IEEE Computational Intelligence Society — Transactions on Evolutionary Computation

A Learning Automata based Multiobjective Hyper-heuristic

SCHOLARONE™
Manuscripts

A Learning Automata based Multiobjective Hyper-heuristic

Wenwen Li, Ender Özcan, and Robert John

Abstract—Metaheuristics, being tailored to each particular domain by experts, have been successfully applied to many computationally hard optimisation problems. However, once implemented, their application to a new problem domain or a slight change in the problem description would often require additional expert intervention. There is a growing number of studies on reusable cross-domain search methodologies, such as, selection hyper-heuristics, which are applicable to problem instances from various domains, requiring minimal expert intervention or even none. This study introduces a new learning automata based selection hyper-heuristic controlling a set of multiobjective metaheuristics. The approach operates above three well-known multiobjective evolutionary algorithms and mixes them, exploiting the strengths of each algorithm. The performance and behaviour of two variants of the proposed selection hyper-heuristic, each utilising a different initialisation scheme are investigated across a range of unconstrained multiobjective mathematical benchmark functions from two different sets and the real-world problem of vehicle crashworthiness. The empirical results illustrate the effectiveness of our approach for cross-domain search, regardless of the initialisation scheme, on those problems when compared to each individual multiobjective algorithm. Moreover, both variants perform signicantly better than some previously proposed selection hyper-heuristics for multiobjective optimisation, thus signicantly enhancing the opportunities for improved multiobjective optimisation.

Index Terms—Online learning, Multiobjective optimisation, Hyper-heuristics, Evolutionary algorithms, Operational research

I. INTRODUCTION

Multiobjective optimisation problems (MOPs) require simultaneous handling of various and often conflicting objectives during the search process. The solution methods designed for MOPs seek a set of 'equivalent' solutions, each reflecting a *trade-off* between different objectives.

There are distinct complexities associated with MOPs making the development of effective and efficient solution methods extremely challenging (e.g., very large search spaces, noise, uncertainty, etc.). Metaheuristics, in particular, multiobjective evolutionary algorithms (MOEAs) are the most commonly used search methods in the area of solving MOPs. One of the main advantages of MOEAs is that they are population based techniques, capable of obtaining a set of trade-off solutions with reasonable quality even in a single run [1]. Even though 'optimality' can not be guaranteed, empirical results indicate the success of MOEAs on a variety of problem domains, including planning and scheduling ([2], [3]), data mining [4],

W. Li, E. Özcan and R. John are with the ASAP research group, School of Computer Science, University of Nottingham, UK
E-mail: {psxwl8,pszeo,pszrij}@nottingham.ac.uk

and circuits and communications [5]. There are different types of MOEAs, each utilising different algorithmic components during the search process and so perform differently. In the majority of the previous studies, individual MOEAs are designed and applied to a particular problem in hand. More on MOEAs and their applications to various multiobjective problems can be found in [6].

On the other hand, there is a growing number of studies on *selection hyper-heuristics* which provide a general-purpose heuristic optimisation framework for utilising the strengths of multiple (meta)heuristics [7]. Selection hyper-heuristics control and mix low level (meta)heuristics, automatically deciding which one(s) to apply to the candidate solution(s) at each decision point of the iterative search process [8]. Raising the generality level of heuristic optimisation methods is one of the main motivations behind the hyper-heuristic studies. The idea is, through automation of the heuristic search, to provide effective and reusable *cross-domain* search methodologies which are applicable to the problems with different characteristics from various domains without requiring much expert involvement.

Learning is key to develop an effective selection hyper-heuristic with the adaptation capability. There are some recent studies looking into the interplay between data science techniques, particularly machine learning algorithms and selection hyper-heuristics leading to an improved overall performance. For example, [9] and [10] used tensor analysis as a machine learning approach to decide which low level heuristics to employ at different stages of the search process. In [11], the feasibility and effectiveness of using reinforcement learning to improve the performance of metaheuristics and hyper-heuristics have been discussed in depth. [12] introduced an effective multi-stage hyper-heuristic for cross-domain search which first, reduces the low level heuristics to be used in the following stage based on a multiobjective learning strategy and then mixes them under a stochastic local search framework. More recently, computational intelligence techniques have been used as components of general purpose methods managing low level (meta)heuristics for overall performance improvement. For example, [13] introduced a fuzzy inference selection based hyper-heuristic which mixed and controlled four search operators, each derived from a different meta-heuristic to solve a computationally hard problem of t-way test suite generation. However, the aforementioned studies all focus on single objective optimisation. There have been some studies on combining the strengths of multiple MOEAs with the aim of providing a better overall performance for multiobjective optimisation under a selection hyper-heuristic

framework (e.g., [14], [15]). From this point onward, we will refer to such selection hyper-heuristics as *multiobjective hyper-heuristics* (MOHHs).

In this study, we present a new learning automata based selection hyper-heuristic framework with implementation of two variants, Learning Automata based Hyper-heuristic (HH-LA) and Learning Automata based Hyper-heuristic with a Ranking Scheme Initialisation (HH-RILA) for multiobjective optimisation. Both selection hyper-heuristics mix and control a set of three well-known MOEAs: nondominated sorting genetic algorithm (NSGA-II) [16], strength Pareto evolutionary algorithm 2 (SPEA2) [17] and indicator based evolutionary algorithm (IBEA) [18]. The learning automaton acts as a guidance for choosing the appropriate MOEA at each decision point while solving a given problem. The proposed two variants of selection hyper-heuristics mainly differ in their initial set-up process. HH-LA employs all three low level MOEAs and gives an equal chance initially to each algorithm making a random start. HH-RILA applies a ranking scheme which eliminates the relatively poor performing MOEA(s) and uses the remaining MOEAs in the improvement process (Section III-A). The performance of the proposed hyper-heuristics are investigated against a variety of other multiobjective approaches across a range of multiobjective problems, including well-known benchmark functions and a real-world problem of vehicle crashworthiness. The empirical results indicate the effectiveness and generality of the proposed hyper-heuristics with novel components.

The rest of the paper is organised as follows. Section II introduces some essential concepts of MOPs, selection multiobjective hyper-heuristics as well as learning automata and provides background for vehicle crashworthiness. Section III presents the details of the proposed method which embeds three novel components. Firstly, the learning automaton component designed for multiobjective optimisation operates in a non-traditional way as explained in Section III-B. The second component, as described in Section III-C supports the development of a two-stage metaheuristic selection approach based on the information obtained from the learning process, enabling the use of two different metaheuristic selection methods at different stages. The third component as described in Section III-D adaptively decides when to switch to another MOEA depending on a tuned improvement threshold parameter. The parameter tuning and setting are included in Section IV, as well as the discussion and analysis of the experimental results. Section V concludes this study and provides directions for future work.

II. BACKGROUND

A. Related Work on Multiobjective Selection Hyper-heuristics

MOEAs and other multiobjective approaches aim to identify true Pareto fronts (PFs), i.e., equal quality optimal trade-off solutions. If the true PFs are unknown, then MOEAs are used to generate 'good' approximations [19]. The majority of the multiobjective approaches contain certain algorithmic components to achieve the following key goals [1]: i) preserve nondominated solutions; ii) progress towards the true PFs; iii) maintain a diverse set of solutions in the objective space.

WFG [20] and DTLZ [21] are two widely used test suites in the MOEA literature that provide benchmark functions with various characteristics. The comparison of different PFs obtained from different MOEAs is not trivial because multiple aspects should be considered, such as convergence (how close the final fronts to the true PFs are) and diversification (how dispersed the obtained fronts are) capabilities. There are a variety of performance indicators including the convergence indicators, such as *hypervolume*, $\epsilon+$. [19]. Hypervolume measures the size of objective space covered by the resultant front with respect to a reference point, while $\epsilon+$ is the minimum distance that a solution front needs to move in all dimensions to dominate the reference front. As for diversification, the most commonly used indicators include *spread* [16] and *generalised spread* [22] which extends *spread* to higher than two dimensions. Generalised spread is computed based on the mean Euclidean distance of any nearest pairs of neighbours in the nondominated solution set. The smaller the value, the better the spread of the resultant front. More analysis and review of various performance indicators for MOEAs can be found in [19].

Designing, implementing and maintaining a (meta)heuristic for a particular problem is a time-consuming process requiring a certain level of expertise in both the problem domain and heuristic optimisation. Once implemented, application of a metaheuristic to a new problem domain or even a slight change in the problem description would often require the intervention of an expert. This is basically due to the fact that metaheuristics are often customised for a particular problem domain (benchmark). On the other hand, *hyper-heuristics* have emerged as automated general-purpose cross-domain optimisation methods with reusable components which can be applied to multiple problem domains/benchmarks with the least modification [7]. Dealing with multiple problem domains and problem instances means dealing with various scales of objective values, making it extremely difficult to compare the cross-domain performances of algorithms. Which method to use for performance comparison of hyper-heuristics across multiple problem domains (distributions/benchmarks) and how the performance comparison should be done are still open issues in the hyper-heuristic research. Currently, there are two commonly used metrics in the area: *Formula* 1 *ranking* ([9], [12]) and μ_{norm} ([23], [24]). In this work, we preferred the latter one (details are in Section IV-B) which is a more informative metric taking into account of the mean performance of algorithms using normalised performance indicator values over a given number of trials on the instances from multiple problem domains/benchmarks.

The focus of this study is on selection hyper-heuristics which choose and apply from a set of low level (meta)heuristics at each decision point of the search [8]. A key component in a selection hyper-heuristic is the (meta)heuristic selection method which should be capable of adapting itself depending on the situation to choose the appropriate low level (meta)heuristic at each decision point. Hence, learning is a crucial component of (meta)heuristic selection methods. Additionally, move acceptance technique is another key component of selection hyper-heuristics ([25], [26]), which

determines whether or not newly generated solution(s) should be accepted as the input solution(s) to the next step/stage. The majority of the previous studies on selection hyper-heuristics focus on optimisation of single objective problems. Still, there are a few studies on multiobjective selection hyper-heuristics investigating either the use of selection hyper-heuristics controlling multiple operators or mixing multiple multiobjective metaheuristics.

[27] presented a selection hyper-heuristic (HH-AP) using an online learning heuristic selection method based on adaptive pursuit [28] managing five domain-specific perturbation operators. HH-AP is utilised for solving a multiobjective design problem for an Earth observation satellite system. [29] proposed a hyper-heuristic which mixes four different indicators, each from a well-established MOEA, including NSGA-II, SPEA2 and two IBEA variants to rank individuals for mating. An indicator gets selected depending on the associated probability for each individual and four subpopulations are constructed. Mating occurs within each subpopulation using binary tournament selection and eventually, four offspring pools are formed constituting to the new population. The indicator probabilities are maintained during the search via mixture experiments based on a statistical model. [30] and [31] incorporated a roulette wheel based heuristic selection mechanism [32] into their multiobjective hyper-heuristic evolutionary algorithm to select low level mutation operators. [33] developed a hyper-heuristic based on two heuristic selection methods (choice function [14] and multi-armed bandit [34]) for choosing from multiple mutation and crossover operators during the search for the multiobjective integration and test order problems [35].

