, HH-PC has the best overall performance followed by HH-Cl and HH-PD regarding both APSC and EET.

2) Effectiveness of Hyper-heuristic Algorithms regarding Hypervolume and IGD:

In order to further compare HH-HD, HH-PD, HH-Cl and HH-PC, we adopt the hypervolume and Inverted General Distance (IGD) indicators to measure the quality of the Pareto fronts produced by the four hyper-heuristic algorithms. An algorithm with lower hypervolume and IGD values is more competitive.

Table III summarizes the average hypervolume in 30 running times for HH-HD, HH-PD, HH-Cl and HH-PC. In Table III, for all subjects, the average hypervolume values of HH-PD, HH-Cl and HH-PC are lower than HH-HD, which means that the hyper-heuristic algorithms with the proposed evaluation strategies are more effective for MoTCP problems. Among our hyper-heuristic algorithms, HH-PC has the lowest hypervolume results for all subjects, followed by HH-Cl and HH-PD. In conclusion, on average, HH-PC has the best performance followed by HH-Cl, HH-PD and HH-HD regarding the hypervolume indicator.

Figure 3 presents the hypervolume values in 30 running times that are sorted in ascending order for HH-HD, HH-PD, HH-Cl and HH-PC. The three highest hypervolume values for HH-HD (much higher than the hypervolume values of the other hyper-heuristic algorithms) are not shown in Figure 3(c).

In Figure 3(a), (c)-(g), HH-PC has the best overall performance followed by HH-Cl, HH-PD and HH-HD, as most of the time the hypervolume values of HH-PC are lowest followed by HH-Cl, HH-PD and HH-HD. In Figure 3(b), the lines of HH-Cl and HH-PC are almost coincident when the rank value is less than 18, and the lines of HH-HD and HH-PD are also almost coincident when the rank value is less than 22. Overall speaking, HH-PC achieves the lowest hypervolume result followed by HH-Cl, HH-PD and HH-HD. That is, HH-PC performs best followed by HH-Cl, HH-PD and HH-HD.
TABLE II
Comparison results of the average APSC and EET for HH-HD, HH-PD, HH-CI and HH-PC.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>HH-HD</th>
<th>HH-PD</th>
<th>HH-CI</th>
<th>HH-PC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APSC EET</td>
<td>APSC EET</td>
<td>APSC EET</td>
<td>APSC EET</td>
</tr>
<tr>
<td>commons-io</td>
<td>0.1424A</td>
<td>67.9163</td>
<td>93.1455</td>
<td>67.4391</td>
</tr>
<tr>
<td>commons-lang</td>
<td>98.8080%</td>
<td>13.6345</td>
<td>93.1424</td>
<td>67.4391</td>
</tr>
<tr>
<td>joda-time</td>
<td>98.8080%</td>
<td>13.6345</td>
<td>93.1424</td>
<td>67.4391</td>
</tr>
<tr>
<td>commons-math</td>
<td>89.1194%</td>
<td>3536.03</td>
<td>89.7356</td>
<td>3538.91</td>
</tr>
<tr>
<td>camel-core</td>
<td>91.7302%</td>
<td>5832.75</td>
<td>92.1104</td>
<td>5832.75</td>
</tr>
<tr>
<td>flex</td>
<td>99.8447%</td>
<td>4.30774</td>
<td>99.8447</td>
<td>4.30774</td>
</tr>
<tr>
<td>bash</td>
<td>98.1808%</td>
<td>13.6345</td>
<td>98.1808</td>
<td>13.6345</td>
</tr>
<tr>
<td>V8</td>
<td>99.9451%</td>
<td>0.08816</td>
<td>99.9451</td>
<td>0.08816</td>
</tr>
</tbody>
</table>

