A multi-objective hyper-heuristic algorithm based on adaptive epsilon-greedy selection

Tailong Yang¹ · Shuyan Zhang¹ · Cuixia Li¹

Received: 5 July 2020 / Accepted: 3 November 2020 / Published online: 3 January 2021
© The Author(s) 2021

Abstract

A variety of meta-heuristics have shown promising performance for solving multi-objective optimization problems (MOPs). However, existing meta-heuristics may have the best performance on particular MOPs, but may not perform well on the other MOPs. To improve the cross-domain ability, this paper presents a multi-objective hyper-heuristic algorithm based on adaptive epsilon-greedy selection (HH_EG) for solving MOPs. To select and combine low-level heuristics (LLHs) during the evolutionary procedure, this paper also proposes an adaptive epsilon-greedy selection strategy. The proposed hyper-heuristic can solve problems from varied domains by simply changing LLHs without redesigning the high-level strategy. Meanwhile, HH_EG does not need to tune parameters, and is easy to be integrated with various performance indicators. We test HH_EG on the classical DTLZ test suite, the IMOP test suite, the many-objective MaF test suite, and a test suite of a real-world multi-objective problem. Experimental results show the effectiveness of HH_EG in combining the advantages of each LLH and solving cross-domain problems.

Keywords Hyper-heuristics · Multi-objective optimization · Adaptive epsilon-greedy selection strategy · Cross-domain problems

Introduction

Multi-objective optimization problems (MOPs) with m objectives can be described as $\text{Min } F(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$, $\mathbf{x} \in \Omega$ [29]. The solution $\mathbf{x}$ of an MOP is an n -dimensional decision vector $\mathbf{x} = (x_1, \dots, x_n)$, and $F(\mathbf{x})$ is the phenotype of the solution $\mathbf{x}$ in the objective space. A solution $\mathbf{u} = (u_1, \dots, u_n)$ Pareto dominates another solution $\mathbf{v} = (v_1, \dots, v_n)$ if and only if the objective value of $\mathbf{u}$ is less than or equal to that of $\mathbf{v}$ on all objectives, and the objective value of $\mathbf{u}$ is less than that of $\mathbf{v}$ on at least one objective. Solutions that are not dominated by any solution are called non-dominated solutions. The set of all non-dominated solutions is called the Pareto

optimal set (PS), and the mapping of the Pareto optimal set in the objective space is called the true Pareto front (PF).

For NP-hard problems, researchers often design customized heuristics to obtain reasonable solutions within a polynomial time. When solving MOPs, the commonly used heuristics are multi-objective evolution algorithms (MOEAs) [5]. One of the major advantages of MOEAs could be that they are population-based algorithms which can obtain a set of solutions approximating the PF in a single run. However, MOEAs sometimes can obtain high performance for several problems but may be ineffective when solving other problems or problems from similar domains [35]. For instance, some dominance-based MOEAs are only effective for MOPs with less than 4 objectives [33]; the decomposition-based MOEAs using fixed reference vectors are not good at solving MOPs with irregular PFs [13, 14].

A hyper-heuristic (HH) is a methodology which has with cross-domain search abilities. This methodology is a high-level strategy by controlling a set of low-level heuristics (LLHs). Instead of solving the problem directly, at each decision point, an HH tries to select one LLH and apply the selected LLH to solve the problem at hand. If an HH is applied

✉ Shuyan Zhang
syzhang@zzu.edu.cn

Tailong Yang
tailongyangzhu@163.com

Cuixia Li
lcxxcl@zzu.edu.cn

¹ School of Software, Zhengzhou University, No.97 Wenhua Street, Jinshui Zone, Zhengzhou 450002, People's Republic of China

to another problem, the high-level strategy does not need to be redesigned [1].

This study proposes an adaptive epsilon-greedy selection based multi-objective hyper-heuristic algorithm (HH_EG) that applies multiple MOEAs as LLHs. In each iteration, HH_EG uses an online-learning strategy to score each LLH and utilizes an adaptive epsilon-greedy selection strategy to select one LLH based on the scores. The selected LLH is then used to solve the problem in this iteration. Specifically, the contribution of this paper consists of the following aspects:

1. An adaptive epsilon-greedy selection strategy is proposed for the hyper-heuristic algorithm to select LLH when solving MOPs. At early stages, the adaptive epsilon-greedy selection strategy prefers to select LLHs randomly while at later stages, the adaptive strategy tends to select the current best LLH using a greedy strategy. The adaptive method updates the probability of using random or greedy algorithms to balance exploration and exploitation.
2. In HH_EG, no parameter is needed to be tuned. Thus, HH_EG could be combined with various indicators to adapt to the preference of users.
3. HH_EG is tested on multiple test suites. We evaluate HH_EG with NSGA-II [9], MOEA/D [37] and, IBEA [40] as LLHs on classical DTLZ [10] and the real-world vehicle crashworthiness problem (VC) [20] test suites. We also conducted an experiment on a challenging IMOP [32] test suite by altering LLHs with NSGA-II, MOEA/D and MSEA [34]. Additionally, the MaF [3] test suite with 5 and 10 objectives are also tested with NSGA-III [8], MOEA/D and MSEA as LLHs. Experimental results show the advantages of HH_EG and its cross-domain search abilities.

Related work

Multi-objective hyper-heuristics

A hyper-heuristic algorithm is an automated methodology for selecting or generating heuristic algorithms to solve computational search problems [1]. There are two main aspects to be considered in the search process of hyper-heuristics: (1) the nature of heuristic search space; (2) the feedback during learning. These factors are orthogonal because heuristic search spaces can be combined with different feedbacks [1]. In terms of the nature of heuristic search space, hyper-heuristics can be divided into selection based hyper-heuristics (methodologies for choosing or selecting existing heuristics) and generation based hyper-heuristics (methodologies for generating new heuristics from components of existing heuristics). In terms of the feedback,

hyper-heuristics can be divided into online learning hyper-heuristics, offline learning hyper-heuristics and non-learning hyper-heuristics. In the online learning hyper-heuristics, the learning process occurs at the same time as the algorithm solves the problem instance. The most common example is reinforcement learning [19, 36]. In the offline learning hyper-heuristics, the learning process collects knowledge from a set of training examples in the form of a certain rule or procedure, which can be generalized to solve new examples. Typical offline learning strategies are classifier systems [26], case-based reasoning [27], and genetic programming [15]. The non-learning hyper-heuristics, such as random hyper-heuristics (choosing LLHs randomly), do not deal with the feedback.

In recent years, many studies on hyper-heuristics for solving MOPs have been proposed. Many of these studies focus on online-learning selection hyper-heuristics. In [2], a multi-objective selection hyper-heuristic (TSRoulette Wheel) based on tabu search was proposed. In TSRoulette Wheel, tabu search is used to select LLHs. Experimental results show that TSRoulette Wheel has a good effect on the space allocation and timetabling problems. McClymont and Keedwell [24] proposed an online learning selection hyper-heuristic algorithm (MCHH) based on the Markov chain. Markov chain guides the selection of LLHs, and an online reinforcement learning updates the transfer weights between heuristics. When solving the DTLZ test suite, MCHH performs better than the random hyper-heuristic (HH_RAND) and the TSRoulette Wheel. Subsequently, MCHH was applied to a water distribution networks design problem [25]. In [23], an online learning multi-objective hyper-heuristic algorithm based on choice function (HH_CF) was proposed. HH_CF uses the choice function as the selection method, while employing four indicators and a specific ranking strategy as the learning and feedback strategies. HH_CF uses three well-known MOEAs (NSGA-II, SPEA2 [41], and MOGA [11]) as LLHs. Experiments confirm the ability of HH_CF when solving continuous MOPs. Later, HH_CF is combined with different move acceptance strategies, including the great deluge algorithm and late acceptance in [22].

