A Choice Function based hyper-heuristic for Multi-objective Optimization

Mashael Maashi, Student Member, IEEE, Graham Kendall, Senior Member, IEEE and Ender Özcan, Member, IEEE

Abstract—Hyper-heuristics are emerging methodologies that perform a search over the space of heuristics to solve difficult computational optimization problems. There are two main types of hyper-heuristics: selective and generative hyper-heuristics. An online selective hyper-heuristic framework manages a set of low level heuristics and aims to choose the best one at any given time using a performance measure for each low level heuristic. In this study, we propose a selective hyper-heuristic choice function based to solve multi-objective optimization problems. This hyper-heuristic controls and combines the strengths of three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA) as the low level heuristics. The choice function heuristic selection method uses a scheme which ranks four performance measurements. All-Moves is employed as an acceptance strategy, meaning that we accept the output of each low level heuristic whether it improves the quality of the solution or not. Our proposed approach compared to the three low level heuristics when used in isolations and two multi-objective hyper-heuristics; a random hyper-heuristics and an adaptive multi-method search namely (AMALGAM). The experimental results demonstrate the effectiveness of the hyper-heuristic approach over the WFG test suite, a common benchmark for multi-objective optimization.

Index Terms—Hyper-heuristics, multi-objective evolutionary algorithms, multi-objective optimization

I. INTRODUCTION

Hyper-heuristics have drawn increasing attention from the research community in recent years. However, the majority of research in this area has been limited to single objective optimization. Hyper-heuristics are methodologies primarily used to generate high quality solutions to optimization problems by performing a search over the space of heuristics rather than searching the solution space directly. One of their aims is to raise the level of generality at which search methodologies operate. Many hyper-heuristic papers have been published, but they have been mainly limited to single objective optimization [1].

Hyper-heuristics for multi-objective optimization problems is a new area of research in Evolutionary Computation and Operational Research [1], [2]. To date, few studies, have been identified that deal with hyper-heuristics for multi-objective problems. The first approach [3] is a multi-objective hyper-heuristic based on tabu search (TSRoulette Wheel). The key feature of this paper lies in choosing a suitable heuristic at each iteration to tackle the problem at hand by using tabu search as a high-level search strategy. The proposed approach was applied to space allocation and timetabling problems and produced results with acceptable solution quality. Another approach [4] comprises two phases: the first phase aims to produce an efficient Pareto front (this may be of low quality based on the density), while the second phase aims to deal with a given problem in a flexible way to drive a subset of the population to the desired Pareto front. This approach was evaluated on the multi-objective traveling salesman problems with eleven low level heuristics. It is compared to other multi-objective approaches from the literature which reveals that the proposed approach generates good quality results but future work is still needed to improve the methodology. In [5] an online selective hyper-heuristic, Markov chain based, (MCHH) is investigated. The Markov chains guide the selection of heuristics and applies online reinforcement leaning to adapt transition weights between heuristics. In MCHH, hybrid meta-heuristics; Evolution Strategies was incorporated and applied to the DTLZ test problems [6] and compared to a (1+1) Evolution Strategy meta-heuristic, a random hyper-heuristic and TSRoulette Wheel [3]. The comparison shows the efficacy of the proposed approach in terms of Pareto convergence and learning ability to select good heuristics combinations. Further work is needed in terms of diversity preserving mechanisms. The MCHH was applied to the WFG test problems [7], the experiments shows efficacy of the method but future work is still needed in terms of acceptance strategies to improve the search [5]. The MCHH has also been applied to real-world water distribution networks design problems and produced competitive results [5]. A new hyper-heuristic based on the multi-objective evolutionary algorithm NSGAII [8] is proposed in [9]. The main idea of this method is in producing the final Pareto-optimal set, through a learning process that evolves combinations of condition-action rules based on NSGAII. The proposed method was tested on many instance of irregular 2D cutting stock benchmark

This work was supported in part by University of Tabuk and Ministry of Higher Education in Saudi Arabia.