In Table IV, MWU represents the p-value of MWU test, and VDA represents the \( A_{12} \) estimate value. In the comparison results between our hyper-heuristic algorithms and HH-HD, all the values of MWU for flex are less than 0.05, the values of MWU for HH-CI and HH-PC are also less than 0.05 in commons-io, commons-math and camel-core, and only the value of MWU for HH-PC is less than 0.05 in commons-lang. The remaining values of MWU are all greater than 0.05. In conclusion, for commons-io, commons-math, camel-core and flex, HH-CI and HH-PC are significantly different from HH-HD in hypervolume distribution, the difference in hypervolume distribution between HH-PC and HH-HD is also significant in commons-lang, and besides that, the hypervolume distributions of HH-HD and HH-PD are non-identical in flex, but not in other seven subjects. From the \( A_{12} \) estimate results, we can find that HH-PD, HH-CI and HH-PC are better than HH-HD, as all the \( A_{12} \) estimate values are greater than 0.5.

In the comparison results among our hyper-heuristic algorithms, the values of MWU for HH-PD in comparison with HH-PC are less than 0.05 in commons-io and camel-core, and the value of MWU for HH-PD in comparison with HH-CI is also less than 0.05 in commons-io. The remaining values of MWU are all greater than 0.05. In conclusion, for commons-io and camel-core, HH-PD is significantly different from HH-PC in hypervolume distribution, the hypervolume distributions of HH-PD and HH-CI are non-identical in commons-io, and besides that, the

TABLE III
Comparison results of the average hypervolume for HH-HD, HH-PD, HH-CI and HH-PC.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>HH-HD</th>
<th>HH-PD</th>
<th>HH-CI</th>
<th>HH-PC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APSC EET</td>
<td>APSC EET</td>
<td>APSC EET</td>
<td>APSC EET</td>
</tr>
<tr>
<td>commons-io</td>
<td>0.111571</td>
<td>0.085703</td>
<td>0.049468</td>
<td>0.064706</td>
</tr>
<tr>
<td>commons-lang</td>
<td>0.188543</td>
<td>0.141425</td>
<td>0.126550</td>
<td>0.109645</td>
</tr>
<tr>
<td>joda-time</td>
<td>0.046234</td>
<td>0.019421</td>
<td>0.015170</td>
<td>0.037938</td>
</tr>
<tr>
<td>commons-math</td>
<td>0.208748</td>
<td>0.160421</td>
<td>0.141425</td>
<td>0.018329</td>
</tr>
<tr>
<td>camel-core</td>
<td>0.226664</td>
<td>0.160421</td>
<td>0.158773</td>
<td>0.109179</td>
</tr>
<tr>
<td>flex</td>
<td>0.059584</td>
<td>0.064706</td>
<td>0.046234</td>
<td>0.04654</td>
</tr>
<tr>
<td>bash</td>
<td>0.128518</td>
<td>0.109645</td>
<td>0.090472</td>
<td>0.082427</td>
</tr>
<tr>
<td>V8</td>
<td>0.023138</td>
<td>0.015170</td>
<td>0.009989</td>
<td>0.06687</td>
</tr>
</tbody>
</table>

In Figure 3(b), looking at Figure 3(h), the lines of HH-HD, HH-PD, HH-CI and HH-PC are almost coincident when the rank value is less than 23. Furthermore, when the rank value is more than 25, HH-PC achieves the lowest hypervolume values followed by HH-CI, HH-PD and HH-HD. Therefore, HH-PC has the best overall performance followed by HH-CI, HH-PD and HH-HD in Figure 3(h). In conclusion, the hyper-heuristic algorithms with the proposed evaluation strategies can solve MoTCP problems more effectively, and HH-PC performs best overall followed by HH-CI and HH-PD.