Li et al. [18] used a multi-objective selection hyper-heuristics to solve the multi-objective wind farm layout optimization problem. Later, Li et al. [19] proposed two novel online learning multi-objective hyper-heuristics (HH_LA and HH_RILA) based on learning automata. Learning automata is used as the learning strategy, and ϵ -RouletteGreedy is proposed to select LLHs from three MOPs (NSGA-II, SPEA2, and IBEA). Experiments on DTLZ, WFG and VC test suites showed that HH_LA and HH_RILA perform better than HH_CF and HH_RAND. Gonçalves et al. [12] used an innovative Restless Multi-Armed Bandit to expand the MOEA/D framework and

proposed a multi-objective selection hyper-heuristic algorithm MOEA/D-RMAB. The RMAB strategy is used to select DE operator which should be used by each individual during the operation of the MOEA/D framework to model and process the dynamic behavior of various operators. Yao et al. [36] proposed multi-objective hyper-heuristic algorithms RL-MOHH and RL-PMOHH based on reinforcement learning. By learning and selecting a set of individually designed LLHs, the algorithm is tested on the safety-index map constructed from the historical data of New York City. The results show that the proposed hyper-heuristics have faster convergence speed on the multi-objective route planning in a smart city. Zhang et al. [38] presented a perturbation adaptive pursuit strategy based hyper-heuristic (PAPHH) to manage a set of diversity mechanisms or a set of MOEAs. This method uses a perturbation adaptive pursuit strategy to learn the performance of each LLH, and then use the roulette wheel method to select an LLH based on the selection probability vector. Experimental results show that PAPHH has highly competitive performance in terms of Spacing metric and Hypervolume metric.

Epsilon-greedy

The selection method is important for multi-objective selection hyper-heuristics when taking an action (i.e., selecting an LLH). Several classical selection methods, such as the random algorithm or the greedy algorithm, have been used in the literature. Those methods are different from the function of selection probabilities. A random algorithm takes an action with the same probabilities. This method is easy and has no extra parameter to be considered. The greedy algorithm only selects the best action which is with the highest probabilities. The greedy method has a strong exploitation ability, but always neglects ‘poor’ actions which may perform good in the future. The greedy method easily falls into the local optima.

The Epsilon-greedy method combines the random algorithm and the greedy algorithm to deal with the exploration and exploitation dilemma [16, 28]. The main idea of the Epsilon-greedy is to control the utilization rate of the greedy algorithm or the random algorithm through a small probability ϵ (smaller than 1) with the aim to make the behavior of the Epsilon-greedy to be greedy most of the time, but random once in a while. Specifically, when selecting an action at one decision point, the Epsilon-greedy method first generates a random value rnd in the interval $[0, 1]$, if rnd is larger than ϵ , then the Epsilon-greedy uses the greedy algorithm to select an action, otherwise, the random algorithm is used to select an action.

Adaptive epsilon-greedy selection based multi-objective hyper-heuristic

Framework of HH_EG

HH_EG is a multi-objective selection hyper-heuristic within $iter_total$ iterations. During each iteration (also called a decision point), an LLH is selected to solve the problem at hand. As shown in Algorithm 1, given a set of LLHs, denoted as $H = \{h_1, h_2, \dots, h_l\}$, HH_EG firstly gets an initial population Pop of size N and initializes the score vector $S = (s_1, s_2, \dots, s_l)$, where s_i is the score of h_i , $i \in \{1, \dots, l\}$ (step 1). At each iteration (steps 2–9), HH_EG selects an LLH h_{cur} according to the score vector (step 3) and then uses h_{cur} to evolve the population Pop to generate a new population Pop (step 4). The score vector S is updated according to the degree of improvement after applying h_{cur} (step 5). If the population $'Pop$ is better than Pop , then Pop is updated (steps 6–8). If the termination criterion is not met, the algorithm goes to step 2.

The framework consists of four key components: initialization (step 1), adaptive epsilon-greedy selection (step 3), score update (step 5) and move acceptance (step 6).

Algorithm 1. HH_EG framework

```

1:  $[S, Pop] = \text{Initialise}(H)$ ;

2: while  $iter < iter\_total$  do

3:    $h_{cur} = \text{EG\_SelectLLH}(S, H)$ ;

4:    $Pop' = \text{runLLH}(h_{cur}, Pop)$ ;

5:    $S = \text{UpdateScore}(h_{cur}, Pop', Pop)$ ;

6:   if  $(\text{Improve}(Pop', Pop))$  then

7:      $Pop = Pop'$ ;

8:   end if

9: end while

```

Initialization strategy

The initialization strategy is used to evaluate the initial performance of each LLH. Beginning with generating a random population, each LLH h_i evolves the random population separately with the same number of generations, and obtains a resultant population Pop_i . The indicator value m_{pop_i} evaluated by indicator M for the resultant population Pop_i obtained

by h_i is then translated into an initial score s_i based on the normalized function, as shown in formula (1).

$$s_i = (m_{Pop_i} - m_{min}) / (m_{max} - m_{min}) \quad (1)$$

where m_{min} and m_{max} are, respectively, the minimum and maximum indicator values found so far by all LLHs in the initialization strategy. HH_EG can employ various indicators in the framework. If the indicator M is a maximization (or minimization) problem, the larger (or the smaller) score value equals to the better performance. Eventually, the resultant population from the best scoring LLH is then used as the initial population Pop of HH_EG.

Adaptive Epsilon-greedy selection strategy

An adaptive epsilon-greedy selection method is designed as a selection strategy to improve the decision-making ability of HH_EG. The main idea is that the adaptive epsilon-greedy selection strategy first focuses on exploring using the random algorithm to select an LLH. Then, the selection method begins to be greedier using the greedy algorithm. That is, we randomly generate a value in $[0, 1]$. If the random value is smaller than ϵ , a random LLH from the LLH set is selected to be applied in the next iteration; otherwise, the best scored LLH selected by the greedy algorithm will be applied.

In the traditional Epsilon-greedy algorithm, ϵ is a fixed parameter which is needed to be tuned manually. This paper uses a formula (2) to update the ϵ -value according to the number of iterations:

$$\epsilon = (iter_total - iter) / iter_total. \quad (2)$$

The ϵ -value presents a linear decreasing trend. Its value is large at former search processes, so HH_EG has a great probability to randomly select an LLH. At later stages, ϵ -value becomes smaller (down to 0), which means that HH_EG is to select the currently best-scored LLH. This strategy of selection makes the algorithm focusing on exploring different LLHs at early stages and then turns to greedy exploitation gradually as the search progresses.

Score update strategy

As discussed in [17, 38], the LLHs luckily to be chosen at the early stages are easy to get larger improvement values. However, the LLH used at later stages always only brings small improvements even if it is more effective than the rest of LLHs. So the raw indicator value is not able to reflect the actual performance of each LLH throughout the entire search process [38].

In the proposed algorithm, we use the indicator improvement rate with the indicator value of the output population as

the baseline to update the score vector S based on the feedback of the selected LLH h_{cur} during searching. If h_{cur} obtains a positive feedback after being applied at one iteration, then its score is rewarded. Otherwise, its score is punished.

Specifically, after applying h_{cur} at one iteration, the input population Pop and the output population Pop' are evaluated using the indicator M , and then the score value s_{cur} of h_{cur} is updated according to Eq. (3):

$$s_{cur} = s_{cur} + (m_{out} - m_{in}) / m_{out} \quad (3)$$

where m_{in} and m_{out} denote the indicator values of the input population and the output population. For a maximization indicator M , if the increment (we mean $m_{out} - m_{in}$) is greater than 0, we denote it that an improvement (positive feedback) has obtained after applying h_{cur} . The score s_{cur} is rewarded. Otherwise, it means a deteriorated operation (negative feedback) is executed by applying h_{cur} . The score s_{cur} is punished. For a minimization indicator M , the situation is reversed.

Move acceptance strategy

In this research, the only-improving acceptance strategy [22] is used to determine whether to accept the output population or not when solving MOPs. At each iteration, the indicator M is used to evaluate the input population Pop and the output population Pop' . We denote m_{in} and m_{out} as the indicator values of Pop and Pop' , respectively. The only-improving only accepts the improved new population. For a maximization (or minimization) indicator M , $m_{in} < m_{out}$ (or $m_{in} > m_{out}$) indicates that Pop' is improved compared to Pop , and Pop' will be the input population of the next iteration; otherwise, Pop is.

Experimental settings

To evaluate the effectiveness, we compare HH_EG with each LLH inside HH_EG and four multi-objective hyper-heuristic algorithms (HH_RAND, HH_CF [23], HH_RILA [19], and PAPHH [38]) on DTLZ (DTLZ1-7) [10], IMOP (IMOP1-8) [32], MaF (MaF1-7) [3] and VC (VC1-4) [20] test suites.

Test suites

DTLZ is a classical test suite which is often used as a baseline to test the performance of multi-objective algorithms. DTLZ1 contains linear and multimodal characteristics. The PFs of DTLZ2-6 are concave. Besides, DTLZ3 is a multimodal problem; DTLZ4 is biased; DTLZ5 is degenerate; and DTLZ6 is a biased and degenerate problem. DTLZ7 has mixed, discontinuous and biased characteristics. These

characteristics enable DTLZ to comprehensively test the performance of various algorithms. Experiments are conducted on three-objective DTLZ1–7 test problems.