Some offline learning techniques have also been seen in recent MOHHs studies, e.g., genetic programming techniques in ([36], [37], [38], [39], [40]), grammatical evolution [41] in [42] and top-down induction of decision trees in [43].

On the other hand, there are a few studies on multiobjective search methods that make use of multiple MOEAs. [44] proposed a multialgorithm genetically adaptive multiobjective (AMALGAM) method performing *cooperative search* using various MOEAs. AMALGAM executes all MOEAs simultaneously, each with a separate subpopulation at each step, and a pool of offspring gets generated by each MOEA. Those offspring pools from MOEAs are merged to form the new population. Afterwards, fast nondominated sorting is applied to the union of the new and previous populations to choose the elite solutions surviving to the next generation. The size of the subpopulation for each MOEA gets updated adaptively based on the number of surviving solutions from each MOEA. The search continues until a set of termination criteria is satisfied.

[14] introduced a powerful online learning selection hyper-heuristic for multiobjective optimisation, namely choice function based MOHH (HH-CF), managing NSGA-II, SPEA2 and MOGA [45]. The proposed choice function maintains an adaptively changing score for each low level MOEA during the search process based on two key components: individual performance and time elapsed since the last call of an MOEA. The former component uses four different indicators, including hypervolume, uniform distribution, ratio of nondominated

individuals and algorithm effort [46]. It is for exploitation, advocating the invocation of the most successful MOEA with the highest score repeatedly, while the other component is for exploration, giving a chance to the MOEAs which were used the least. The MOEA with the top score is always chosen and applied at each decision point. The results in [14] show that HH-CF outperforms not only the three underlying MOEAs which are executed individually, but also AMALGAM and a random hyper-heuristic on the majority cases of bi-objective WFG benchmark functions.

In this study, we focus on online learning techniques as a part of selection MOHHs. It has already been observed that different MOEAs show strengths with respect to different metrics on different multiobjective optimisation problem domains [47]. The learning ability for detecting the best performing (meta)heuristic and/or identifying the *synergetic* (meta)heuristics ([10], [48]) over time is crucial to design an effective selection hyper-heuristic. Hence, it is reasonable to incorporate different MOEAs within an online learning selection hyper-heuristic framework for improving the cross-domain performance of the overall approach which can benefit from adaptively switching between those MOEAs over time.

HH-CF [14] is one of the best performing online learning multiobjective hyper-heuristics, to the extent of our knowledge. Similar to HH-CF, the proposed hyper-heuristics can also perform exploration and exploitation. A major difference is that the online learning method is based on *learning automata* within our selection hyper-heuristics for multiobjective optimisation. Additionally, there is an adaptive mechanism to ensure that the balance between the exploration and exploitation is maintained based on the information gathered by this machine learning technique during the search. A variant of learning automata was embedded into a single-objective hyper-heuristic i.e., AdapHH [48] which won the CHeSC competition[1] across six problem domains: Max-SAT, Bin Packing, Personnel Scheduling, Flow Shop, Travelling Salesman Problem and Vehicle Routing Problem. The importance of learning in selection hyper-heuristics and the success of AdapHH in solving single objective optimisation problems motivated us to employ an online learning mechanism within our multiobjective hyper-heuristics for cross-domain search.

B. Learning Automata

Learning automata, introduced by Testlin [49] as a reinforcement learning method, has been used in a range of fields, including pattern classification [50] and signal processing [51]. A learning automaton performs an action and then classifies it as desirable or not based on a reinforcement signal (negative/penalty or positive/reward) from the environment [52]. The learning scheme then updates the reward or penalty on this action depending on the reinforcement signal. The set of actions processed by learning automata is problem dependent and varies from one application to another, for example, it could be choices of a parameter value in [50], heuristics in [51] or partitions in [53].

---

[1]http://www.asap.cs.nott.ac.uk/external/chesc2011/

More formally, a learning automaton is defined as a quadruple (A, β, p, U), where A is the action set, β (equals to 0 or 1) represents the (penalty or reward) *feedback* or *reinforcement* signal obtained from the environment after taking the chosen action a_i at a given time t, p is the (action) selection probability vector, where each entry indicates the probability of an action being selected, and U is the update scheme. The action set A is commonly considered to be a finite set, i.e. $A = \{a_1, a_2, \ldots, a_r\}$. Thus, the traditional model of a learning automaton is referred to *finite action learning automaton* [52], which is denoted as LA in this paper. At a given time (t), the action selection method chooses an action (say, a_i) based on p. After the selected action a_i is performed, p is updated by the scheme U as defined in Equation (1) and (2)) using the feedback $\beta(t)$ received from the environment. The sum of all selection probabilities in p is always equal to 1.

If a_i is the action chosen at time step t

$$p_i(t+1) = p_i(t) + \lambda^{(1)}\beta(t)(1-p_i(t)) - \lambda^{(2)}(1-\beta(t))p_i(t) \quad (1)$$

For other actions $a_j \neq a_i$,

$$p_j(t+1) = p_j(t) - \lambda^{(1)}\beta(t)(p_j(t)) + \lambda^{(2)}(1-\beta(t))\left[\frac{1}{r-1} - p_j(t)\right] \quad (2)$$

The parameters $\lambda^{(1)}$ and $\lambda^{(2)}$ are the reward and penalty rates respectively. When $\lambda^{(1)} = \lambda^{(2)}$, the model is referred as *linear reward-penalty (L_{R-P})*. In case of $\lambda^{(2)} = 0$, it is referred to as *linear reward-inaction (L_{R-I})*. If $\lambda^{(2)} < \lambda^{(1)}$, it is called *linear reward-ϵ-penalty ($L_{R-\epsilon P}$)*.

C. Vehicle Crashworthiness Problem (VCP)

In the automotive industry, *crashworthiness* refers to the ability of a vehicle and its components to protect its occupants during an impact or crash [54]. The crashworthiness design of vehicles is of special importance, yet, highly demanding for high-quality and low-cost industrial products. The structural optimisation of the vehicle design involves multiple criteria to be considered. [55] presented a multiobjective model for the vehicle design which minimises three objectives: weight ($Mass$), acceleration characteristics (A_{in}) and toe-board intrusion ($Intrusion$). More specifically, the weight of the vehicle is to be minimised for enabling economic mass production. An important goal of the vehicle design is to reduce any potential harm to occupant(s). When the front of a vehicle hits an object, it first begins to decelerate by the impact. The velocity decreases to zero when the vehicle comes to a halt. As the vehicle begins to bounce back, the velocity increases. This acceleration can cause head injuries to occupant(s) and be dangerous to other road users, because the vehicle is now moving in the opposite direction. To reduce the acceleration due to collision and possible head injuries to occupants caused by the worst scenario of the acceleration pulse [56], minimising an integration of collision acceleration between 0.05-0.07 seconds in the 'full frontal crash' is set as the second objective. Another mechanical injury to occupants may come from the toe-board intrusion during the crash. It could hurt the knee trajectories of occupants and influence the steering

of the vehicle. Therefore, minimising the toe-board intrusion in the 40% offset frontal crash is chosen as the third objective. The decision variables are the thickness of five predefined reinforced points, say x_1, x_2, x_3, x_4 and x_5, around the frontal structure of a vehicle. Each decision variable is between 1 mm to 3 mm. The VCP model is formulated as follows.

$$\textbf{minimise } F(X) = [(Mass, A_{in}, Intrusion)]$$
$$\textbf{subject to } 1.0 \leq x_i \leq 3.0, \qquad i = 1, 2, \ldots, 5 \quad (3)$$
$$X = (x_1, x_2, \ldots, x_5)^T$$

where,

$$Mass = 1640.2823 + 2.3573285x_1 + 2.3220035x_2 \\ + 4.5688768x_3 + 7.7213633x_4 + 4.4559504x_5 \quad (4)$$

$$A_{in} = 6.5856 + 1.15x_1 - 1.0427x_2 + 0.9738x_3 \\ + 0.8364x_4 - 0.3695x_1x_4 + 0.0861x_1x_5 + 0.3628x_2x_4 \quad (5) \\ - 0.1106x_1^2 - 0.3437x_3^2 + 0.1764x_4^2$$

$$Intrusion = -0.0551 + 0.0181x_1 + 0.1024x_2 \\ + 0.0421x_3 - 0.0073x_1x_2 + 0.024x_2x_3 - 0.0118x_2x_4 \quad (6) \\ - 0.0204x_3x_4 - 0.008x_3x_5 - 0.0241x_2^2 + 0.0109x_4^2$$

Apart from the original problem instance requiring optimisation of all the three objectives, we formed additional instances by considering pairs of objectives leading to four VCPs, including VC1: minimise {Mass, A_{in}, Intrusion}, VC2: minimise {Mass, A_{in}}, VC3: minimise {Mass, Intrusion} and VC4: minimise {A_{in}, Intrusion} for our study.

III. METHODOLOGY

The proposed learning automata based multiobjective hyperheuristic framework enabling control of multiple MOEAs operates as illustrated in Algorithm 1. Firstly, given a set of MOEAs (H), the initialisation process takes place to set up the relevant data structures (line 1). Our learning automaton requires the maintenance of a *transition matrix* (P) which describes the selection probabilities of metaheuristics transitioning from the previously selected metaheuristics. At the end of initialisation step, the transition matrix is set up and a sub- or full set of MOEAs (A) is determined as the input of the following learning scheme, as well as the input heuristic (h_i) and population (Pop_{curr}) (See Section III-A).

The chosen MOEA (h_i) is applied (line 3) to the incumbent set of solutions (Pop_{curr}) to the problem instance dealt with for a fixed number of generations/iterations (g), producing a new set of solutions (Pop_{next}). The new population then replaces the current population (line 4). If the conditions of switching to another metaheuristic (line 5) are satisfied, the reinforcement learning scheme updates the transition matrix (line 6) based on the feedback received during the search. Afterwards, the selection mechanism makes use of the updated transition matrix (P) to decide which MOEA (h_i) to run in the next iteration. Then all those steps are repeated until the termination criteria are satisfied.

The framework consists of four key components: initialisation process, reinforcement learning scheme, metaheuristic

(action) selection method and the method deciding when to switch to another metaheuristic. Two multiobjective hyper-heuristics, referred to as HH-LA and HH-RILA are designed under this framework in this study. HH-LA and HH-RILA differ only in their initialisation processes. The remaining components are the same. The following subsections describe each component in detail.