In order to study the significance of the differences in hypervolume, two statistical analysis results are presented in Table IV for pairwise comparisons among HH-HD, HH-PD, HH-CI and HH-PC. The Mann-Whitney-U (MWU) test with the 5% level of significance is adopted to evaluate the difference between the distributions of the hypervolume results for a pair of hyper-heuristic algorithms. A p-value that is less than 0.05 indicates that we can reject the hypothesis that the hypervolume distributions for the two compared hyper-heuristic algorithms are identical. The Vargha and Delaney \( A_{12} \) (VDA) measure [31] is adopted to show the probability that the hypervolume result of one hyper-heuristic algorithm is lower than another. A estimate of \( A_{12} \) is equal to 0.5 indicates that the difference between two sets of hypervolume values is not significant. If the \( A_{12} \) estimate value is greater than 0.5, the first set is lower than the second set, i.e., the first algorithm is better than the second one. The exact opposite occurs if the \( A_{12} \) estimate value is less than 0.5.
differences in hypervolume distribution between HH-CI and HH-PC are not significant for all subjects. Overall speaking, there are not significant differences among our hyper-heuristic algorithms in hypervolume distribution. What’s more, the estimate values for HH-PC in comparison with HH-CI are less than 0.5 in commons-math and V8, and the remaining estimate values are all greater than 0.5. Based on the above observation, HH-PC has the best overall performance followed by HH-CI and HH-PD.

Table V summarizes the average IGD in 30 running times for HH-HD, HH-PD, HH-CI and HH-PC. In Table V, the average IGD values of HH-PD, HH-CI and HH-PC are lower than HH-HD for all subjects. Among our hyper-heuristic algorithms, for all subjects, HH-PC achieves the lowest IGD results followed by HH-CI and HH-PD except for joda-time and V8. The IGD results of HH-CI are lower than HH-PD and HH-PC in joda-time and V8. Furthermore, the IGD result of HH-PC is lower than HH-PD in joda-time, but not in V8. In conclusion, on average, HH-PC has the best overall performance followed by HH-CI, HH-PD and HH-HD regarding the IGD indicator.

Figure 4 presents the box-plots diagrams for all subjects which show the IGD distributions of HH-HD, HH-PD, HH-CI and HH-PC.
TABLE IV
Comparison between different pairs of hyper-heuristic algorithms in hypervolume based on MWU and VDA.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>HH-PD vs HH-HD</th>
<th>HH-CI vs HH-HD</th>
<th>HH-PC vs HH-HD</th>
<th>HH-CI vs HH-PC</th>
<th>HH-PC vs HH-CI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MWU</td>
<td>VDA</td>
<td>MWU</td>
<td>VDA</td>
<td>MWU</td>
</tr>
<tr>
<td>commons-io</td>
<td>0.416</td>
<td>0.561</td>
<td>0.007</td>
<td>0.703</td>
<td>0.001</td>
</tr>
<tr>
<td>commons-lang</td>
<td>0.690</td>
<td>0.530</td>
<td>0.209</td>
<td>0.594</td>
<td>0.037</td>
</tr>
<tr>
<td>joda-time</td>
<td>0.802</td>
<td>0.519</td>
<td>0.337</td>
<td>0.572</td>
<td>0.071</td>
</tr>
<tr>
<td>commons-math</td>
<td>0.657</td>
<td>0.533</td>
<td>0.015</td>
<td>0.682</td>
<td>0.037</td>
</tr>
<tr>
<td>camel-core</td>
<td>0.095</td>
<td>0.626</td>
<td>0.022</td>
<td>0.672</td>
<td>0.000</td>
</tr>
<tr>
<td>flex</td>
<td>0.041</td>
<td>0.654</td>
<td>0.010</td>
<td>0.694</td>
<td>0.003</td>
</tr>
<tr>
<td>bash</td>
<td>0.478</td>
<td>0.553</td>
<td>0.147</td>
<td>0.602</td>
<td>0.062</td>
</tr>
<tr>
<td>V8</td>
<td>0.965</td>
<td>0.303</td>
<td>0.193</td>
<td>0.598</td>
<td>0.626</td>
</tr>
</tbody>
</table>