IMOP is a novel test suite with extremely irregular PFs proposed recently. IMOP1–3 are two-objective MOPs. IMOP1 and IMOP2 have convex and concave PFs with sharp tails. The PF of IMOP3 is a multi-segment discontinuous curve. IMOP4–8 are three-objective MOPs with varied shaped PFs. For IMOP4, its PF is a wavy line. The PF of IMOP5 is composed of 8 independent circular faces. The PF of IMOP6 is divided into several grids. The PF of IMOP7 is part of 1/8 sphere, while the PF of IMOP8 looks like a continuous plane, but actually consists of 100 discontinuous small pieces.

MaF is a recently proposed test suite for many-objective optimization. We choose MaF 1–7 with 5 and 10 objectives in the experiments. The seven MaF problems are with diverse properties, such as being multimodal, disconnected, degenerate, and biased. The aim of using MaF is to verify the effectiveness of HH_EG on many-objective optimization problems with various real-world scenarios.

VC is a real-world multi-objective optimization problem referring to the ability of vehicles and components to protect passengers during a collision in the automotive industry. The VC test suite involves the optimization of weight (Mass), acceleration characteristics (Ain), and toe-board intrusion (Intrusion). Four VC problems are used in the experiments, including VC1: $\min F(x) = [\text{Mass}, \text{Ain}]$, VC2: $\min F(x) = [\text{Mass}, \text{Intrusion}]$, VC3: $\min F(x) = [\text{Ain}, \text{Intrusion}]$, and VC4: $\min F(x) = [\text{Mass}, \text{Ain}, \text{Intrusion}]$ with VC1–3 from [22] and VC4 from [20].

Performance indicators

In the experiments, IGD [4] and HV [42] are employed in the proposed HH_EG framework as performance indicators to evaluate each LLH during iterations of HH_EG. The main reasons for choosing IGD and HV are that they are commonly used and can evaluate multiple aspects, such as diversity and convergence, of the population. The IGD indicator evaluates the averaged minimum distance between each reference point on PF and the obtained population got by an algorithm. The HV indicator evaluates the objective space covered by a given population. For the two indicators, IGD requires prior knowledge of the PF, while HV does not. But HV is with a slower resolving speed, especially for high-dimensional MOPs. Since DTLZ, IMOP and MaF test suites have known PF, HH_EG incorporates IGD as a performance indicator to serve the high-level online learning strategy as the feedback when solving the two test suites. Aimed at the problems of VC test suite have unknown PFs, HH_EG uses HV instead of IGD as the performance indicator for solving VC test problems.

Low-level heuristics

Five MOEAs are used to form the low-level heuristic set, denoted as h_1 (NSGA-II [9]), h_2 (MOEA/D [37]), h_3 (IBEA [40]), h_4 (MSEA [34]), and h_5 (NSGA-III [8]). Generally, any population-based MOEAs can be incorporated into HH_EG as LLHs. The main reasons for choosing NSGA-II, MOEA/D, and IBEA are that they are classical and always be the baseline for comparison [31, 39]. MSEA is a recently proposed MOEA for getting better distributed population and NSGA-III is designed for many-objective optimization problems (having four or more objectives). Besides, the five MOEAs are from different categories of MOEAs. Specifically, NSGA-II and IBEA are the dominance-based MOEA and the indicator-based MOEA, respectively, while both MOEA/D and NSGA-III are the decomposition-based MOEA [21]. MSEA is the multi-stage MOEA [34]. Each of the five MOEAs is to promote the approximation population along with the PF, but they are different in utilizing different algorithm components during searching.

h_1 (NSGA-II): Each solution in the population is sorted according to its Pareto-optimal level. Only solutions in low levels are selected into the next generation. Besides, the crowding distance assignment strategy is used to maintain the diversity of the population during the selection process.

h_2 (MOEA/D): MOEA/D uses a set of uniformly distributed weight vectors to decompose the MOPs into multiple single-objective sub-problems. Each sub-problem is optimized by utilizing the information of neighborhood sub-problems, and the given weight vectors are used to control the diversity.

h_3 (IBEA): IBEA uses binary indicators to redefine the optimization goal. The indicator values are used as fitness in the environment selection, and no additional diversity maintaining strategy is needed to control diversity.

h_4 (MSEA): MSEA is a multi-stage MOEA for getting better-distributed population. The optimization process is divided into multiple stages according to the population in each generation, and the population is updated by different steady-state selection schemes in different stages. MSEA can solve MOPs with highly irregular PF.

h_5 (NSGA-III): NSGA-III is a reference-point-based evolutionary algorithm based on the NSGA-II framework with the aim to solve many-objective optimization problems. Solutions in the population are ranked by the non-dominated sorting method and the reference points are adopted to control the diversity.

Parameter settings

All LLHs use the simulated binary crossover operator and the polynomial distribution operator [7] for recombination and mutation respectively. The crossover probability is set to

Table 1 Parameter settings

Problem	m	N	$G_{\max}$	SEs	Algorithm	$iter_total/generation$	Other parameters
DTLZ1-7	3	91	1000	91,000	HH_EG	25/40	–
					PAPHH	25/40	$K = 20$
					HH_RILA	–	$g = 10$
					HH_CF	25/40	$\alpha = 100$
					HH_RAND	25/40	–
					Meta_algorithm	–	–
MaF1-7	5	126	500	63,000	HH_EG	25/20	–
					PAPHH	25/20	$K = 20$
					HH_RILA	–	$g = 10$
					HH_CF	25/20	$\alpha = 100$
					HH_RAND	25/20	–
					Meta_algorithm	–	–
VC1-3	2	91	1000	91,000	HH_EG	50/20	–
					PAPHH	50/20	$K = 20$
					HH_RILA	–	$g = 50$
					HH_CF	50/20	$\alpha = 100$
					HH_RAND	50/20	–
					Meta_algorithm	–	–

m is the number of objectives. N is the population size. $G_{\max}$ stands for the max generations. SEs is the solution evaluations. $iter_total$ and $generation$ are used in hyper-heuristics, standing for the number of total iterations and the number of generations of the selected LLH running in one iteration, respectively. Parameters K , g and α are the threshold used in PAPHH, the max number of generations in each iteration used in HH_RILA and the intensification parameter used in HH_CF, respectively

1 and the mutation probability is set to $1/n$ (n is the number of decision variables of a given problem). The crossover and mutation distribution index are both set to 20. MOEA/D uses PBI [37] as the decomposition method. The neighborhood size of MOEA/D is set to $0.1 \times N$ (N indicates the population size), and the maximum number of solutions replaced by each offspring is set to $0.01 \times N$. The uniformly distributed weight vectors are generated using Das and Dennis' sampling method [6]. The archive size of IBEA is the same as the population size N , and the fitness scaling factor is set to 0.05. Following the recommendation in [8, 19, 23], the parameter settings of each algorithm on DTLZ, IMOP, MaF, and VC test suites are given in Table 1. Other parameters are the same as used in their original papers.

The parameter k in DTLZ1-7 is set to $\{5, 10, 10, 10, 10, 10, 20\}$ [10]. The parameters of IMOP1–IMOP8 are set to $\{I = 5, L = 5, a_1 = 0.05, a_2 = 0.05, a_3 = 10\}$ [32], where I and L control the length of decision variables, while a_1 – a_3 control the difficulty of diversity protection.

The indicator IGD is used for DTLZ, IMOP, and MaF test suites. We sample 10,000 uniformly distributed solutions in the objective space from the PF of each DTLZ, IMOP and MaF test problems. The value of IGD ranges in $[0, \infty)$, the smaller value indicating a better performance. HV is used for the VC test suite. Before calculation, the population is

normalized. In the measurement of HV, the reference points are set as $r^i = z^{\text{nadir}_i} + 0.5(z^{\text{nadir}_i} - z^{\text{ideal}_i})$ for the i th objective [19], where $i \in 1 \dots m$ is the index of the objective and m is the number of objectives, z^{nadir_i} and z^{ideal_i} are the maximum and minimum values of the i th objective in the population to be calculated respectively. After normalization with the reference point, the value of HV ranges in $[0, 1]$. The larger HV value indicates a better performance.