M. Maashi, G. Kendall and E. Özcan are with the University of Nottingham, Nottingham, UK (e-mail: psxm3@email.nottingham.ac.uk, Graham.Kendall@nottingham.my, Ender.Ozcan@nottingham.ac.uk). Manuscript received month xx, 20xx.
problems and produced promising results. In [10], [11], a hyper-heuristic-based codification is proposed for solving strip packing and cutting stock problems with two objectives that maximize the total profit and minimize the total number of cuts. Experimental results show that the proposed hyper-heuristic outperforms the other single heuristics. A adaptive multi-method search called AMALGAM is proposed in [12]. It employs multiple search algorithms; (NSGAII [8], PSO [13], AMS [14], and DE [15]) simultaneously using the concepts of multi-method search and adaptive offspring creation. AMAMLGAM is applied to set of well known multi-objective test problems and it was superior to other methods [12]. It also applied to solve a number of water resource problems and it yielded very good solutions [16],[17]. A multi-strategy ensemble multi-objective evolutionary algorithm called MS-MOEA for dynamic optimization is proposed in [17]. It is combining different strategies including a memory strategy, a genetic and differential operators to adaptively create offspring and achieve fast convergence speed. Experimental result shows that MS-MOEA is able to obtain promising results. None of the above studies have used multi-objective evolutionary algorithms (MOEAs), expect in [9] and [12] and no standard multi-objective test problems studied, expect in [5] and [12].

In this paper, we propose an online selective hyper-heuristic choice function based for multi-objective optimization (HHMO_CF) that controls and combines the strengths of three well-known multi-objective evolutionary algorithms (NSGAII [8], SPEA2 [18], and MOGA [19]), which are utilized as the low level heuristics. A choice function acts as the high level strategy which adaptively scores the performance of three low level heuristics, during the search process. In this paper we will refer to these algorithms as low level heuristics and to the proposed HHMO_CF approach as HH_CF for short. Our multi-objectives hyper-heuristics framework addresses three main research areas, these being: multi-objective evolutionary algorithms, hyper-heuristics, and multi-objective test problems. In Section II we will briefly discuss each of these areas. The rest of the paper is organized as follows. In Section III, the details of a choice function based hyper-heuristic framework for multi-objective optimization is described. The comparison of HH_CF and the low level heuristics are presented in Section IV. The comparison of HH_CF and other multi-objective hyper-heuristics is presented in Section V. Finally, conclusions are provided in Section VI.

II. BACKGROUND

A. Multi-objective Optimization

A multi-objective problem (MOP) comprises several objectives (two or more), which need to be minimized or maximized depending on the problem. Each objective has some measure as to the quality of the solution. Some objectives cannot be measured by the same scalar unit. These criteria are often described as incommensurable criteria [20]. In addition, some objectives maybe, and usually are, in conflict with each other, i.e. an improvement in one objective value leads to deterioration in the values of at least one of the other objectives. The conflict in objectives can be totally-conflicting, non-conflicting or partially-conflicting. According to [21] if the objective functions are totally-conflicting, there is no need for an optimization process, while if the objectives are non-conflicting the objectives can be converted into a single objective and tackled as a single optimization problem. However, if the objectives are partially-conflicting multi-objective optimization techniques are required to solve these cases. In the scientific literature, many common techniques are presented for multi-objective optimization. An example is a posteriori technique. In this technique, the search is conducted to find solutions for the objective functions. Following this, a decision process selects the most appropriate solutions (often involving a trade off). Examples of this technique are multi-objective evolutionary optimization (MOEA) methods, whether non Pareto-based or Pareto-based methods. The Pareto-based evaluation is a method used to evaluate the quality of MOP solutions. In Pareto-based methods, all objectives are optimized simultaneously applying Pareto dominance concepts. The idea behind the dominance concept is to generate a preference between MOP solutions since there is no information regarding the objective preference provided by the decision maker. A more formal definition of Pareto dominance, from [21],[22].

Definition 1:

A vector \( u = (u_1, ..., u_k) \) is said to dominate another vector \( v = (v_1, ..., v_k) \) (denoted by \( u \preceq v \)) according to \( k \) objectives if and only if \( u \) is partially less than \( v \), i.e.,

\[
\forall \ i \in \{1, ..., k\}, \ u_i \leq v_i \land \exists \ i \in \{1, ..., k\}: u_i < v_i.
\]

In the literature, various features for multi-objective optimization test problems are presented. Those features are designed to make the problems difficult enough to examine algorithmic performance. Examples of these features are deception [23], [24], multimodality [25], noise [26], and epistasis [27]. Moreover, other features of test problems are suggested in [28], such as multimodality, deceptive, isolated optimum and collateral noise. These features can cause difficulties for evolutionary optimizers in terms of converging to the Pareto optimal front (POF) and maintaining the population diversity. Furthermore, some characteristics of the POF such as convexity or nonconvexity, discreteness, and nonuniformity could cause difficulties in term of the population diversity [29]. Branke in [30] asserted that the test problems should be simple and straightforward in order to
understand the behavior of the optimization algorithm more easily. In addition, they should be describable and analyzable, and their parameters should be tunable. Nevertheless, they should be complicated enough to provide a true reflection of real world problems. The main features of test problems for multi-objective optimization presented in [6] including the simplicity of formation, scalability to any number of decision variables, scalability to any number of objectives, accurate and specific knowledge of the shape and location of the Pareto fronts, finding a widely distributed set of Pareto solutions, and the capability to overcome the difficulty in converging to the true Pareto front. Furthermore, Huband et al. in [7] introduced the following key features of multi-objective test problems which present varying degrees of problem difficulty for the multi-objective optimizers:

- POF Geometry such as convex, linear, concave, mixed, degenerate, and disconnected.
- Parameter Dependencies which refer to whether the problem objective is separable or nonseparable.
- Bias feature which refers to whether the test problem may or may not be biased.
- Many-to-One Mappings which refer to the fitness landscape, which are either one-to-one or many-to-one.
- Modality feature which refers to the problem objective; this may be unimodal or multimodal (can be a deceptive multimodality).

In [7] some useful recommendations for designing multi-objective test problems are also made including:

- No extremal parameters to the test problem in order to prevent exploitation by truncation operators.
- No medial parameters for the test problem in order to prevent exploitation by intermediate recombination.
- Scalability in the number of decision variables.
- Scalability in the number of objectives.
- The parameters of the test problem should have domains of dissimilar magnitude to encourage an optimizer to scale the strengths of the mutation operator.
- Knowledge of the POF in order to support the analysis of the results.

B. Multi-objective Evolutionary Algorithms

Many EA researchers would argue that evolutionary algorithm(s) are more suitable methods to deal with multi-objective optimization problems [22], [31], [32], [33], [34], [19], [35], [11] because of their population-based nature, which means they can find Pareto-optimal sets (trade-off solutions) in a single run which allow a decision maker to select a suitable compromise solution.

This section will focus on the Pareto-based approaches particularly MOGA [19], NSGAII [8], and SPEA2 [18], because they incorporate the MOEA theory and their general algorithm complexity is less than other MOEAs [36].

1) Multi-objective Genetic Algorithm (MOGA): It was proposed by Fonseca and Fleming in [37]. The Pareto ranking scheme is used i.e. each solution in the current population is given a rank based on their dominance rank [36], [38]. All solutions in the Pareto optimal set have a rank of 1. A niche-formation method (fitness sharing) is employed in phenotypic-based cases to maintain a well-distributed population over the POF [22]. The average value of the fitness for all solutions that have the same rank is assigned to these solutions [38]. A modified version of this algorithm has been proposed in [19]. This version employed restricted sharing between solutions that have the same rank and distance between two solutions is computed and compared to the key sharing parameter \( \sigma_{share} \). While MOGA is efficient and easy to implement, its fitness sharing method prevents two vectors that have the same value in the objective space existing simultaneously unless the fitness sharing is genotypic-based.

2) Non-dominated Sorting Genetic Algorithm II (NSGAIi): The original version of the Non-dominated Sorting Genetic Algorithm (NSGA) was proposed in [36]. It employs a dominance depth based on the Pareto ranking scheme [36]. Moreover, a dummy fitness value, proportional to the population size, is used to classify all solutions in the Pareto optimal set. The fitness sharing method is quite similar to that used in MOGA but it is genotypic-based and applied to each level to maintain the diversity of the population and to obtain a uniform distribution of the Pareto optimal front [29]. Once all solutions in the population are classified, the first Pareto front is assigned to the maximum fitness value. Therefore, the first Pareto front must have more copies than the other solutions in the population. A stochastic remainder of the proportionate selection strategy is employed for this purpose [22]. The complexity of NSGA is exhibited in its fitness sharing mechanism which assigns the fitness values to solutions in the current population [39], and many other researchers, have reported that NSGA has a poorer performance than MOGA. It is also more sensitive to the sharing parameter \( \sigma_{share} \) than MOGA. However, some researchers point out that NGSA helps to obtain a well-spread POF [22].

A modified version of NSGA was proposed in [8]. The modified version (NSGAII) is a non-explicit building block MOEA technique that incorporates the concept of elitism [22], [40]. The solutions compete, then each solution is ranked and sorted based on its Pareto-optimal level. Genetic operators are applied to generate a new group of children who are then merged with parents in the population [22]. Furthermore, a niching method based on crowding distance is used during the selection process in order to maintain a diverse Pareto front [35]. Although NSGAII is more efficient than NSGA, it still has some drawbacks. It
cannot simply generate a Pareto optimal set in some regions in the search space, particularly unpopulated regions [41]. In addition, its search bias strongly appears as the number of objectives rises [42]. In other words, the algorithm seems to have bias towards some regions in the search space.