In Figure 4(a), the upper quartiles and medians of HH-PD, HH-CI and HH-PC are much lower than HH-HD, so our hyper-heuristic algorithms are better than HH-HD. Furthermore, HH-PC with the lowest median performs best followed by HH-CI and HH-PD. In Figure 4(b) and (g), HH-PC has the best performance, as the upper quartile and median of HH-PC are much lower than HH-HD, HH-PD and HH-CI. In addition, in Figure 4(b), the performances of HH-PD and HH-CI are similar, while they outperform HH-HD with lower medians. As shown by Figure 4(g), HH-CI with the lowest median performs best followed by HH-PD and HH-HD. In Figure 4(c), HH-PD with lower median is better than HH-HD. What’s more, the upper quartiles, medians and lower quartiles of HH-CI and HH-PC are much lower than HH-HD and HH-PD, and the medians of HH-CI and HH-PC are similar. Therefore, in Figure 4(c), HH-CI and HH-PC are better than HH-PD, and besides that, HH-CI with shorter IQR has more stable performance than HH-PC. Looking at Figure 4(d), (e) and (h), the upper quartiles, medians and lower quartiles of HH-PD, HH-CI and HH-PC are much lower than HH-HD, so our hyper-heuristic algorithms are better than HH-HD. To be more specific, in Figure 4(d), HH-CI and HH-PC with lower medians perform better than HH-PD. Furthermore, HH-CI has higher median but much shorter IQR than HH-PC, so HH-CI has more stable performance than HH-PC. In Figure 4(e), the performances of HH-PD and HH-CI are similar, and HH-PC with much lower median is better than HH-PD and HH-CI. In Figure 4(h), HH-CI with the lowest median performs best followed by HH-PD and HH-PC. Figure 4(f) shows that our hyper-heuristic algorithms are better than HH-HD, as the upper quartiles and medians of HH-PD, HH-CI and HH-PC are much lower than or similar to HH-HD. In addition, HH-CI and HH-PC with lower medians are better than HH-PD, and HH-PC with similar median and shorter IQR performs better than HH-CI. Based on the above analysis, we can conclude that HH-PC has the best overall performance for MoTCP problems followed by HH-CI, HH-PD and HH-HD.

In order to study the significance of the differences in IGD, two statistical analysis results are presented in Table VI for pairwise comparisons among HH-HD, HH-PD, HH-CI and HH-PC. Based on the comparison results between our hyper-heuristic algorithms and HH-HD, the observations are as follows: all the values of MWU for commons-math and camel-core are less than 0.05; the values of MWU for HH-CI and HH-PC are less than 0.05 in commons-io; for flex and commons-lang, the values of MWU for HH-PC are less than 0.05; the values of MWU for HH-PD and HH-CI are less than 0.05 in V8; and the remaining values of MWU are all greater than 0.05. In conclusion, in IGD distribution, both HH-PD and HH-CI are significantly different from HH-HD in commons-math, camel-core and V8, and the difference between HH-CI and HH-HD is also significant in commons-io. In addition, there are significant differences between the IGD distributions of HH-PC and HH-HD in flex, commons-io, commons-lang, commons-math and camel-core. Considering the $\hat{A}_{12}$ estimate results, it
appears that our hyper-heuristic algorithms are better than HH-HD, as all the $\hat{A}_{12}$ estimate values are greater than 0.5.

Based on the comparison results among our hyper-heuristic algorithms, we have the following observations: the values of MWU for HH-CI and HH-PC in comparison with HH-PD are less than 0.05 in commons-io; the values of MWU for HH-PC in comparison with HH-PD are less than 0.05 in commons-lang and commons-math; and the remaining values of MWU are all greater than 0.05. In conclusion, for commons-io, HH-PD is significantly different from HH-CI in IGD distribution, the IGD distributions of HH-PD and HH-PC are non-identical in commons-io, commons-lang and commons-math, and besides that, the differences in IGD distribution between HH-CI and HH-PC are not significant for all subjects. Overall speaking, there are not significant differences among our hyper-heuristic algorithms in IGD distribution. In the $\hat{A}_{12}$ estimate results, the $\hat{A}_{12}$ estimate values for HH-PC in comparison with HH-CI are less than 0.5 in joda-time and V8, the $\hat{A}_{12}$ estimate value for HH-PC in comparison with HH-PD is also less than 0.5 in V8, and the remaining $\hat{A}_{12}$ estimate values are all greater than 0.5. Based on that we can observe, HH-PC performs best overall followed by HH-CI and HH-PD.