All algorithms are run for 30 independent trials and compared using one-tailed Wilcoxon rank-sum test method. '+', '–', and '=' represent the comparing algorithm is “superior to”, “inferior to”, and “similar to” HH_EG at 0.05 significance level. All source codes are implemented with a popular public-domain software named PlatEMO [30] performed on an Intel core i7 3.1 GHz/8G/1T computer.

Experimental results

Experiments in this section are divided into four parts. In “[Experimental results on DTLZ](#)”, we evaluate the effectiveness of HH_EG by controlling three LLHs, $H = \{h_1, h_2, h_3\}$ on the DTLZ test suite. IGD is employed as the indicator since the PFs of DTLZs are known. In “[Experimental results on IMOP](#)” and “[Experimental results on MaF](#)”, we evalu-

ate the cross-domain ability of HH_EG by solving the two novel test suites, IMOP and MaF. With unchanged high-level framework, HH_EG only replaces LLHs with $H = \{h_1, h_2, h_4\}$ for low-dimensional IMOP test suite, $H = \{h_2, h_4, h_5\}$ for high-dimensional MaF test suite. IGD is used as an indicator. In “[Experimental results on VC](#)”, we evaluate HH_EG with the real-world multi-objective problem VC with $H = \{h_1, h_2, h_3\}$ as the LLH set. Since there is no known PFs in VC, HV is incorporated into HH_EG as the indicator.

The mean, standard deviation and ranking performance of HH_EG and the comparing algorithms with respect to IGD or HV indicators got from 30 independent trials on DTLZ, IMOP, MaF, and VC test suites are shown in Tables 2, 3, 4 and 5, respectively. The best mean IGD or HV value on each problem is highlighted in bold. In each table, the ranking values across all test problems are summarized for every algorithm (denoted as “Total”). After the summarization, the final ranking order is displayed sorted by the “Total” value (denoted as “Final Rank”). The corresponding box plots are illustrated in Figs. 1, 2, 3, 4 and 5, where the red lines in each box plot are the median value among 30 trials.

Experimental results on DTLZ

From Table 2, we can see from the statistical results of the IGD values on DTLZ test suite from 30 trials that HH_EG works the best on DTLZ2 and DTLZ4–7. HH_EG is significantly better than NSGA-II, IBEA, HH_RANDOM, HH_RILA, and PAPHH on all DTLZ problems, while HH_EG is significantly better than MOEA/D and HH_CF on five DTLZ problems, including DTLZ2 and DTLZ4–7. For the remaining two problems (DTLZ1, DTLZ3), HH_EG and HH_CF perform similarly. MOEA/D works the best on DTLZ1 and DTLZ3, followed by HH_EG. The superior performance of MOEA/D on DTLZ1 and DTLZ3 is attributed to its uniformly distributed weight vectors which make MOEA/D good at dealing with problems with regular PF. However, HH_EG works better than MOEA/D both on irregular-shaped problems (DTLZ5–7) and the rest regular-shaped problems (DTLZ2 and DTLZ4). Furthermore, HH_EG is relatively stable compared to comparing hyper-heuristics (HH_RANDOM, HH_CF, HH_RILA, and PAPHH), except DTLZ5 for HH_CF. Corresponding box plots are shown in Fig. 1. According to Total and Final Rank, HH_EG performs the best across all DTLZ test problems in the overall.

Experimental results on IMOP

Table 3 shows that HH_EG is superior to all comparing algorithms on all IMOP test problems with respect to the IGD values, except that HH_EG is inferior to MSEA on IMOP3. From the statistical results of the Wilcoxon rank-sum test, HH_EG is significantly better than NSGA-II,

MOEA/D, HH_RANDOM, and PAPHH on all IMOP test problems. HH_EG is significantly better than MSEA on five instances (IMOP1 and IMOP5–8), and performs similarly to MSEA on IMOP2 and IMOP4. With similar performance on IMOP4–5 for HH_CF and IMOP5 for HH_RILA, HH_EG performs significantly better than HH_CF and HH_RILA on the rest IMOP test problems. Among hyper-heuristics, HH_EG is more stable than HH_RANDOM and HH_RILA on all test problems. The stability of HH_EG is superior to that of HH_CF and PAPHH except for two test problems (IMOP2 and IMOP4 for HH_CF, IMOP6 and IMOP7 for PAPHH). Figure 2 shows corresponding box plots. The final rank in Table 3 assesses the generality level of each algorithm across IMOP1–8. The rank results show that HH_EG delivers the best performance.

Experimental results on MaF

Experimental results in Table 4 show that HH_EG is the top algorithm on 10 out of 14 MaF test problems except for 5-objective MaF4 and MaF6 test problems, 10-objective MaF4 and MaF7 test problems. On the four MaF test problems, MSEA performs the best. From the statistical results of the Wilcoxon rank-sum test, HH_EG is significantly better than MOEA/D, HH_RANDOM, and HH_CF on all MaF test problems. HH_EG is significantly better than NSGA-III, HH_RILA, and PAPHH on 13, 12, and 11 MaF test problems respectively, and performs similarly to NSGA-III, HH_RILA, and PAPHH on the rest test problems (10-objective MaF7 for NSGA-III, 10-objective MaF5 and 5-objective MaF7 for HH_RILA, 10-objective MaF1, MaF5, and MaF7 for PAPHH). HH_EG performs significantly better than MSEA on 10 MaF test problems, while MSEA performs significantly better than HH_EG on 3 MaF test problems. From Table 4, HH_EG is more stable among hyper-heuristics, except 5-objective MaF4 for PAPHH and 5-objective MaF5 for HH_RILA. HH_EG ranks first compared with all comparing algorithms according to the final rank, which confirms that HH_EG is effective when solving many-objective MaF test problems. Corresponding box plots of MaF test suite are shown in Figs. 3 and 4.

Experimental results on VC

Table 5 shows the statistical performance comparison of HH_EG and comparing algorithms from 30 independent runs with respect to the HV values for solving VC test problems. For the HV indicator, a higher value indicates better performance, and the box plots of each algorithm are visualized in Fig. 5. From Table 5, we can see that, among the three LLHs, IBEA and NSGA-II perform the best on VC1 and VC2–4, respectively. MOEA/D performs the worst. Besides, HH_EG is significantly better than NSGA-II, MOEA/D, and

Table 2 The statistical results (mean, standard deviation and ranking) of the IGD values obtained by comparing algorithms and HH_EG on DTLZ1 to DTLZ7

Problem	NSGA-II	MOEA/D	IBEA	HH_RAND	HH_CF	HH_RILA	PAPHH	HH_EG
DTLZ1	2.8004e-2 (1.28e-3) 5 –	2.0570e-2 (1.60e-5) 1 +	1.7638e-1 (2.82e-2) 8 –	5.7092e-2 (6.08e-2) 7 –	2.2711e-2 (3.70e-3) 4 =	3.4497e-2 (2.54e-2) 6 –	2.1171e-2 (1.57e-3) 3 –	2.0720e-2 (2.71e-4) 2
DTLZ2	7.2249e-2 (2.46e-3) 5 –	5.4482e-2 (1.66e-7) 2 –	8.4110e-2 (2.95e-3) 8 –	6.8025e-2 (1.25e-2) 4 –	7.2937e-2 (2.85e-3) 6 –	6.2018e-2 (1.26e-2) 3 –	7.5665e-2 (1.11e-2) 7 –	5.4464e-2 (4.26e-7) 1
DTLZ3	7.2574e-2 (3.32e-3) 5 –	5.5039e-2 (6.93e-4) 1 +	4.7580e-1 (1.17e-2) 8 –	2.7046e-1 (2.08e-1) 7 –	6.0746e-2 (8.59e-3) 4 =	1.8796e-1 (1.53e-1) 6 –	6.0155e-2 (7.41e-3) 3 –	5.9358e-2 (7.10e-3) 2
DTLZ4	7.0731e-2 (2.06e-3) 2 –	3.0878e-1 (2.93e-1) 8 –	8.2814e-2 (2.40e-3) 5 –	1.2707e-1 (2.23e-1) 6 –	7.1780e-2 (3.89e-3) 3 –	1.3239e-1 (2.21e-1) 7 –	7.7582e-2 (8.29e-3) 4 –	5.4510e-2 (1.83e-4) 1
DTLZ5	6.3639e-3 (3.89e-4) 2 –	3.3904e-2 (1.28e-5) 8 –	1.6774e-2 (1.55e-3) 6 –	2.1275e-2 (1.22e-2) 7 –	6.4238e-3 (4.72e-4) 3 –	8.9902e-3 (4.58e-3) 5 –	7.1347e-3 (4.82e-3) 4 –	6.1016e-3 (9.31e-4) 1
DTLZ6	6.4957e-3 (3.39e-4) 3 –	3.3921e-2 (5.92e-6) 8 –	2.6099e-2 (2.96e-3) 7 –	1.8224e-2 (1.11e-2) 5 –	2.2482e-2 (1.18e-2) 6 –	1.0890e-2 (8.42e-3) 4 –	6.4305e-3 (3.04e-4) 2 –	6.0537e-3 (2.09e-4) 1
DTLZ7	1.0767e-1 (8.20e-2) 4 –	1.7574e-1 (1.18e-1) 8 –	1.0791e-1 (9.14e-2) 5 –	1.2229e-1 (1.32e-1) 6 –	1.2613e-1 (1.02e-1) 7 –	8.6972e-2 (5.55e-2) 3 –	8.1314e-2 (1.82e-2) 2 –	7.1764e-2 (1.85e-3) 1
+/-/=	0/7/0	2/5/0	0/7/0	0/7/0	0/5/2	0/7/0	0/7/0	
Total	26	36	47	42	33	34	25	9
Final rank	3	6	8	7	5	4	2	1