A. Initialisation

HH-LA utilises all r MOEAs and the transition matrix P is initially created so that each MOEA has the same probability of being selected, i.e., $1/r$. The initial population for HH-LA is generated randomly.

HH-RILA uses a more elaborate initialisation process. We propose a ranking scheme to form a reduced subset of MOEAs, eliminating the ones with relatively poor performance. The ranking process begins with running each MOEA successively for a number of stages. The number of stages is set to the number of low level metaheuristics for giving each MOEA an equal chance to show its performance. Initial population is generated randomly. The resultant population obtained at the end of each stage is directly fed into the following stage for each MOEA. The hypervolume values for all resultant populations obtained at the end of each stage from each MOEA is computed based on the normalised objective values, i.e., $(f_i(x) - f_i^{min})/(f_i^{max} - f_i^{min})$ for the ith objective, where the extreme objective values for each dimension, i.e. f_i^{max}, f_i^{min} are updated using the maximum and minimum values found so-far by all MOEAs. This process enables performance comparison of all MOEAs with respect to hypervolume for all stages. Then we count the number of stages (frequencies), denoted as $Frq_{best}(h_i)$ (the higher, the better) that each MOEA becomes the best performing algorithm out of all stages. These counts are then used for ranking all MOEAs. If more than one MOEA has the same rank, ties are broken

---

Algorithm 1: Learning Automaton based Hyper-heuristic Framework

---

Pop_{curr} : set (population) of input solutions,
Pop_{next} : set of solutions surviving to the next stage,
H: set of metaheuristics (MOEAs) $\{h_1, ..., h_i, ..., h_r\}$,
P: transition matrix, g: fixed number of generations

1 $[A, P, h_i, Pop_{curr}] \leftarrow$ Initialise(H) ; // $A \subseteq H$
2 **while** *(termination criteria not satisfied)* **do**
3     $Pop_{next} \leftarrow$ ApplyMetaheuristic(h_i, Pop_{curr}, g);
4     $Pop_{curr} \leftarrow$ Replace(Pop_{curr}, Pop_{next});
    // Decide whether to switch to
      another metaheuristic
5   **if** *(switch())* **then**
6   LearningAutomataUpdateScheme(P);
       // Decision Point for
         metaheuristic selection
7        $h_i \leftarrow$ SelectMetaheuristic(P, A);
    end
  end

---

using the diversification indicator of generalised spread (the smaller, the better). Then MOEA(s) that rank worse than the median MOEA get excluded from the low level MOEA set. For example, if h_3 becomes the top ranking metaheuristic in all three stages, while h_1 and h_2 do not in any of the three stages, then $Frq_{best}(h_1), Frq_{best}(h_2), Frq_{best}(h_3)$ are 0, 0 and 3, respectively. Consequently, the rank of each MOEA with respect to normalised hypervolume is 2, 2 and 1. Suppose h_1 has a smaller generalised spread value than h_2, then final ranks of h_1, h_2, h_3 are 2, 3 and 1, respectively. Eventually, h_2 gets excluded from the following stage of the learning process.

Then HH-RILA operates as HH-LA with a reduced subset of low level MOEAs for the remaining search process using the final population from the best ranking MOEA as input.

B. Reinforcement Learning Scheme

The reinforcement learning scheme sits at the core of the metaheuristic selection process. The system learns a mapping (or policy) from situations to actions through a trial-and-error process with the goal of maximising the overall reward. To explore the possible cooperation among different action pairs, the learning scheme in this study updates the transition probability $(p_{(i,j)})$ from a preceding action (a_i) to a given successor (a_j), depending on the performance after applying a_j. The chosen heuristics logically form a chain of a heuristic sequence as the search progresses. Although there are previous studies ([9], [48], [57], [58], [59]) using some notion of transition probabilities to keep track of the performance of heuristics invoked successively, none of them employed the same reinforcement learning scheme as we proposed. More importantly, all the previously mentioned algorithms were tested on single objective optimisation problems under a single point based search framework managing move operators rather than metaheuristics.

In the proposed learning scheme, an action (say h_i) corresponds to the selection of an MOEA, and the tth time step is analogous to the tth decision point when an MOEA is selected and applied to the trade-off solutions in hand. The *linear reward-penalty* scheme is used to update the transition probability from h_i to h_j at time $(t + 1)$, i.e. $p_{(i,j)}(t + 1)$. The update is performed as provided in Equation (7) and (8) [52]. The value of $\beta(t)$ is set to 1 for positive (or preferable) feedback, 0 otherwise.

If the successor metaheuristic h_j of h_i is selected:

$$p_{(i,j)}(t + 1) = p_{(i,j)}(t) + \lambda_{(i,j)}(t)\beta(t)(1 - p_{(i,j)}(t)) - \lambda_{(i,j)}(t)(1 - \beta(t))p_{(i,j)}(t) \quad (7)$$

For the rest of the metaheuristics that are not chosen, indexed as l, where $l \neq j$:

$$p_{(i,l)}(t + 1) = p_{(i,l)}(t) - \lambda_{(i,l)}(t)\beta(t)p_{(i,l)}(t) + \lambda_{(i,l)}(t)(1 - \beta(t))\left[\frac{1}{r - 1} - p_{(i,l)}(t)\right] \quad (8)$$

We use the 'change in the hypervolume value' measured before and after selecting and applying an MOEA for rewarding/penalising during the learning process for two reasons. First, hypervolume is the only known unary *Pareto compliant*

indicator ([19], [60]) i.e., if a PF P1 dominates P2, the indicator value of P1 should be better than that of P2. Second, theoretical studies show that maximising the hypervolume indicator during the search is equivalent to optimising the overall objective leading to an optimal approximation of the true PF ([61], [62]).

Due to the non-stationary nature of the search process, it is reasonable to give more weight to the recent rewards than the long-past ones. One of the common ways of doing this is to discount the past reward at a fixed ratio (α) [63]. The reward is denoted as $Q_{(i,j)}(k+1)$, meaning the estimated action value of the transition pair (h_i, h_j) occurring its $(k+1)^{th}$ times at the tth decision point:

$$Q_{(i,j)}(k+1) = Q_{(i,j)}(k) + \alpha[r_{(i,j)}(k+1) - Q_{(i,j)}(k)] \quad (9)$$

where $r_{(i,j)(k+1)}$ is the current reward obtained by pair (h_i, h_j)

$$r_{(i,j)}(k+1) = v_j(t) - v_i(t-1) \quad (10)$$

where $v_j(t)$ is the hypervolume obtained by executing the action h_j at the current t^{th} decision point, $v_i(t-1)$ is the hypervolume obtained by action h_i at the $(t-1)^{th}$ decision point. α is commonly fixed as 0.1 [63] as in this study. The hypervolume here is computed in the normalised objective space as described in Section III-A.

Given the varying performance of each MOEA pair (h_i, h_j) during the search, instead of fixing the reward and penalty rates of $\lambda_{(i,j)}$, it is adaptively updated using the estimated action value of each transition pair ($Q_{(i,j)}$) at each decision point. The calculation of λ is used to update both reward and penalty rates as follows.

$$\lambda_{(i,j)}(t) = 0.1 + mQ_{(i,j)}(k+1) \quad (11)$$

where m is fixed as small positive multipliers (e.g. 2) to amplify the effect of the estimated action value $Q_{(i,j)}(t)$ on the reward/penalty parameter.

Due to the nature of the search space and amplifying multipliers, it is possible that the adaptive reward and penalty rates ($\lambda_{(i,j)}(t)$) can get out of the [0,1] range and so the transition probabilities. In such cases, the value of $\lambda_{(i,j)}(t)$ is reset to the closest extreme value (0 or 1) ensuring that it stays within the range.

C. Metaheuristic Selection Method

In reinforcement learning, in order to take an action (i.e., choosing a metaheuristic), a selection method is required. This method is normally based on a function of the selection probabilities (utility values) to select an action at a given certain point. Several selection methods are commonly used in the scientific literature, such as roulette wheel, or greedy [63]. Those methods differ when exploring new actions and exploiting the knowledge obtained from the previous actions. The roulette wheel selection method chooses an action with a probability proportional to its utility value. The advantage of this method is its straightforwardness and it does not introduce any extra parameters. However, it has less chance to exploit the best-so-far actions when compared to the other selection methods, in particular when the selection probabilities of actions are

similar. The greedy selection method only chooses the action with the highest selection probability. As a drawback, this method could overlook the other potentially good performing actions which might give higher rewards in the later stages. Further details on different selection methods can be found in [63]. Each selection method has its strengths and weaknesses. To exploit the merits of both roulette and greedy selection methods, we propose a new selection method, named as ϵ-RouletteGreedy selection. The main idea is that the selection method first focuses on exploring different transition pairs by performing a certain number of trials to get a better view of the pairwise performances of metaheuristics at the early stage. Then, the selection method becomes more and more greedy exploiting the accumulated knowledge.

The proposed selection method works as follows. The exploration phase parameter τ is fixed to a value in [0,1]. During the first $\tau n_{totalIter}$ iterations, where $n_{totalIter}$ is the total number of iterations, roulette wheel selection is solely used to choose an action (say, h_j) out of all the possible successors of action h_i based on the transition probability $p_{(i,j)}$. Following this exploration phase, the probability ϵ of applying the greedy selection method is increased linearly by the formula $\tau + (1.0 - \tau)n_{iter}/n_{totalIter}$, where n_{iter} denotes the number of iterations has passed since the beginning of the algorithm. We randomly generate a value between 0 and 1. If that value is less than or equal to ϵ, the best action (with the highest transition probability from the previously selected action) is chosen to be performed at the next decision point. If the random value is greater than ϵ, the next action is selected by roulette wheel selection method.

D. Switching to Another Metaheuristic

In this study, we propose a threshold method to stop the application of a selected MOEA (h_i) repeatedly, enabling the hyper-heuristic to switch to another MOEA, adaptively. A selected metaheuristic is applied as long as there is an improvement in the hypervolume as compared to the previous iteration above an expected level. Hence, application of the selected MOEA halts if the hypervolume improvement $\delta(v_{iter})$ is less than a threshold value of Δ_v at a given iteration, or the maximum number of iterations (denoted as K) for applying a low level MOEA is exceeded. The hypervolume improvement (change) $\delta(v_{iter})$ is computed as $(v_{iter} - v_{(iter-1)})/v_{(iter-1)}$, where v_{iter} is the hypervolume of the trade-off solutions obtained after the application of h_i at the current iteration, and $v_{(iter-1)}$ is the hypervolume obtained from h_i at the previous iteration.