3) Efficiency of Hyper-heuristic Algorithms:
Table VII summarizes the average iteration number and execution time for HH-HD, HH-PD, HH-CI and HH-PC.

Combining with the above analysis of the effectiveness of the hyper-heuristic algorithms, the results of HH-HD and HH-PD show that HH-PD is more efficient than HH-HD, as HH-PD requires fewer or same iterations and less execution time to obtain better APSC and EET results. More specifically, the time cost is reduced by dozens or even hundreds of seconds in other subjects. In particular, the time cost is reduced by 423.7s in HH-PC.

For flex, bash and commons-io, HH-CI requires more iterations and thus longer time to obtain better results of the two objectives than HH-HD, and the time cost is increased by at most 1.6s, which means that HH-CI has stronger search capability. For all Java subjects except commons-io, HH-CI uses more or same iterations but less running time to obtain higher quality results, and the time cost is reduced by dozens or even hundreds of seconds. In particular, the time cost is reduced by 423.7s in HH-PC.

When considering the efficiency of HH-PC, the results for flex and commons-io show that HH-PC has stronger search capability than HH-HD, as HH-PC requires more iterations for the execution with longer time and achieves better APSC and EET. The time cost is only increased by several seconds in flex and commons-io. The results for bash, commons-lang and camel-core show that HH-PC can select more efficient low-level solutions, as HH-PC uses more or same iterations to achieve better APSC and EET within less running time. To be more specific, the time cost is only reduced by 2.2s in bash, while the reduction of time cost is hundreds of seconds in commons-lang and camel-core. Furthermore, for other three subjects, HH-PC requires fewer iterations but longer time, and the time cost is increased by tens of seconds.

In conclusion, most of the time, HH-PD, HH-CI and HH-PC only require less execution time to obtain better test case sequences. As a result, the hyper-heuristic algorithms with the proposed evaluation strategies are more efficient for MoTCP problems than HH-HD.

**V. CONCLUSION AND FUTURE WORK**

In this paper, in order to evaluate solutions more accurately, Pareto Dominance (PD) and Convergence Information (CI) are adopted to compare individuals in a population in terms of convergence, including the present and last generations. Three evaluation strategies based on the PD and CI metrics are proposed for the learning agent of the hyper-heuristic framework for MoTCP, i.e., ESPD, ESCI and ESPC. Experiments are conducted on some industrial applications in Java.
and C++ to compare the effectiveness of the hyper-heuristic algorithms with different evaluation strategies, i.e., HH-HD, HH-PD, HH-CI and HH-PC, in terms of two optimization objectives for MoTCP (APSC and EET) and two performance indicators (hypervolume and IGD). Furthermore, the efficiency of the four hyper-heuristic algorithms is analyzed in terms of iterations and execution time. The experimental results show that the hyper-heuristic algorithms with the proposed evaluation strategies are more effective and efficient for MoTCP problems than HH-HD, as most of the time our hyper-heuristic algorithms can obtain better test case execution sequences within shorter time. In particular, HH-PC performs best overall followed by HH-CI and HH-PD regarding effectiveness aspect.

In conclusion, the proposed evaluation strategies can evaluate the low-level solutions more accurately and thus the learning agent can select more appropriate solution execution sequence for different testing scenarios.

In the future, a better selection strategy should be considered for the learning agent, instead of the rule that simply selects the solution with the maximum evaluation function value for the next generation. For the future work, different algorithms and crossover operators should be combined to determine the best solution repository for the low level. Furthermore, more optimization objectives and industrial applications can be used in the experiments to further verify the effectiveness of the proposed evaluation strategies.

ACKNOWLEDGMENTS

The work described in this paper is supported by the National Natural Science Foundation of China under Grant No.61872026, 61672085 and 61702029, and the Fundamental Research Funds for the Central Universities(XK1802-4).

REFERENCES