The best results are highlighted in bold

Fig. 1 Box plots of each algorithm on DTLZ test suite from 30 trials with respect to the IGD values

IBEA on all VC test problems. Compared with the four hyper-heuristics, HH_EG achieves the best mean HV values than HH_RAND, HH_CF, HH_RILA, and PAPHH on VC1–4. In the final rank, HH_EG ranks first.

In general, across the DTLZ, IMOP, MaF, and VC test problems, HH_EG is superior to each LLHs on almost all test problems (except DTLZ1 and DTLZ3 for MOEA/D, IMOP3, 5-objective MaF4, 10-objective MaF4 and MaF7 for MSEA), indicating that HH_EG can combine the advan-

Table 3 The statistical results (mean, standard deviation and ranking) of the IGD values obtained by comparing algorithms and HH_EG on IMOP1 to IMOP8

Problem	NSGA-II	MOEA/D	MSEA	HH_RAND	HH_CF	HH_RILA	PAPHH	HH_EG
IMOP1	6.7431e-3 (2.84e-4) 2 –	3.5303e-1 (1.91e-3) 8 –	7.5132e-3 (1.14e-3) 3 –	1.2809e-1 (1.70e-1) 7 –	9.0327e-2 (1.54e-1) 6 –	5.5962e-2 (1.08e-1) 5 –	1.1191e-2 (4.51e-3) 4 –	6.0319e-3 (4.18e-4) 1
IMOP2	6.5317e-3 (2.80e-4) 4 –	7.8491e-1 (2.16e-4) 8 –	5.9139e-3 (7.51e-4) 2 =	3.3039e-2 (1.13e-1) 7 –	6.4355e-3 (3.18e-4) 3 –	1.8131e-2 (1.16e-2) 6 –	1.0928e-2 (2.97e-3) 5 –	5.9016e-3 (4.39e-4) 1
IMOP3	4.7808e-3 (1.59e-4) 4 –	1.5891e-1 (8.85e-2) 8 –	3.8808e-3 (1.64e-4) 1 +	2.0114e-2 (2.64e-2) 7 –	4.8572e-3 (2.08e-4) 5 –	6.3910e-3 (6.22e-3) 6 –	4.3420e-3 (1.09e-3) 3 –	3.9706e-3 (1.78e-4) 2
IMOP4	9.8784e-3 (4.59e-4) 4 –	2.1675e-2 (1.17e-2) 6 –	3.0984e-1 (3.52e-1) 8 =	1.6374e-2 (6.56e-3) 5 –	8.2521e-3 (1.06e-3) 2 =	3.3096e-2 (1.28e-1) 7 –	8.7831e-3 (1.41e-3) 3 –	8.0503e-3 (1.06e-3) 1
IMOP5	4.9382e-2 (3.01e-3) 4 –	5.5993e-1 (9.58e-2) 8 –	9.9954e-2 (1.02e-1) 7 –	5.0686e-2 (9.31e-3) 5 –	3.9556e-2 (6.08e-3) 2 =	3.9935e-2 (8.10e-3) 3 =	5.9427e-2 (5.86e-3) 6 –	3.6695e-2 (2.45e-3) 1
IMOP6	6.3011e-2 (9.10e-3) 7 –	1.3995e-1 (1.89e-1) 8 –	4.6006e-2 (7.13e-2) 6 –	4.5164e-2 (9.28e-3) 5 –	4.1134e-2 (1.31e-2) 4 –	3.4216e-2 (6.13e-3) 3 –	3.2925e-2 (2.26e-3) 2 –	3.1856e-2 (2.73e-3) 1
IMOP7	5.2057e-2 (4.23e-3) 6 –	9.3864e-1 (1.07e-4) 8 –	3.8811e-2 (1.34e-3) 3 –	7.6943e-2 (1.61e-1) 7 –	4.9523e-2 (6.06e-3) 5 –	4.8236e-2 (1.17e-2) 4 –	3.8545e-2 (1.98e-3) 2 –	3.8111e-2 (2.22e-3) 1
IMOP8	1.1719e-1 (5.18e-3) 6 –	1.0621e+0 (2.72e-3) 8 –	7.3167e-2 (1.07e-3) 2 –	1.2071e-1 (3.72e-2) 7 –	8.3846e-2 (1.97e-2) 5 –	7.6321e-2 (1.05e-2) 4 –	7.5970e-2 (8.01e-3) 3 –	7.2678e-2 (5.05e-3) 1
+/-/=	0/8/0	0/8/0	1/5/2	0/8/0	0/6/2	0/7/1	0/8/0	
Total	37	62	32	50	32	38	28	9
Final rank	5	8	3	7	3	6	2	1

The best results are highlighted in bold

Fig. 2 Box plots of each algorithm on IMOP test suite from 30 trials with respect to the IGD values

Table 4 The statistical results (mean, standard deviation and ranking) of the IGD values obtained by comparing algorithms and HH_EG on MaF1 to MaF7 with 5 and 10 objectives