IV. COMPUTATIONAL EXPERIMENTS

The proposed multiobjective selection hyper-heuristics, HH-LA and HH-RILA controlling three low level MOEAs {NSGA-II, SPEA2 and IBEA} are studied using a range of three-objective benchmark functions from the WFG [20] and DTLZ [21] test suites. The number of stages in the initialisation for HH-RILA is set to 3. The performances of HH-LA and HH-RILA are not only compared to each

individual low level MOEA, but also to random choice hyper-heuristic (HH-RC) serving as a reference approach utilising no learning as well as the online learning hyper-heuristic of HH-CF [14] using the same set of low level MOEAs. The jMetal software platform [64] embedding implementations of the WFG and DTLZ problems and three low level MOEAs are used for the development of all the algorithms experimented within this study.

A. Experimental Settings

Each experiment with an algorithm is repeated for 30 times on each problem instance. The WFG and DTLZ benchmark functions are all parameterised. Each WFG benchmark function has 20 distance and 4 position (total 24) parameters, while DTLZ1, 2-6 and 7 have 7, 12 and 22 parameters, respectively. Those parameter values are fixed as in [20] for the WFG and [21] for the DTLZ problems.

It is commonly known that the performance of meta-heuristics can be improved through *parameter tuning*, that is, detecting the best settings (configuration) for the algorithmic parameters ([65], [66]). Considering the large set of parameters and their values associated with the proposed hyper-heuristics and MOEAs used in this study, it is not feasible to test all the combinations of settings considering the immense amount of required computational budget. Instead, parameters of HH-LA and HH-RILA are tuned based on the Taguchi experimental design [67]. Whereas, the recommended configurations and parameter settings are used for all the other algorithms, including MOEAs ([16], [17], [68], [69]) and HH-CF [14] from the scientific literature. Simulated binary crossover (also known as SBX) and polynomial mutation [70] are used as the MOEA operators. The distribution parameters of the crossover and mutation operators are fixed as $\{\eta_c = 20.0\}$ and $\{\eta_m = 20.0\}$, respectively. The crossover and mutation probabilities are set to $\{p_c = 0.9\}$ and $\{p_m = 1/n_p\}$, where n_p is the number of parameters. Parents are selected using the binary tournament operator [71]. The maximum number of solution evaluations for each WFG and DTLZ problem is set to 50,000 and 100,000, respectively [68]. This particular setting is always maintained for all algorithms tested in this study for a fair performance comparison between them. The population and archive sizes are both fixed as 100 for MOEAs. The number of iterations for HH-CF and HH-RC is set to the recommended value of 25, and intensification parameter of HH-CF to 100 [14]. The number of generations for each iteration is fixed as $g = 10$ for HH-LA and HH-RILA.

For a fair comparison, the number of evaluations used for the initialisation in HH-RILA are deducted from the total. As mentioned above, parameters of the proposed hyper-heuristics are tuned for an improved performance. The parameter tuning experiments and sensitivity analysis of each parameter for HH-LA and HH-RILA are provided in the following subsection.

B. Parameter Tuning of HH-LA and HH-RILA and Sensitivity Analysis

Our multiobjective selection hyper-heuristics contains four main parameters: exploration phase τ, reward/penalty multiplier m, maximum number of iterations K for applying a low
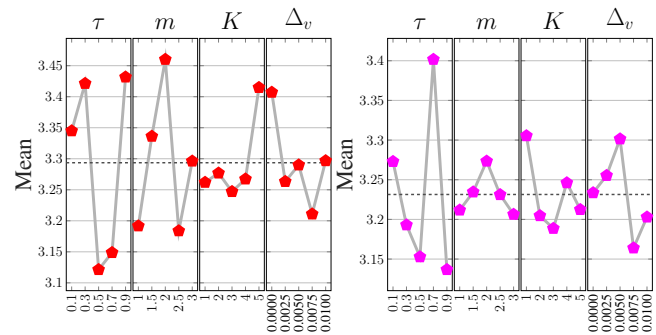

Fig. 1: Main effects plots for HH-LA (left) and HH-RILA (right) for each parameter: exploration phase (τ), multiplier (m), maximum iterations (K) for applying a low leve MOEA, and hypervolume improvement threshold (Δ_v).

level MOEA, and hypervolume improvement threshold Δ_v. Five different values for each parameter are considered: $\tau \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, $m \in \{1.0, 1.5, 2.0, 2.5, 3.0\}$, $K \{1, 2, 3, 4, 5\}$, $\Delta_v \in \{0.0, 0.0025, 0.005, 0.0075, 0.01\}$. Even with this sample of five settings for each of the four parameters, 625 parameter tuning experiments would have been required for testing all combinations of the parameter settings. In this study, the Taguchi orthogonal arrays experimental design method ([67], [72]) is used for parameter tuning. Sampling the configurations based on the orthogonal array, denoted as L_{25}, reduces the number of parameter tuning experiments to 25 configurations for each algorithm, which are tested on the benchmark functions.

The measurement used during the tuning experiments is μ_{norm}. The original μ_{norm} is defined for the minimisation problems. Since we are maximising hypervolume, we slightly modify the formulation of μ_{norm} as follows. Let $S_{(x,n)}$ be the set of hypervolume (30 hypervolume values in our case resulting from 30 trials) obtained by an algorithm x, where $x \in X$ on a problem n, where $n \in N$; X and N are the sets of algorithms and problems, respectively. Let $S_n^{min} = MIN_{\forall s \in S_{(x,n)}, \forall x \in X}$ be the minimum and $S_n^{max} = MAX_{\forall s \in S_{(x,n)}, \forall x \in X}$ be the maximum hypervolume obtained by all the algorithms on a problem n. The normalised hypervolume of an algorithm x on a problem n is computed as $f_{(x,n)}^{norm} = \frac{S_n^{max} - AVG_{s \in S_{(x,n)}}(s)}{S_n^{max} - S_n^{min}}$. The average of $f_{(x,n)}^{norm}$ defined as $\mu_{norm}(x) = AVG_{\forall n \in N}(f_{(x,n)}^{norm})$ serves as the measurement for the tuning experiments. The lower the $\mu_{norm}(x)$ value, the better the performance of the algorithm x.

The main effects plots in Figure 1 indicate the mean effect of each parameter setting on the performances of HH-LA and HH-RILA. The parameter setting that achieves the lowest mean μ_{norm} averaged across all trials using that setting regardless of the remaining parameter settings would be the best value for that parameter. Thus, the best configuration for HH-LA is $\{\tau = 0.5, m = 2.5, K = 3, \Delta_v = 0.0075\}$, and for HH-RILA is $\{\tau = 0.9, m = 3.0, K = 3, \Delta_v = 0.0075\}$. Both settings are used in this paper for the rest of the experiments.

Analysis of Variance (ANOVA) [73] test is performed to observe how sensitive the performance of proposed hyper-heuristics to the parametric settings is by looking into the

significance and contribution (in percentage) of each parameter. Table I shows that exploration phase parameter τ has the most significant influence on the performance of both HH-LA and HH-RILA at a significance level of 5% (i.e., p-value < 0.05). The parameter τ has the highest percentage contribution of 43.93% and 53.61% to the performance of HH-LA and HH-RILA, respectively. The reward/penalty multiplier m also significantly contribute to the performance of HH-LA with the second largest percentage contribution of 25.75%, while this parameter has almost no contribution (3.16%) to the performance of HH-RILA. The remaining two parameters are not significantly influential on the performance of either proposed hyper-heuristics.

TABLE I: ANOVA test to identify the contribution (%) of each parameter for HH-LA and HH-RILA (DoF: degrees of freedom, SS: sum of squares, MS: mean squares, F: variance ratio).

HH-LA

Parameters	DoF	SS	MS	F	p-value	contribution (%)
τ	4	0.44	0.11	8.16	0.0063	43.93
m	4	0.26	0.06	4.78	0.0289	25.75
K	4	0.09	0.02	1.73	0.2357	9.32
Δ_v	4	0.10	0.03	1.90	0.2037	10.24
Residual	8	0.11	0.01			
Total	24					100

HH-RILA

Parameters	DoF	SS	MS	F	p-value	contribution (%)
τ	4	0.24	0.06	5.05	0.0250	53.61
m	4	0.01	0.00	0.30	0.8715	3.16
K	4	0.04	0.01	0.92	0.4983	9.74
Δ_v	4	0.05	0.01	1.16	0.3972	12.27
Residual	8	0.09	0.01			
Total	24					100

C. Experimental Results on WFG and DTLZ

In this section, we use hypervolume as the main performance indicator. One-tailed Wilcoxon rank-sum test (also known as Mann-Whitney U test) is applied based on the raw hypervolume values obtained from 30 trials of each algorithm to test if there is a statistically significant performance difference between a pair of algorithms. The significance level is set to 5%. The reference (or nadir) point (denoted as r) for the WFG and DTLZ benchmark problems are chosen as follows. For each WFG problem, the reference point is set as $r^i = 2i + 1$, where $i = 1, 2, ..., k$ is the index of the objective and k is the total objective number. Thus, for each WFG problem, the reference point is (3, 5, 7). The reference point for DTLZ problems is set as $r^i = 0.5$ for DTLZ1, $r^i = 1.0$ for DTLZ2 to DTLZ6 and $r^i = 1.0$ if $i < k$, otherwise, $r^k = 2k$ for DTLZ7.

The convergence indicator $\epsilon+$ is utilised as an additional performance comparison indicator. We notice that in some cases, the performance differences between algorithms are not distinguishable if the raw values are plotted directly. Here only for the visualisation purposes, we map the raw hypervolume/$\epsilon+$ value into the range of [0,1] via normalisation using the extreme (minimum and maximum) values collected from all algorithms over 30 trials on each instance, then the mean hypervolume and $\epsilon+$ values are plotted in Figure 2. Higher the hypervolume or lower the $\epsilon+$ value means a better performance.

Figure 2 shows that IBEA performs the best on WFG benchmark with respect to both hypervolume and $\epsilon+$ in the overall. HH-RILA and HH-LA follow the performance of IBEA closely. NSGA-II clearly performs the worst on WFG. The performance of IBEA gets much poorer and becomes overall the worst approach for the DTLZ benchmark functions with respect to both metrics. SPEA2 performs the best on over half of DTLZ benchmark. HH-RILA and HH-LA always achieve the second best performance on most DTLZ benchmark or even the best on DTLZ7. In addition, the hypervolume based performance ranking of all algorithms on each benchmark problem is almost fully consistent with the $\epsilon+$ based ranking except for WFG1. On WFG1, IBEA achieves the best rank with respect to hypervolume, however, IBEA performs slightly worse than SPEA2 on WFG1 with respect to the $\epsilon+$ indicator. This inconsistency, also discussed in [60], is possibly due to the different working principles of both indicators.