Problem		NSGA-III	MOEA/D	MSEA	HH RAND	HH_CF	HH_RILA	PAPHH	HH_EG
MaF1	5	2.8320e−1 (3.21e−2) 8 −	2.0312e−1 (5.40e−2) 7 −	1.2600e−1 (8.70e−4) 2 −	1.9165e−1 (5.10e−2) 6 −	1.4065e−1 (4.40e−2) 4 −	1.7180e−1 (5.85e−2) 5 −	1.3078e−1 (7.38e−3) 3 −	1.2328e−1 (1.11e−3) 1
	10	2.8638e−1 (5.66e−3) 5 −	5.2061e−1 (2.76e−2) 8 −	2.3777e−1 (2.41e−3) 2 −	2.9574e−1 (6.73e−2) 7 −	2.8774e−1 (7.15e−3) 6 −	2.5095e−1 (5.18e−2) 3 −	2.5138e−1 (4.55e−2) 4 =	2.2114e−1 (3.11e−3) 1
MaF2	5	1.2867e−1 (3.72e−3) 7 −	1.2882e−1 (2.44e−4) 8 −	1.0857e−1 (1.81e−3) 2 −	1.2130e−1 (8.36e−3) 6 −	1.1329e−1 (4.69e−3) 4 −	1.1658e−1 (6.54e−3) 5 −	1.0950e−1 (2.90e−3) 3 −	1.0166e−1 (1.24e−3) 1
	10	2.3511e−1 (4.49e−2) 6 −	2.6175e−1 (5.45e−4) 8 −	2.4973e−1 (1.79e−2) 7 −	2.3111e−1 (4.26e−2) 5 −	2.1272e−1 (3.36e−2) 3 −	2.0348e−1 (1.77e−2) 2 −	2.1996e−1 (4.75e−2) 4 −	1.7401e−1 (6.83e−3) 1
MaF3	5	1.8532e−1 (4.20e−1) 5 −	1.1194e−1 (4.33e−3) 4 −	4.6720e+0 (8.53e+0) 8 −	1.0287e−1 (2.57e−2) 3 −	4.6354e+0 (1.56e+1) 7 −	1.1714e+0 (3.09e+0) 6 −	9.9779e−2 (2.51e−2) 2 −	8.5560e−2 (1.35e−2) 1
	10	3.3780e+4 (1.23e+5) 4 −	1.4170e−1 (1.26e−3) 2 −	1.0614e+11 (1.52e+11) 8 −	3.5960e+10 (1.29e+11) 7 −	2.3031e+6 (4.68e+6) 5 −	5.7114e+9 (3.02e+10) 6 −	1.1560e+0 (4.46e+0) 3 −	1.0597e−1 (1.34e−2) 1
MaF4	5	3.6130e+0 (2.27e+0) 4 −	1.0813e+1 (9.65e−1) 8 −	2.0750e+0 (6.94e−2) 1 +	6.1858e+0 (2.90e+0) 6 −	6.0055e+0 (4.20e+0) 5 −	6.4156e+0 (8.07e+0) 7 −	3.6124e+0 (1.73e+0) 3 −	2.8676e+0 (1.76e+0) 2
	10	9.9485e+1 (7.10e+0) 4 −	5.0483e+2 (2.71e+1) 8 −	5.1518e+1 (2.69e+0) 1 +	3.3793e+2 (2.12e+2) 7 −	1.5678e+2 (1.27e+2) 5 −	2.0193e+2 (2.18e+2) 6 −	9.8945e+1 (4.23e+1) 3 −	5.6849e+1 (6.13e+0) 2
MaF5	5	2.3972e+0 (3.61e−3) 4 −	8.3502e+0 (1.97e+0) 8 −	2.2327e+0 (3.93e−2) 2 −	4.3174e+0 (3.45e+0) 7 −	2.4196e+0 (1.30e−1) 6 −	2.3570e+0 (2.76e−2) 3 −	2.4181e+0 (3.19e−1) 5 −	2.1922e+0 (4.90e−2) 1
	10	7.8122e+1 (7.96e−1) 5 −	3.0161e+2 (1.58e+0) 8 −	8.8286e+1 (8.87e+0) 6 −	1.5475e+2 (1.17e+2) 7 −	7.0783e+1 (7.85e+0) 3 −	6.9355e+1 (4.48e+1) 2 =	7.0953e+1 (2.46e+1) 4 =	5.9380e+1 (4.09e+0) 1
MaF6	5	5.4528e−2 (8.31e−2) 6 −	2.0669e−1 (2.37e−1) 8 −	3.3991e−3 (4.13e−5) 1 =	1.4197e−2 (1.43e−2) 3 −	8.4306e−2 (2.09e−1) 7 −	1.5407e−2 (1.79e−2) 4 −	1.7600e−2 (2.63e−2) 5 −	3.4101e−3 (8.09e−5) 2
	10	3.6128e−1 (1.08e−1) 6 −	1.5362e−1 (2.16e−1) 5 −	5.8146e+0 (1.37e+1) 8 −	7.6935e−3 (9.05e−3) 3 −	1.3668e+0 (3.64e+0) 7 −	6.5604e−3 (7.95e−3) 2 −	1.4213e−2 (3.54e−2) 4 −	1.7013e−3 (2.77e−5) 1
MaF7	5	3.4414e−1 (3.08e−2) 2 −	6.4139e−1 (1.69e−1) 6 −	1.2407e+0 (6.13e−1) 8 −	8.2903e−1 (6.94e−1) 7 −	3.5125e−1 (3.59e−2) 3 −	3.9040e−1 (2.50e−1) 4 =	4.3857e−1 (2.90e−1) 5 −	3.2034e−1 (1.14e−1) 1
	10	1.0356e+0 (7.35e−2) 3 =	2.6300e+0 (7.03e−1) 8 −	9.2418e−1 (1.90e−2) 1 +	2.1721e+0 (2.20e+0) 7 −	1.9116e+0 (4.63e−1) 6 −	1.2034e+0 (2.83e−1) 5 −	1.1367e+0 (3.88e−1) 4 =	1.0175e+0 (1.27e−1) 2
+/-/=		0/13/1	0/14/0	3/10/1	0/14/0	0/14/0	0/12/2	0/11/3	
Total		69	96	57	81	71	60	52	18
Final rank		5	8	3	7	6	4	2	1

The best results are highlighted in bold

Table 5 The statistical results (mean, standard deviation and ranking) of the HV values obtained by comparing algorithms and HH_EG on VC1 to VC4

Problem	NSGA-II	MOEA/D	IBEA	HH RAND	HH_CF	HH_RILA	PAPHH	HH_EG
VC1	3.4438e-3 (4.64e-7) 4 –	7.3343e-4 (3.43e-8) 7 –	3.4451e-3 (1.62e-7) 2 –	7.1369e-4 (4.86e-5) 8 –	3.4437e-3 (4.98e-7) 5 –	3.4447e-3 (6.21e-7) 3 –	3.3577e-3 (4.96e-4) 6 –	3.4476e-3 (8.77e-6) 1
VC2	9.7955e-3 (3.07e-3) 2 –	3.3471e-8 (2.12e-8) 8 –	9.5286e-3 (2.98e-3) 3 –	1.9109e-3 (1.29e-3) 7 –	8.1872e-3 (3.91e-3) 6 –	8.4150e-3 (3.85e-3) 5 –	9.1282e-3 (3.68e-3) 4 =	1.0592e-2 (2.10e-3) 1
VC3	4.0719e-1 (4.41e-2) 3 –	6.9021e-2 (9.08e-3) 8 –	3.6426e-1 (7.00e-2) 6 –	2.1724e-1 (1.48e-1) 7 –	4.1061e-1 (4.10e-2) 2 –	3.9321e-1 (5.17e-2) 4 –	3.9264e-1 (5.33e-2) 5 –	4.1946e-1 (2.70e-2) 1
VC4	7.9906e-3 (2.57e-3) 5 –	2.3485e-4 (2.70e-6) 8 –	6.1816e-3 (2.02e-3) 6 –	1.1351e-3 (7.93e-4) 7 –	9.2205e-3 (2.36e-3) 2 –	8.2241e-3 (2.35e-3) 4 –	8.9191e-3 (2.41e-3) 3 =	9.2468e-3 (2.40e-3) 1
+/-/=	0/4/0	0/4/0	0/4/0	0/4/0	0/4/0	0/4/0	0/2/2	
Total	14	31	17	29	15	16	18	4
FinalRank	2	8	5	7	3	4	6	1

The best results are highlighted in bold

Fig. 3 Box plots of each algorithm on MaF test suite with 5 objectives from 30 trials with respect to the IGD values

tages of each LLH. Compared with HH_RAND, HH_CF, HH_RILA, and PAPHH, HH_EG performs the best on all DTLZ, IMOP, MaF, and VC test problems, and is more stable. The superiority of HH_EG shows the effectiveness of HH_EG in learning and selection.

Analysis of hyper-heuristics

The average utilization rate of LLHs

Figures 6, 7, 8, 9 and 10 show the average utilization rate of each LLH when HH_EG solves DTLZ, IMOP, MaF and VC

test problems (left). The utilization rate of an LLH means the number of invocations of the LLH divided by the number of total iterations of HH_EG in one run. Figures 6, 7, 8, 9 and 10 also show the average indicator values (IGD for DTLZ, IMOP and MaF, HV for VC) of three MOEAs, which are used as LLHs in HH_EG, running independently over 30 trials (right).

We conclude our results as follows. Firstly, HH_EG biases to the best performing LLH with respect to the indicator values. As shown in Figs. 6, 7, 8 and 9, the LLH with the lowest IGD value is always selected at the highest frequency by HH_EG on 22 out of 29 problems except for DTLZ4, IMOP5, IMOP7, 5-objective MaF7 and 10-objective MaF2, MaF5–6.

Fig. 4 Box plots of each algorithm on MaF test suite with 10 objectives from 30 trials with respect to the IGD values

Fig. 5 Box plots of each algorithm on VC test suite from 30 trials with respect to the HV values

Fig. 6 The average utilization rate of each LLH by HH_EG (left) and the average IGD values of each LLH run independently (right) over 30 trials on DTLZ test suite

Specifically, for the DTLZ and IMOP test suites, HH_EG chooses NSGA-II more frequently on DTLZ5–6, IMOP1, and IMOP4, while providing a bias towards MOEA/D for on DTLZ1–3. HH_EG prefers IBEA on DTLZ7, and MSEA on IMOP2–3, IMOP6, IMOP8. For the high-dimensional MaF test suite, HH_EG gives MOEA/D more chance on 5-objective and 10-objective MaF3. But HH_EG uses MSEA with a high probability on 5-objective MaF1–2, MaF4–6, and 10-objective MaF1, MaF4, MaF7. Figure 10 shows that the LLH with the highest HV (i.e. IBEA for VC1 and NSGA-II

for VC2–4) is selected the most by HH_EG when solving all VC test problems. The best performed LLH is always selected more frequently is mainly because that the IGD or HV is served as the feedback guidance in the learning strategy. Secondly, in certain cases, such as DTLZ4, IMOP5, IMOP7, 5-objective MaF7 and 10-objective MaF2, MaF5–6, HH_EG does not give the best performed LLH a higher utilization rate. For DTLZ4, MOEA/D is given more chance instead of NSGA-II. MSEA is selected more than the best performed LLH (i.e. NSGA-II) on IMOP5, while NSGA-II

Fig. 7 The average utilization rate of each LLH by HH_EG (left) and the average IGD values of each LLH run independently (right) over 30 trials on IMOP test suite

Fig. 8 The average utilization rate of each LLH by HH_EG (left) and the average IGD values of each LLH run independently (right) over 30 trials on MaF test suite with 5 objectives

Fig. 9 The average utilization rate of each LLH by HH_EG (left) and the average IGD values of each LLH run independently (right) over 30 trials on MaF test suite with 10 objectives

Fig. 10 The average utilization rate of each LLH by HH_EG (left) and the average HV values of each LLH run independently (right) over 30 trials on VC test suite

is selected more than the best LLH (i.e. MSEA) on IMOP7. HH_EG prefers MSEA on 5-objective MaF7, 10-objective MaF2, MaF5, and MOEA/D on 10-objective MaF6. This might indicate that combining LLHs together could exploit the potential of ‘poor’ LLH and achieve a better result finally.

The number of invocations of each LLH versus the average indicator values

To further understand the search progress induced by each LLH, Fig. 11 provides plots based on the number of invocations of each LLH at each iteration obtained from HH_EG

over 30 trials on DTLZ2 and IMOP5 (left). Each point on the left subplot indicates the total number of trials of the corresponding LLH been selected by HH_EG, with its value ranges in [0, 30]. Figure 11 also plots the average IGD values versus iterations from HH_EG and three MOEAs running alone over 30 trials (right).

On DTLZ2, in the initial stages, the left subplot shows that the number of invocations of NSGA-II, MOEA/D, and IBEA are similar, which is caused by the randomness of the random algorithm in the adaptive epsilon-greedy selection strategy in the beginning exploration phase. Later, HH_EG slowly reduces the usage of NSGA-II and IBEA, while increas-

Fig. 11 The number of invocations of each LLH (left) and the average IGD values of four algorithms (right) versus iterations over 30 trials on DTLZ2 and IMOP5

ing the usage of MOEA/D. MOEA/D is even predominately used during the final stages. According to the right subplot, MOEA/D is the best performed (with lowest IGD values) algorithm compared to NSGA-II and IBEA. It is not difficult to understand that as the search progresses, the adaptive epsilon-greedy selection strategy becomes more exploitative and prefers invoking the best performed LLH.

On IMOP5, the invocations of NSGA-II, MOEA/D, and MSEA are not dominated with each other, especially in the first ten iterations. After that, the invocations of MSEA is gradually increased with significant superiority to NSGA-II and MOEA/D. But, from the right subplot, it is surprising to find that MSEA is only the second-best LLH when running alone, following the best performing LLH (NSGA-II). Meanwhile, HH_EG obtains the optimal IGD value on IMOP5 (as shown in Table 3 and right subplot of Fig. 11), which also shows that after combining each LLH, HH_EG produces better results than each LLH running alone.

The scores of each LLH versus the IGD values

This section investigates the performance of each LLH in HH_EG from the best trail from 30 trials, as shown in Figs. 12 and 13. The left subplots show the score values of each LLH during the solving of HH_EG. The lower score value indicates a better performance. Correspondingly, the right subplots show the IGD values of HH_EG where each tick in the right plots indicate the IGD value obtained after HH_EG applies the selected LLH during searching.

Fig. 12 The score values of each LLH (left) and IGD values of HH_EG (right) versus iterations on DTLZ2 from the trail with the lowest final IGD value among 30 trials

Fig. 13 The score values of each LLH (left) and IGD values of HH_EG (right) versus iterations on IMOP5 from the trail with the lowest final IGD value among 30 trials

Each problem has its own pattern. For example, for DTLZ2 (see Fig. 12), MOEA/D is invoked and applied at the 2nd, 3rd, and 8th iterations, where the IGD values decrease and the corresponding score values of MOEA/D decrease, too. That is the scores of MOEA/D are rewarded. At the 5th, 7th, and 9th iterations, NSGA-II is invoked, while at the 1st, 4th, and 6th iterations, IBEA is invoked. The IGD values at the six iterations are unchanged, but their corresponding score values increase, indicating that un-improved populations are generated. The scores of NSGA-II and IBEA are punished. As we can see, MOEA/D has the lowest scores (performing the best). As the iteration progresses, the adaptive epsilon-greedy selection strategy becomes greedier and MOEA/D has more chance to be selected, which are consistent with the high utilization rate of MOEA/D on DTLZ2 in previous reports in “The average utilization rate of LLHs”.

For IMOP5 (see Fig. 13), NSGA-II is selected at the 1st and 2nd iterations, while MSEA is selected at the 3rd, 7th, 8th and 24th iterations. The IGD values decrease at the six iterations and the corresponding score values decrease. For the rest iterations, the IGD values do not change, but the corresponding score values increase, indicating that the new generated population is not improved after applying LLHs at these iterations. From Fig. 13, the score values of NSGA-II decrease below MSEA in the beginning and increase above MSEA later. According to the score values, HH_EG considers that NSGA-II is superior at the early stages, while

MSEA works better at later stages. For the adaptive epsilon-greedy selection strategy bias to select the best performed LLH as the search progresses, the probability of selecting MSEA is generally increasing. This explains why MSEA as the second-best LLH is selected the most on IMOP5 (shown in Figs. 7 and 11).

Conclusion

This paper proposes a novel multi-objective hyper-heuristic, HH_EG. This algorithm combines the advantages of existing MOEAs to produce better results on multiple types of problems. Four strategies make up the main high-level framework of HH_EG, including initialization, adaptive epsilon-greedy selection, score update, and move acceptance. Performance indicators are utilized to provide feedback to the high-level online learning strategy. In the proposed algorithm, no parameter is needed to be tuned. This suggests that HH_EG requires little expert intervention. This simplification makes it feasible to incorporate variety of indicators into HH_EG. Experimental results show that HH_EG is more competitive than each LLH running alone and is also superior to four classical hyper-heuristics when dealing with MOPs. Results on DTLZ, IMOP, MaF, and VC show the competitiveness and cross-domain search abilities of the proposed hyper-heuristic.

In future work, we will apply HH_EG to different types of MOPs, such as many-objective, multi-model, or real-world problems, to assess its effectiveness as well as the cross-domain ability. Another research direction is the impact of hyper-heuristics with different algorithm components as LLHs. Besides the IGD and HV indicators in our work, other indicators could be combined with hyper-heuristics. The impact of indicator values as the feedback in learning strategies is also left as future work.

Acknowledgements The authors would like to thank Professor Ye Tian for insightful comments and suggestions.

Funding The Technological Research Projects in Henan Province (grant number 192102210107) and the National Key R&D Project (grant number 2020YFB1712401).

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Availability of data and material All data generated or analyzed during this study are included in this published article.