One-tailed Wilcoxon rank-sum test at 5% significance level is conducted on the performance of each pair of algorithms with respect to hypervolume. The statistical test results are summarised in Table II and we have the following observations.

In the overall, both of our MOHHs deliver a better performance than any of the individual MOEAs run on its own on the WFG and DTLZ benchmarks. The statistical test results show that HH-LA and HH-RILA outperform NSGA-II on all nine WFG benchmark functions while 3 out of 7 DTLZ problems, including DTLZ1-2 and DTLZ5. HH-RILA additionally performs significantly better than NSGA-II on DTLZ6 and DTLZ7. HH-LA and HH-RILA perform significantly better than SPEA2 on the same 8 out of 9 WFG benchmark functions including WFG1, WFG3-9. HH-RILA additionally outperforms SPEA2 on DTLZ2 and DTLZ7. Although IBEA delivers a good overall performance on the WFG benchmark, both our algorithms still manage to outperform IBEA on 5 out of 7 DTLZ problems including DTLZ1, DTLZ3 and DTLZ5-7. HH-RILA also performs significantly better than IBEA on WFG2.

Both HH-LA and HH-RILA outperform HH-CF and HH-RC. Specifically, both of our hyper-heuristics perform significantly better than HH-CF on 11 benchmark functions out of total 16, including the same eight WFG benchmark functions (WFG1, WFG3-9) and three DTLZ benchmark functions (DTLZ1-3 for HH-LA, while DTLZ1-2 and DTLZ6 for HH-RILA). The performance difference between each of the proposed hyper-heuristics and HH-RC is statistically significant with respect to hypervolume on 10 out of 16 problems which include the same seven WFG benchmark functions (WFG3-9) and three slightly different DTLZ problems: HH-LA outperforms HH-RC on DTLZ1-2 and DTLZ5, while DTLZ1-2 and DTLZ7 for HH-RILA. HH-CF only outperforms HH-RC on DTLZ7, while they perform similarly on 8 out of
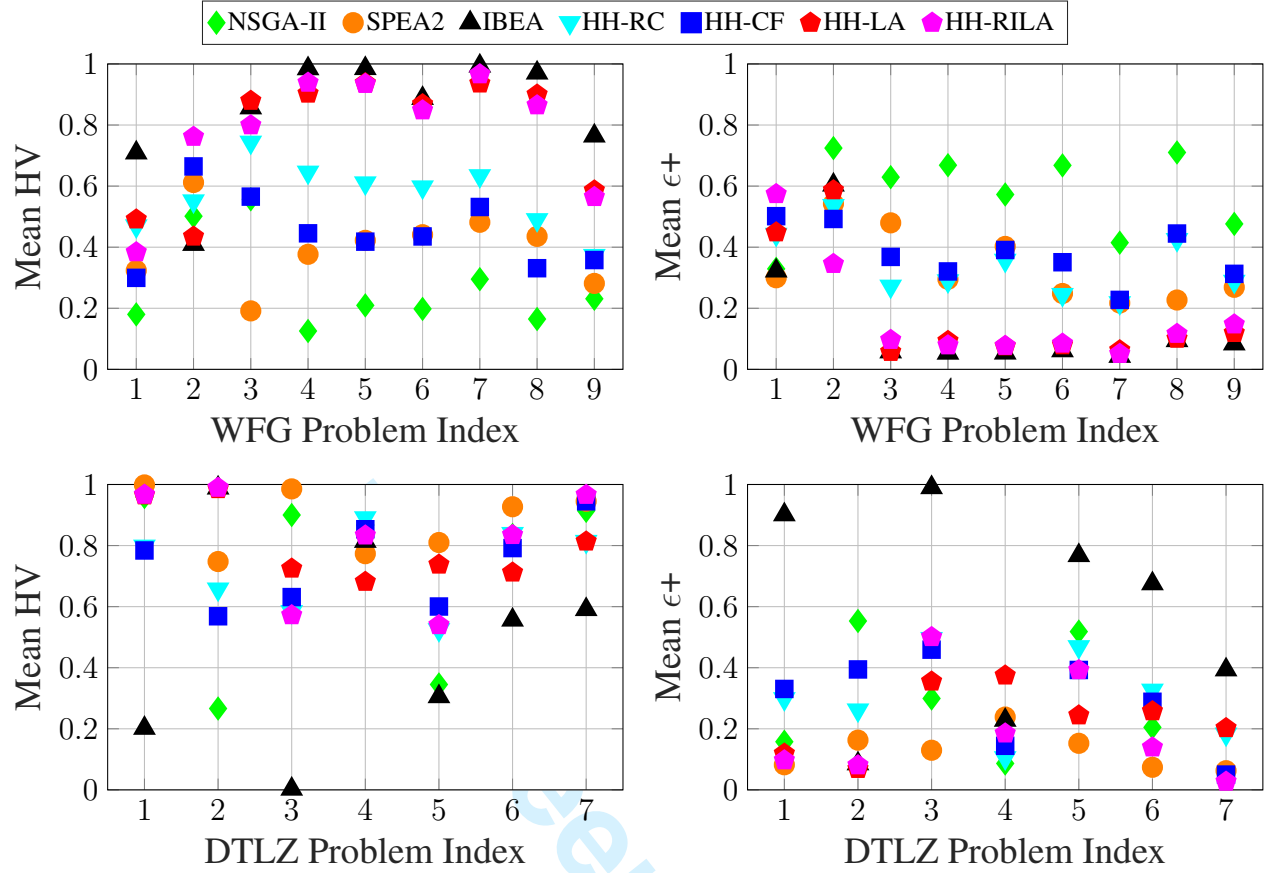
Fig. 2: Performance comparison of all the algorithms with respect to hypervolume and $\epsilon+$ on 3D WFG and DTLZ problems.

TABLE II: One-tailed Wilcoxon rank-sum test at 5% significance level on WFG and DTLZ benchmark problems with respect to hypervolume. 'W' and 'D' are short for 'WFG' and 'DTLZ' respectively. '>' means the significantly better than, '<' significantly worse than, '∼' no significant difference.

	W1	W2	W3	W4	W5	W6	W7	W8	W9	D1	D2	D3	D4	D5	D6	D7
HH-LA vs HH-CF	>	∼	>	>	>	>	>	>	>	>	>	>	∼	∼	∼	∼
HH-LA vs HH-RC	∼	∼	>	>	>	>	>	>	>	>	∼	∼	∼	>	∼	∼
HH-LA vs NSGA-II	>	>	>	>	>	>	>	>	>	>	>	∼	∼	>	∼	<
HH-LA vs SPEA2	>	∼	>	>	>	>	>	>	>	<	<	<	∼	<	<	<
HH-LA vs IBEA	<	∼	∼	<	<	∼	<	<	<	>	∼	>	∼	>	>	>
HH-CF vs HH-RC	<	∼	<	<	<	<	<	<	∼	∼	∼	∼	∼	∼	∼	>
HH-RILA vs HH-CF	>	∼	>	>	>	>	>	>	>	>	>	∼	∼	∼	>	∼
HH-RILA vs HH-RC	∼	∼	>	>	>	>	>	>	>	>	∼	∼	∼	∼	∼	>
HH-RILA vs HH-LA	>	>	<	>	∼	<	>	<	∼	>	∼	∼	∼	<	>	∼
HH-RILA vs NSGA-II	>	>	>	>	>	>	>	>	>	>	>	∼	<	>	<	<
HH-RILA vs SPEA2	>	∼	>	>	>	>	>	>	>	<	<	<	∼	<	<	<
HH-RILA vs IBEA	<	>	<	<	<	<	<	<	<	>	∼	>	∼	>	>	>
NSGA-II vs SPEA2	<	∼	>	<	<	<	<	<	∼	<	<	<	>	<	<	<
NSGA-II vs IBEA	<	∼	<	<	<	<	<	<	<	>	>	>	∼	>	>	>
SPEA2 vs IBEA	<	∼	<	<	<	<	<	<	<	>	<	>	<	>	>	>

16 problems (WFG2, WFG9 and DTLZ1-6). HH-CF delivers a significantly worse performance than HH-RC on the rest of the seven problems (WFG1 and WFG3-8).

As for the performance comparison between HH-LA and HH-RILA, HH-LA is slightly better than HH-RILA in the overall on the WFG problems. This performance difference is statistically significant on four WFG problems including WFG1, WFG3, WFG6 and WFG8, while HH-RILA performs significantly better than HH-LA on three WFG problems: WFG2, WFG4 and WFG7. However, considering DTLZ

benchmark, HH-RILA performs slightly better than HH-LA in the overall. This performance difference is statistically significant on DTLZ1 and DTLZ6, while HH-LA only outperforms HH-RILA on DTLZ5.

D. Analysis of Hyper-heuristics on WFG and DTLZ

1) Utilisation of Low Level Metaheuristics: The 'utilisation rate' of a low level metaheuristic is the number of invocations of this metaheuristic divided by the total number of meta-heuristic selection decision points in a given trial. The mean
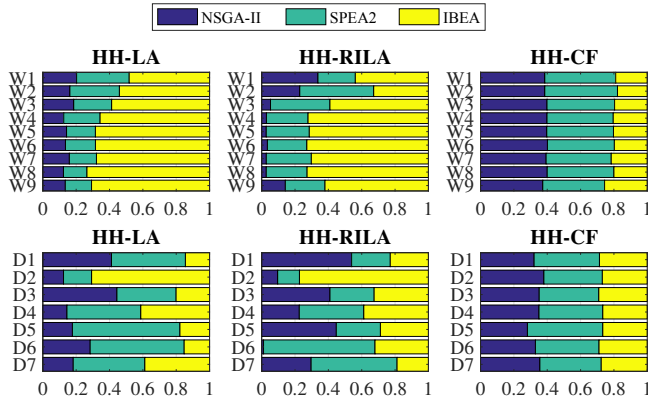
Fig. 3: The mean utilisation rate of each metaheuristic by HH-LA (left), HH-RILA (middle) and HH-CF (right) over 30 trials on WFG ('W') and DTLZ ('D').
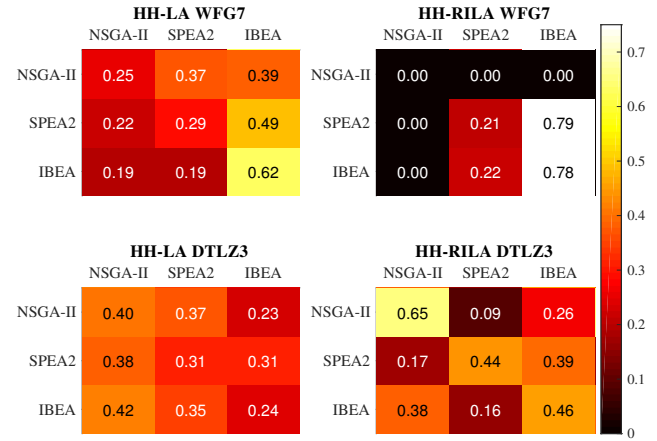

Fig. 4: The averaged transition probability matrices (over 30 trials) produced by HH-LA (left column) and HH-RILA (right column) while solving WFG7 and DTLZ3. The lighter the colour, the higher the transition probability.

utilisation rates of the three MOEAs i.e., NSGA-II, SPEA2 and IBEA averaged over 30 trials on the WFG and DTLZ benchmark functions produced by HH-LA, HH-RILA and HH-CF [14] are illustrated in Figure 3.

Figure 3 shows the differences in learning characteristics of these three online learning MOHHs. Firstly, HH-LA and HH-RILA provide a bias towards using the best performing MOEA with respect to hypervolume. Specifically, both HH-LA and HH-RILA choose IBEA and SPEA2 more frequently while solving the WFG and DTLZ problems, respectively. This is not surprising, considering that hypervolume serves as the main guidance in the learning mechanisms of our hyper-heuristics. Secondly, in certain cases, such as WFG4-8 and DTLZ6, HH-RILA almost exclude NSGA-II which is the worst performed MOEA on those problems. Interestingly, HH-CF generates a similar utilisation rate for low level MOEAs across different problem sets. On average, HH-CF uses NSGA-II, SPEA2 and IBEA for 40%, 40% and 20% of all the decision points, respectively, on the WFG benchmark. Similarly, HH-CF uses NSGA-II, SPEA2 and IBEA for 34%, 38% and 28%, respectively, on the DTLZ benchmark. This might indicate that the adaptation mechanism in HH-CF has some issues controlling these three low level metaheuristics properly on different problem instances.

2) An Analysis of the Transition Probabilities: The proposed hyper-heuristics embed a learning mechanism which maintains the transition probabilities between any pair of MOEAs. Figure 4 provides the final transition probability matrices obtained by HH-LA and HH-RILA averaged over 30 trials for the sample cases of WFG7 and DTLZ3.

Figure 4 illustrates that both HH-LA and HH-RILA yield higher probability entries preferring transitions to IBEA than to other MOEAs for WFG7. This is consistent with the performance assessment of each individual MOEA (Figure 2), which shows that IBEA performs the best on WFG7. Moreover, HH-RILA excludes the worst performing MOEA i.e., NSGA-II after the initialisation stage for solving WFG7. This is likely the reason why HH-RILA performs significantly better than HH-LA on WFG7.

DTLZ3 is an interesting case. IBEA delivers a better perfor-

mance in the early stages, but stagnates and even deteriorates later during the search process. Due to the misleading performance of IBEA in the early stage, HH-RILA rewards IBEA more than the other MOEAs, while excluding the ones with potentially good performance, such as, SPEA2. Consequently, HH-RILA ends up performing significantly worse than HH-LA on DTLZ3.

In summary, the proposed learning mechanism is capable of adaptively updating the transition probabilities between pairs of MOEAs giving bias towards the right algorithms (with good performance) during the search process in an online manner. Moreover, the ranking initialisation scheme is, in some cases, capable of improving the overall performance significantly by detecting and excluding potentially poor performing MOEA(s) in the early stages of the search.

3) An Analysis of Approximate Pareto Fronts: So far, IBEA is a strong competitor of HH-LA and HH-RILA with respect to hypervolume. To get more insights on the distribution of solutions from IBEA and the proposed hyper-heuristics, PFs obtained from HH-RILA and IBEA for WFG7 and DTLZ3 are illustrated in Figure 5. HH-LA produces PFs very similar to HH-RILA on almost all problems, and so we focus on HH-RILA here.

Figure 5 demonstrates that IBEA is prone to be trapped at a local optimum. IBEA produces uneven solution distribution for WFG7, leaving clear gaps between the boundary and inner regions, whereas HH-RILA reaches a better solution distribution for this problem.

IBEA performs poorly on DTLZ3. All the solutions are clustered around the 'corner' points which suggests that the performance of IBEA degrades during the search process. This interesting behaviour of IBEA has also been observed previously by Tušar *et al.* [69] (in Figure 7) and [74]. More importantly, solutions from HH-RILA clearly spread much more evenly on the front than IBEA, possibly due to the utilisation of multiple MOEAs.
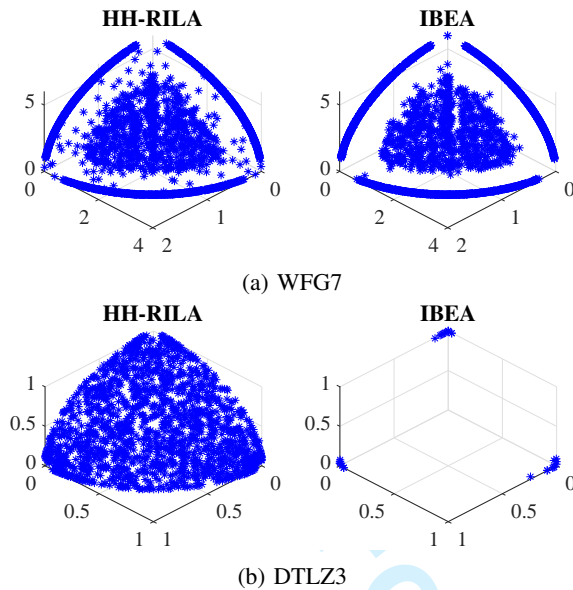
(a) WFG7

(b) DTLZ3

Fig. 5: Approximate PFs produced by HH-RILA (left column) and IBEA (right column) on WFG7 and DTLZ3.
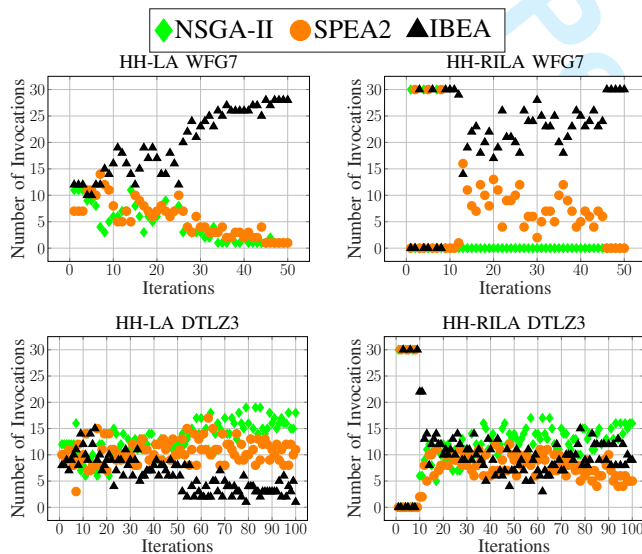

Fig. 6: The number of invocations of each MOEA at each iteration over 30 trials on WFG7 and DTLZ3 obtained from HH-LA (left column) and HH-RILA (right column).

4) An Analysis of Search Dynamics: To have a further understanding of the search progress induced by each low level MOEA at each trial under the proposed hyper-heuristics, we recorded which MOEA was chosen and applied at each iteration of 30 trials. Figure 6 provides plots based on that data as a representative of the search dynamics for HH-LA and HH-RILA on WFG7 and DTLZ3. Each tick on the plot indicates the total number of trials that the relevant MOEA is selected and invoked by the indicated hyper-heuristic at a given iteration. For example, HH-LA calls NSGA-II, 11 times, SPEA2, 7 times and IBEA, 12 times at the first iteration out of 30 trials when solving WFG7.

As observed from Figure 6, HH-LA slowly reduces the usage of poor performing MOEAs which are NSGA-II and SPEA2 for WFG7 during the exploration phase which corresponds to the first half of the search process (τ=0.5). Then the best performing MOEA; i.e., IBEA is predominately used for the remaining iterations. Meanwhile, HH-RILA excludes NSGA-II after the initial ranking stage (first nine iterations). Following the exploration phase, HH-RILA prefers invoking IBEA more and more while SPEA2 less and less as the search progresses. This behaviour is also reflected on the transition probability matrix (Figure 4). The transition probabilities from any MOEA to IBEA are higher than to the other MOEAs for both HH-LA and HH-RILA. Moreover, the transition probabilities to and from NSGA-II are all 0s for HH-RILA on WFG7.

On DTLZ3, HH-LA tends to prefer employing, firstly, NSGA-II and then SPEA2 more than IBEA after the exploration phase. Nevertheless, HH-RILA cannot detect the degrading performance of IBEA rapidly enough during the search, and this results in excluding SPEA2 or NSGA-II in some trials. Therefore, the performance of HH-RILA becomes worse than HH-LA with respect to hypervolume. This behaviour is also consistent with the mean utilisation rates as provided in Figure 3, which shows that HH-LA uses NSGA-II and SPAE2 more than HH-RILA does on DTLZ3.

E. Experimental Results for Vehicle Crashworthiness

HH-LA and HH-RILA are also tested on the real-world problem of VCP. The same experimental settings as in [14] are used for a fair performance comparison. The population size is fixed as 30 in these experiments. Each iteration consists of 50 generations. Each algorithm terminates whenever 75,000 solution evaluations are exceeded. The reference point for each VC instance is set to $r^i = z^{nadir_i} + 0.5(z^{nadir_i} - z^{ideal_i})$, where i is the index of an objective, z^{nadir_i} and z^{ideal_i} are the nadir (highest) and ideal (lowest) value of objective i, respectively. The performances of each algorithm with respect to hypervolume and $\epsilon+$ are provided in Table III. The results show that NSGA-II performs the best (with the highest hypervolume and lowest $\epsilon+$) on VC1 and VC2, while HH-RILA performs the best on VC3 and VC4 based on both performance indicators when compared to the other algorithms including each MOEA used on its own. The proposed hyper-heuristic exploits the synergy between MOEAs with different performances leading to an improved overall performance.

The one-tailed Wilcoxon rank-sum test comparing the performance difference of all pairs of multiobjective approaches are summarised in Table IV). The results suggest that both NSGA-II and SPEA2 perform significantly better than IBEA. HH-RILA outperforms HH-CF on 2 out of 4 VC problem instances (VC3 and VC4), and HH-RC on 3 out of 4 VC problem instances (VC2-4), and beats IBEA on all four VC problems. HH-LA performs similar as HH-CF, significantly better HH-RC on VC2, and also outperforms IBEA on all four VC problems. HH-RILA outperforms HH-LA on VC2-4.

F. Generality Analysis

Different approaches perform the best on different problems. The cross-domain search performance of algorithms indicating

TABLE III: Mean$_{STD}$ hypervolume and $\epsilon+$ values for the vehicle crashworthiness problems, averaged over 30 trials and their standard deviations provided as subscript entries. The best mean hypervolume and $\epsilon+$ on each problem instance is highlight in bold.