Code availability The custom code of the current study is available from the corresponding author.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Burke EK, Hyde MR, Kendall G, Ochoa G, Özcan E, Woodward JR (2019) A classification of hyper-heuristic approaches: revisited. In: Gendreau M, Potvin J-Y (eds) *Handbook of metaheuristics*. Springer International Publishing, Cham, pp 453–477. https://doi.org/10.1007/978-3-319-91086-4_14
- Burke EK, Silva J, Soubeiga E (2005) Multi-objective hyper-heuristic approaches for space allocation and timetabling. In: Ibaraki T, Nonobe K, Yagiura M (eds) *Metaheuristics: progress as real problem solvers*, vol 32. Springer US, Boston, pp 129–158. https://doi.org/10.1007/0-387-25383-1_6
- Cheng R, Li MQ, Tian Y, Zhang XY, Yang SX, Jin YC, Yao X (2017) A benchmark test suite for evolutionary many-objective optimization. *Complex Intell Syst* 3:67–81. <https://doi.org/10.1007/s40747-017-0039-7>
- Coello CAC, Cortés NC (2005) Solving multiobjective optimization problems using an artificial immune system. *Genet Program Evolvable Mach* 6:163–190. <https://doi.org/10.1007/s10710-005-6164-x>
- Coello CAC, González Brambila S, Figueroa Gamboa J, Castillo Tapia MG, Hernández Gómez R (2020) Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex Intell Syst*. <https://doi.org/10.1007/s40747-019-0113-4>
- Das I, Dennis JE (1998) Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM J Optim* 8:631–657. <https://doi.org/10.1137/S1052623496307510>
- Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. *Complex Syst* 9:115–148
- Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans Evol Comput* 18:577–601. <https://doi.org/10.1109/TEVC.2013.2281535>
- Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6:182–197. <https://doi.org/10.1109/4235.996017>
- Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multiobjective optimization. In: Abraham A, Jain L, Goldberg R (eds) *Evolutionary multiobjective optimization: theoretical advances and applications*. Springer, London, pp 105–145. https://doi.org/10.1007/1-84628-137-7_6
- Glover F (1986) Future paths for integer programming and links to artificial intelligence. *Comput Oper Res* 13:533–549. [https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1)
- Gonçalves RA, de Almeida CP, Venske SMGS, da Silva MRdB, Pozo ATR (2017) A new hyper-heuristic based on a restless multi-armed bandit for multi-objective optimization. In: 2017

- Brazilian conference on intelligent systems (BRACIS), Uberlandia, Brazil, 2017. IEEE, New York, pp 390–395. <https://doi.org/10.1109/BRACIS.2017.67>
13. Hua YC, Jin YC, Hao KR, Cao Y (2020) Generating multiple reference vectors for a class of many-objective optimization problems with degenerate Pareto fronts. *Complex Intell Syst*. <https://doi.org/10.1007/s40747-020-00136-5>
 14. Ishibuchi H, Setoguchi Y, Masuda H, Nojima Y (2017) Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Trans Evol Comput* 21:169–190. <https://doi.org/10.1109/TEVC.2016.2587749>
 15. Kieffer E, Danoy G, Brust MR, Bouvry P, Nagih A (2019) Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic. *IEEE Trans Evol Comput* 24:44–56. <https://doi.org/10.1109/TEVC.2019.2906581>
 16. Kuang NL, Leung CHC (2019) Performance effectiveness of multimedia information search using the epsilon-greedy algorithm. In: 2019 18th IEEE international conference on machine learning and applications (ICMLA), Boca Raton, FL, USA, 16–19 Dec 2019. pp 929–936. <https://doi.org/10.1109/ICMLA.2019.00160>
 17. Li K, Fialho A, Kwong S, Zhang QF (2013) Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput* 18:114–130. <https://doi.org/10.1109/TEVC.2013.2239648>
 18. Li WW, Özcan E, John R (2017) Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation. *Renew Energy* 105:473–482. <https://doi.org/10.1016/j.renene.2016.12.022>
 19. Li WW, Özcan E, John R (2019) A learning automata-based multiobjective hyper-heuristic. *IEEE Trans Evol Comput* 23:59–73. <https://doi.org/10.1109/TEVC.2017.2785346>
 20. Liao XT, Li Q, Yang XJ, Zhang WG, Li W (2007) Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Struct Multidiscip Optim* 35:561–569. <https://doi.org/10.1007/s00158-007-0163-x>
 21. Lin QZ, Liu SB, Zhu QL, Tang CY, Song RZ, Chen JY, Coello CAC, Wong KC, Zhang J (2018) Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems. *IEEE Trans Evol Comput* 22:32–46. <https://doi.org/10.1109/TEVC.2016.2631279>
 22. Maashi M, Kendall G, Özcan E (2015) Choice function based hyper-heuristics for multi-objective optimization. *Appl Soft Comput* 28:312–326. <https://doi.org/10.1016/j.asoc.2014.12.012>
 23. Maashi M, Özcan E, Kendall G (2014) A multi-objective hyper-heuristic based on choice function. *Expert Syst Appl* 41:4475–4493. <https://doi.org/10.1016/j.eswa.2013.12.050>
 24. McClymont K, Keedwell E (2011) Markov chain hyper-heuristic (MCHH): An online selective hyper-heuristic for multi-objective continuous problems. In: Krasnogor N (ed) 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, July 2011. <https://doi.org/10.1145/2001576.2001845>
 25. McClymont K, Keedwell E, Savić D, Randall Smith M (2013) A general multi-objective hyper-heuristic for water distribution network design with discolouration risk. *J Hydroinform* 15:700–716. <https://doi.org/10.2166/hydro.2012.022>
 26. Ortiz-Bayliss JC, Terashima-Marín H, Conant-Pablos SE (2013) Using learning classifier systems to design selective hyper-heuristics for constraint satisfaction problems. In: 2013 IEEE Congress on evolutionary computation, Cancun, Mexico, 20–23 June 2013, pp 2618–2625. <https://doi.org/10.1109/CEC.2013.6557885>
 27. Pillay N, Qu R (2018) Selection constructive hyper-heuristics. In: Bäck T, Kari L (eds) *Hyperheuristics: Theory and applications*. Springer International Publishing, Cham, pp 7–16. https://doi.org/10.1007/978-3-319-96514-7_2
 28. Sutton RS, Barto AG (2018) *Reinforcement learning: an introduction*. MIT Press, Cambridge
 29. Tan KC, Lee TH, Khor EF (2002) Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons. *Artif Intell Rev* 17:251–290. <https://doi.org/10.1023/A:1015516501242>
 30. Tian Y, Cheng R, Zhang XY, Jin YC (2017) PlatEMO: a MATLAB platform for evolutionary multi-objective optimization. *IEEE Comput Intell Mag* 12:73–87. <https://doi.org/10.1109/MCI.2017.2742868>
 31. Tian Y, Cheng R, Zhang XY, Jin YC (2018) An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. *IEEE Trans Evol Comput* 22:609–622. <https://doi.org/10.1109/10.1109/TEVC.2017.2749619>
 32. Tian Y, Cheng R, Zhang XY, Li MQ, Jin YC (2019) Diversity assessment of multi-objective evolutionary algorithms: performance metric and benchmark problems. *IEEE Comput Intell Mag* 14:61–74. <https://doi.org/10.1109/MCI.2019.2919398>
 33. Tian Y, Cheng R, Zhang XY, Su YS, Jin YC (2018) A strengthened dominance relation considering convergence and diversity for evolutionary many-objective optimization. *IEEE Trans Evol Comput* 23:331–345. <https://doi.org/10.1109/TEVC.2018.2866854>
 34. Tian Y, He C, Cheng R, Zhang XY (2019) A multistage evolutionary algorithm for better diversity preservation in multiobjective optimization. *IEEE Trans Syst Man Cybern Syst*. <https://doi.org/10.1109/TSMC.2019.2956288>
 35. Tian Y, Peng SC, Rodemann T, Zhang XY, Jin YC (2019) Automated selection of evolutionary multi-objective optimization algorithms. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, Xiamen, China, pp 3225–3232. <https://doi.org/10.1109/SSCI44817.2019.9003018>
 36. Yao Y, Peng Z, Xiao B (2018) Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city. *IEEE Trans Veh Technol* 67:10307–10318. <https://doi.org/10.1109/TVT.2018.2868942>
 37. Zhang QF, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput* 11:712–731. <https://doi.org/10.1109/TEVC.2007.892759>
 38. Zhang SY, Ren ZL, Li CX, Xuan JF (2020) A perturbation adaptive pursuit strategy based hyper-heuristic for multi-objective optimization problems. *Swarm Evol Comput* 54:100647. <https://doi.org/10.1016/j.swevo.2020.100647>
 39. Zhang XY, Zheng XT, Cheng R, Qiu JF, Jin YC (2018) A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. *Inf Sci* 427:63–76. <https://doi.org/10.1016/j.ins.2017.10.037>
 40. Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: Yao X, Burke EK, Lozano JA (eds) *Parallel Problem Solving from Nature - PPSN VIII*. Springer, Berlin, Heidelberg, pp 832–842. https://doi.org/10.1007/978-3-540-30217-9_84
 41. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm. TIK-report 103. <https://doi.org/10.3929/ethz-a-004284029>
 42. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans Evol Comput* 3:257–271. <https://doi.org/10.1109/4235.797969>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.