	Hypervolume				$\epsilon+$			
	VC1	VC2	VC3	VC4	VC1	VC2	VC3	VC4
HH-LA	$98.0693_{1.55}$	$53.0715_{2.35}$	$1.6553_{0.10}$	$2.0558_{0.07}$	$0.1354_{0.04}$	$0.0717_{0.12}$	$0.3565_{0.13}$	$0.0844_{0.07}$
HH-RILA	$98.8209_{1.94}$	$53.7729_{1.26}$	$\mathbf{1.7778_{0.11}}$	$\mathbf{2.0924_{0.03}}$	$0.1225_{0.05}$	$0.0374_{0.06}$	$\mathbf{0.1995_{0.15}}$	$\mathbf{0.0495_{0.03}}$
HH-CF	$98.9038_{1.82}$	$53.5628_{1.74}$	$1.6696_{0.11}$	$2.0582_{0.05}$	$0.1194_{0.05}$	$0.0501_{0.09}$	$0.3372_{0.14}$	$0.0794_{0.05}$
HH-RC	$98.2566_{1.75}$	$52.6460_{2.78}$	$1.7182_{0.13}$	$2.0457_{0.06}$	$0.1380_{0.04}$	$0.0982_{0.14}$	$0.2738_{0.16}$	$0.0897_{0.06}$
NSGA-II	$\mathbf{100.2163_{1.48}}$	$\mathbf{53.9514_{0.04}}$	$1.7093_{0.12}$	$2.0880_{0.04}$	$\mathbf{0.1007_{0.04}}$	$\mathbf{0.0362_{0.00}}$	$0.2864_{0.16}$	$0.0552_{0.04}$
SPEA2	$99.553_{2.02}$	$53.6523_{1.76}$	$1.7278_{0.12}$	$2.0669_{0.05}$	$0.1188_{0.05}$	$0.0486_{0.09}$	$0.2651_{0.16}$	$0.0818_{0.06}$
IBEA	$96.9435_{0.57}$	$52.1465_{3.06}$	$1.6846_{0.13}$	$2.0213_{0.06}$	$0.157_{0.02}$	$0.1184_{0.16}$	$0.3119_{0.16}$	$0.1148_{0.06}$

TABLE IV: One-tailed Wilcoxon rank-sum test at 5% significance level on VC problems with respect to hypervolume.'$>$' means the significantly better than, '$<$' significantly worse than, '$\sim$' no significant difference.

	VC1	VC2	VC3	VC4
HH-LA vs HH-CF	$\sim$	$\sim$	$\sim$	$\sim$
HH-LA vs HH-RC	$\sim$	$>$	$\sim$	$\sim$
HH-LA vs NSGA-II	$<$	$<$	$\sim$	$<$
HH-LA vs SPEA2	$\sim$	$\sim$	$\sim$	$\sim$
HH-LA vs IBEA	$>$	$>$	$\sim$	$>$
HH-RILA vs HH-CF	$\sim$	$\sim$	$>$	$>$
HH-RILA vs HH-RC	$\sim$	$>$	$>$	$>$
HH-RILA vs HH-LA	$\sim$	$>$	$>$	$>$
HH-RILA vs NSGA-II	$<$	$<$	$>$	$\sim$
HH-RILA vs SPEA2	$\sim$	$>$	$\sim$	$>$
HH-RILA vs IBEA	$>$	$>$	$>$	$>$
HH-CF vs HH-RC	$\sim$	$\sim$	$<$	$\sim$
NSGA-II vs SPEA2	$\sim$	$>$	$<$	$>$
NSGA-II vs IBEA	$>$	$>$	$>$	$>$
SPEA2 vs IBEA	$>$	$>$	$>$	$>$

which algorithm is more general and the best across a range of problems is often of interest [7]. As mentioned before, μ_{norm} is used in this work to assess the generality level of an algorithm across different problems including the benchmark functions and vehicle crashworthiness problems.

The empirical results based on μ_{norm} are presented in Table V. We also rank all the algorithms with respect to μ_{norm} values. The lower μ_{norm} value is, the better the rank of an algorithm. From Table V, we have the following observations. There is no single MOEA performing well across all three problem domains even though each uses the same perturbative operators. The performance of each individual MOEA varies vastly on different domains. IBEA, SPEA2 and NSGA-II deliver the best performance on WFG, DTLZ and VC, respectively. Although HH-RILA is always the second best performing approach on each domain, it has the best cross-domain performance and so the most general approach among all algorithms tested in this study with a mean μ_{norm} of 0.2089 based on the results from all 20 problem instances. Moreover, HH-LA turns out to be the second best approach with a mean μ_{norm} of 0.2571. The cross-domain performance of our selection hyper-heuristics are followed by IBEA, SPEA2, HH-RC, HH-CF and NSGA-II in that order. If we consider only WFG and DTLZ benchmarks ignoring the vehicle crashworthiness problem and compare the cross-domain performance of all algorithms, HH-RILA and HH-LA still rank the best and the second best based on the mean

TABLE V: Hypervolume μ_{norm} for each algorithm on each test problem instance. $\bar{x}_W$, $\bar{x}_D$, $\bar{x}_{VC}$, $\bar{x}_{WD}$, and $\bar{x}_{All}$ denote the mean μ_{norm} on WFG, DTLZ, VC, both WFG and DTLZ, as well as all three problem domains, respectively. The best mean μ_{norm} on each benchmark is highlight in bold, the second best is in gray box.

	NSGA-II	SPEA2	IBEA	HH-RC	HH-CF	HH-LA	HH-RILA
W1	0.8200	0.6761	0.2911	0.5304	0.7004	0.5091	0.6165
W2	0.4985	0.3881	0.5906	0.4486	0.3356	0.5653	0.2386
W3	0.4431	0.8086	0.1435	0.2564	0.4344	0.1206	0.2008
W4	0.8744	0.6233	0.0154	0.3545	0.5546	0.0971	0.0614
W5	0.7905	0.5776	0.0148	0.3897	0.5819	0.0630	0.0653
W6	0.8021	0.5591	0.1126	0.4029	0.5648	0.1318	0.1519
W7	0.7047	0.5185	0.0066	0.3662	0.4684	0.0639	0.0337
W8	0.8353	0.5647	0.0295	0.5101	0.6688	0.1002	0.1354
W9	0.7689	0.7186	0.2357	0.6273	0.6421	0.4141	0.4357
$\bar{x}_W$	0.7264	0.6038	**0.1600**	0.4318	0.5501	0.2294	0.2155
D1	0.0411	0.0014	0.7981	0.2021	0.2160	0.0354	0.0335
D2	0.7336	0.2522	0.0114	0.3425	0.4315	0.0150	0.0107
D3	0.1000	0.0144	0.9974	0.4164	0.3687	0.2753	0.4280
D4	0.1327	0.2266	0.1859	0.1092	0.1465	0.3186	0.1669
D5	0.6549	0.1900	0.6945	0.4757	0.3996	0.2622	0.4605
D6	0.1654	0.0729	0.4437	0.1606	0.2082	0.2882	0.1654
D7	0.0862	0.0548	0.4093	0.1866	0.0550	0.1867	0.0339
$\bar{x}_D$	0.2734	**0.1160**	0.5057	0.2705	0.2608	0.1974	0.1855
VC1	0.2715	0.3746	0.7804	0.5762	0.4756	0.6054	0.4885
VC2	0.0288	0.0711	0.2839	0.2133	0.0837	0.1531	0.0540
VC3	0.4961	0.4402	0.5708	0.4693	0.6164	0.6596	0.2887
VC4	0.1291	0.2265	0.4378	0.3247	0.2672	0.2782	0.1088
$\bar{x}_{VC}$	**0.2314**	0.2781	0.5182	0.3959	0.3607	0.4241	0.2350
$\bar{x}_{WD}$	0.5282	0.3904	0.3113	0.3612	0.4235	0.2154	**0.2024**
$\bar{x}_{All}$	0.4688	0.3680	0.3526	0.3682	0.4110	0.2571	**0.2089**
Rank	7	4	3	5	6	2	1

μ_{norm} values averaged over all benchmark functions, yielding 0.2024 and 0.2154, respectively.

V. CONCLUSION AND FUTURE WORK

In this study, we proposed two variants of a learning automata based multiobjective selection hyper-heuristic: HH-LA and HH-RILA. The performance and generality of HH-LA and HH-RILA controlling three MOEAs are investigated on three multiobjective continuous optimisation problem domains, including two sets of benchmark functions (WFG and DTLZ) and the real-world problem of vehicle crashworthiness. The experimental results show that the different MOEAs perform the best on different problem domains. However, HH-RILA and HH-LA perform the best and second best, respectively, across all test problem instances based on the μ_{norm} indicator.

In addition, both HH-LA and HH-RILA outperform a state-of-the-art online learning hyper-heuristic and a random choice hyper-heuristic. The results also suggest that the proposed hyper-heuristics are indeed capable of exploiting the strengths of the low level MOEAs delivering an improved performance via a learning automata in this work. The proposed component for metaheuristic (action) selection, that is the ϵ-RouletteGreedy method adaptively balances exploration and exploitation of the use of MOEAs by combining the merits of two regular selection methods of roulette wheel and greedy selection.

In future work, both HH-LA and HH-RILA will be applied to other discrete and continuous real-world problems and their performance as well as the level of generality will be assessed. In single objective optimisation, it has been observed that the performance of selection hyper-heuristics might vary depending on the set of low level heuristics [25], hence another interesting research direction would be investigating the performance of the proposed hyper-heuristics using different sets of low level multiobjective metaheuristics. Moreover, there is a growing interest into many-objective (problems with more than three objectives) optimisation and recent studies show that many-objective optimisation requires different approaches from multiobjective metaheuristics [75]. The tests that we performed in this study is somewhat at the boundary of multi and many-objective optimisation considering that three objective problems are used in the experiments. Therefore, the low level metaheuristics used in the proposed multiobjective selection hyper-heuristic framework can be replaced by other state-of-the-art many-objective metaheuristics (such as, NSGA-III [76] and AGE [77]), and tested on a range of many-objective optimisation problems. Although hypervolume is utilised in this study as the main indicator for guiding the search and combining strengths of different MOEAs for multiobjective optimisation, the proposed framework allows the use of different indicators. It would be worth investigating the behaviour of the proposed selection hyper-heuristics when other convergence/performance indicators are used for guidance, such as the $\epsilon+$ indicator.

REFERENCES

[1] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2002, vol. 242.

[2] M. Saadatseresht, A. Mansourian, and M. Taleai, "Evacuation planning using multiobjective evolutionary optimization approach," *European Journal of Operational Research*, vol. 198, no. 1, pp. 305–314, 2009.

[3] T. Hanne and S. Nickel, "A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects," *European Journal of Operational Research*, vol. 167, no. 3, pp. 663–678, 2005.

[4] B. Alatas, E. Akin, and A. Karci, "Modenar: Multi-objective differential evolution algorithm for mining numeric association rules," *Applied Soft Computing*, vol. 8, no. 1, pp. 646–656, 2008.

[5] E. Masazade, R. Rajagopalan, P. K. Varshney, C. K. Mohan, G. K. Sendur, and M. Keskinoz, "A multiobjective optimization approach to obtain decision thresholds for distributed detection in wireless sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 2, pp. 444–457, 2010.

[6] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[7] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *J Oper Res Soc*, vol. 64, no. 12, pp. 1695–1724, Dec 2013. [Online]. Available: http://dx.doi.org/10.1057/jors.2013.71

[8] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of metaheuristics*. Springer, 2010, pp. 449–468.

[9] S. Asta, E. Özcan, and T. Curtois, "A tensor based hyper-heuristic for nurse rostering," *Knowledge-Based Systems*, vol. 98, pp. 185–199, 2016.

[10] S. Asta and E. Özcan, "A tensor-based selection hyper-heuristic for cross-domain heuristic search," *Information Sciences*, vol. 299, pp. 412–432, 2015.

[11] T. Wauters, K. Verbeeck, P. De Causmaecker, and G. V. Berghe, "Boosting metaheuristic search using reinforcement learning." *Hybrid Metaheuristics*, vol. 434, pp. 433–452, 2013.

[12] A. Kheiri and E. Özcan, "An iterated multi-stage selection hyper-heuristic," *European Journal of Operational Research*, vol. 250, no. 1, pp. 77–90, 2016.

[13] K. Z. Zamli, F. Din, G. Kendall, and B. S. Ahmed, "An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation," *Information Sciences*, vol. 399, pp. 121–153, 2017.

[14] M. Maashi, E. Özcan, and G. Kendall, "A multi-objective hyper-heuristic based on choice function," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4475–4493, 2014.

[15] R. A. Gonçalves, J. N. Kuk, C. P. Almeida, and S. M. Venske, "Moea/d-hh: a hyper-heuristic for multi-objective problems," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 94–108.

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

[17] E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[18] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 832–842.

[19] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, 2003.

[20] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 5, pp. 477–506, 2006.

[21] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable test problems for evolutionary multiobjective optimization*. Springer, 2005.

[22] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion," in *2006 IEEE International Conference on Evolutionary Computation*. IEEE, 2006, pp. 892–899.

[23] L. Di Gaspero and T. Urli, "Evaluation of a family of reinforcement learning cross-domain optimization heuristics," in *Learning and Intelligent Optimization*. Springer, 2012, pp. 384–389.

[24] S. Adriaensen, G. Ochoa, and A. Nowé, "A benchmark set extension and comparative study for the hyflex framework," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 784–791.

[25] B. Bilgin, E. Özcan, and E. E. Korkmaz, "An experimental study on hyper-heuristics and exam timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2006, pp. 394–412.

[26] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[27] N. Hitomi and D. Selva, "A hyperheuristic approach to leveraging domain knowledge in multi-objective evolutionary algorithms," in *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2016, pp. V02BT03A030–V02BT03A030.

[28] D. Thierens, "Adaptive strategies for operator allocation," in *Parameter Setting in Evolutionary Algorithms*. Springer, 2007, pp. 77–90.

[29] J. Vazquez-Rodriguez and S. Petrovic, "A mixture experiments multi-objective hyper-heuristic," *Journal of the Operational Research Society*, vol. 64, no. 11, pp. 1664–1675, 2013.

[30] A. C. Kumari and K. Srinivas, "Scheduling and inspection planning in software development projects using multi-objective hyper-heuristic

evolutionary algorithm," *International Journal of Software Engineering & Applications*, vol. 4, no. 3, p. 45, 2013.

[31] A. C. Kumari and K. Srinivas, "Hyper-heuristic approach for multi-objective software module clustering," *Journal of Systems and Software*, vol. 117, pp. 384–401, 2016.

[32] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in *Metaheuristics: Computer decision-making*. Springer, 2004, pp. 523–544.

[33] G. Guizzo, G. M. Fritsche, S. R. Vergilio, and A. T. R. Pozo, "A hyper-heuristic for the multi-objective integration and test order problem," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1343–1350.

[34] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, "Analyzing bandit-based adaptive operator selection mechanisms," *Annals of Mathematics and Artificial Intelligence*, vol. 60, no. 1-2, pp. 25–64, 2010.

[35] W. K. G. Assunção, T. E. Colanzi, S. R. Vergilio, and A. Pozo, "A multi-objective optimization approach for the integration and test order problem," *Information Sciences*, vol. 267, pp. 119–139, 2014.

[36] A. Masood, Y. Mei, G. Chen, and M. Zhang, "Many-objective genetic programming for job-shop scheduling," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 209–216.

[37] A. Masood, Y. Mei, G. Chen, and M. Zhang, "A pso-based reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling," in *Australasian Conference on Artificial Life and Computational Intelligence (ACALCI)*, Melbourne, Australia, 2017.

[38] D. Karunakaran, G. Chen, and M. Zhang, "Parallel multi-objective job shop scheduling using genetic programming," in *Australasian Conference on Artificial Life and Computational Intelligence*. Springer, 2016, pp. 234–245.

[39] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 193–208, 2014.

[40] M. Freitag and T. Hildebrandt, "Automatic design of scheduling rules for complex manufacturing systems by multi-objective simulation-based optimization," *CIRP Annals-Manufacturing Technology*, 2016.

[41] C. Ryan, J. Collins, and M. O. Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *European Conference on Genetic Programming*. Springer, 1998, pp. 83–96.

[42] T. Mariani, G. Guizzo, S. R. Vergilio, and A. T. Pozo, "Grammatical evolution for the multi-objective integration and test order problem," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*. ACM, 2016, pp. 1069–1076.

[43] M. P. Basgalupp, R. C. Barros, and V. Podgorelec, "Evolving decision-tree induction algorithms with a multi-objective hyper-heuristic," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 110–117.

[44] J. A. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," *Proceedings of the National Academy of Sciences*, vol. 104, no. 3, pp. 708–711, 2007.

[45] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, 1998.

[46] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons," *Artificial intelligence review*, vol. 17, no. 4, pp. 251–290, 2002.

[47] M. Maashi, G. Kendall, and E. Özcan, "Choice function based hyper-heuristics for multi-objective optimization," *Applied Soft Computing*, vol. 28, pp. 312–326, 2015.

[48] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe, "A new hyper-heuristic implementation in hyflex: a study on generality," in *Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory & Application*, 2011, pp. 374–393.

[49] M. Tsetlin, "On behaviour of finite automata in random medium," *Avtomat. i Telemekh*, vol. 22, no. 10, pp. 1345–1354, 1961.

[50] M. Thathachar and P. Sastry, *Adaptive stochastic algorithms for pattern classification*. Singapore: World Scientific, 2001.

[51] T. I. Ahamed, P. N. Rao, and P. Sastry, "A reinforcement learning approach to automatic generation control," *Electric power systems research*, vol. 63, no. 1, pp. 9–26, 2002.

[52] K. S. Narendra and M. A. Thathachar, *Learning automata: an introduction*. Courier Corporation, 2012.

[53] B. J. Oommen and E. d. S. Croix, "Graph partitioning using learning automata," *IEEE Transactions on Computers*, vol. 45, no. 2, pp. 195–208, 1996.

[54] P. Du Bois, C. C. Chou, B. B. Fileta, T. B. Khalil, A. I. King, H. F. Mahmood, H. J. Mertz, J. Wismans, P. Prasad, and J. E. Belwafa, "Vehicle crashworthiness and occupant protection," 2004.

[55] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Structural and Multidisciplinary Optimization*, vol. 35, no. 6, pp. 561–569, 2008.

[56] R. Yang, N. Wang, C. Tho, J. Bobineau, and B. Wang, "Metamodeling development for vehicle frontal impact simulation," *Journal of Mechanical Design*, vol. 127, no. 5, pp. 1014–1020, 2005.

[57] A. Kheiri and E. Keedwell, "A sequence-based selection hyper-heuristic utilising a hidden markov model," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 417–424.

[58] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Practice and theory of automated timetabling III*. Springer, 2000, pp. 176–190.

[59] J. H. Drake, E. Özcan, and E. K. Burke, "A modified choice function hyper-heuristic controlling unary and binary operators," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 3389–3396.

[60] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *Tik report*, vol. 214, pp. 327–332, 2006.

[61] K. Bringmann and T. Friedrich, "The maximum hypervolume set yields near-optimal approximation," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 511–518.

[62] K. Bringmann and T. Friedrich, "Tight bounds for the approximation ratio of the hypervolume indicator," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2010, pp. 607–616.

[63] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[64] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[65] D. B. Gümüş, E. Ozcan, and J. Atkin, "An investigation of tuning a memetic algorithm for cross-domain search," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 135–142.

[66] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, "Taguchi-based parameter designing of genetic algorithm for artificial neural network training," in *Informatics and Creative Multimedia (ICICM), 2013 International Conference on*. IEEE, 2013, pp. 278–281.

[67] R. K. Roy, *A primer on the Taguchi method*. Society of Manufacturing Engineers, 2010.

[68] M. Wagner, K. Bringmann, T. Friedrich, and F. Neumann, "Efficient optimization of many objectives by approximation-guided evolution," *European Journal of Operational Research*, vol. 243, no. 2, pp. 465–479, 2015.

[69] T. Tušar and B. Filipič, "Differential evolution versus genetic algorithms in multiobjective optimization," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 257–271.

[70] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 3, pp. 1–15, 1994.

[71] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[72] L.-Y. Deng, "Orthogonal arrays: Theory and applications," *Technometrics*, vol. 42, no. 4, pp. 440–440, 2000.

[73] N. P. Suh, *The principles of design*. Oxford University Press on Demand, 1990, no. 6.

[74] W. Li, E. Özcan, R. John, J. H. Drake, A. Neumann, and M. Wagner, "A modified indicator-based evolutionary algorithm (mibea)," 2017.

[75] S. Bechikh, M. Elarbi, and L. B. Said, "Many-objective optimization using evolutionary algorithms: A survey," in *Recent Advances in Evolutionary Multi-objective Optimization*. Springer, 2017, pp. 105–137.

[76] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[77] M. Wagner, K. Bringmann, T. Friedrich, and F. Neumann, "Efficient optimization of many objectives by approximation-guided evolution," *European Journal of Operational Research*, vol. 243, no. 2, pp. 465 – 479, 2015.