3) Strength Pareto Evolutionary Algorithm 2 (SPEA2): The first version of Strength Pareto Evolutionary Algorithm was proposed in [43]. It integrates different desirable features in MOEAs which are (i) the use of the concept of dominance in the evaluation and selection process. (ii) The use of an external archive (secondary population) of the Pareto optimal set that was previously obtained. (iii) The use of clustering and niching methods [20]. In each generation, the Pareto-optimal set is added to the secondary population. The solutions in the secondary population are used to evaluate the fitness values for the solution in the current population by summing the solutions’ rank in the secondary population [20], [36]. The Pareto ranking scheme, based on the dominance count and rank, is employed, which means any distance measurement such as niche radius is not required [22]. The secondary population participates in the selection process, which leads to an increase in the population size. Therefore, a clustering technique, namely the average linkage method, is adopted to deal with this issue [22].

Despite SPEA generally having a good performance, it has some potential weak points in terms of fitness assignment, density estimation and archive truncation, which may affect SPEA’s quality [44]. To overcome these, an updated version called SPEA2 was proposed in [18]. SPEA2 differs from the previous version in three aspects: (i) It incorporates a fine-grained fitness assignment strategy which considers the number of individuals for each solution that dominates it and which it is dominated by. (ii) It uses a nearest neighbor density estimation technique in order to increase the efficiency of the search. (iii) It improves the archive truncation method that guarantees the preservation of boundary points by replacing the average linkage method used in the previous version. Experimental results show that SPEA2 performs well in terms of diversity and distribution as the number of objectives increases. In addition, it significantly outperforms its predecessor SPEA.

4) MOEAs Comparison: Generally, most MOEAs have common strategies that are employed in their search process. However, they are different in the way that they apply these strategies. MOGA classifies the solutions based on the ranking scheme using linear or exponential interpolation and applies the sharing scheme in the objective space, while NSGA uses dummy fitness values assigned to the solutions and applies the sharing scheme in the decision variable space [36] and an NSGA with elitism performs as well as a SPEA [29].

In the literature, some studies have compared MOEAs’ performance and quality against each other. A comparison study for SPEA2, NSGAII and MOGA on ZDT4 and ZDT6 problems [29] was presented in [45]. With respect to the ratio of non-dominated individual metric (RNI), NSGAII has better performance than the others on ZDT4. However, SPEA2 outperforms MOGA and NSGAII for the same metric on ZDT6. The authors concluded this study by stating that SPEA2 has advantage in the point of the accuracy than NSGAII. While NSGAII is superior to SPEA2 in finding wide spread solutions. In [46], another a comparative study for NSGAII, SPEA2 and PEAS on four test problems (DTLZ1, DTLZ2, DTLZ3 and DTLZ6) [47] with 2-8 objectives was carried out. Three performance metrics were used for convergence and diversity of the obtained non-dominated set and the running time. SPEA2 performs better than NSGAII in terms of convergence for a small number of objectives. However, both perform similarly for a higher number of objectives. SPEA2 and NSGAII have good performance with respect to the diversity, but they have some difficulties in the closeness of the obtained non-dominated set to the POF. In comparison, PEAS [48] performs very well in converging to the true front but it fails in diversity and running time. However, NSGAII performs better than the others with respect to running time. Another comparative study between NSGAII and SPEA2 on the WFG test problems [7] with 24 real values and a different scale of objectives was presented in [49]. For two objectives, NSGAII bests SPEA2 on the WFG test problems with respect to the epsilon metric and the hypervolume (SSC). In contrast, SPEA2 outperforms NSGAII on all WFG problems except WFG3 in three objectives with respect to the same two metrics.

So, we can note from two last studies that the number of objectives can affect the performance of an algorithm. SPEA2 works well with a high number of objectives for WFG and a low number of objective for DTLZ. The opposite is true for NSGAII. We can observe also from these comparative studies, an algorithm can perform better than other algorithms with respect to a specific metric in a certain problem, while another algorithm performs better than that particular algorithm with respect to another metric for the same problems. Also an algorithm can be performing differentially according to the number of objectives. All these observations could be an advantage to combine different algorithms in hyper-heuristic framework for multi-objective optimization to get benefit from the strengths of the algorithms and avoid their weaknesses. These observations also can be empirical evidence to the No Free Lunch Theorem [50].

C. Hyper-heuristics

Most real-world problems are complex. Due to their (often) NP-hard nature, researchers and practitioners
frequently resort to problem tailored heuristics to obtain a reasonable solution in a reasonable amount of time. Generally, there are two recognized types of heuristics: (i) constructive heuristics which process a partial solution (or solutions) and build a complete solution (or solutions), (ii) perturbative heuristics which operate on complete solution(s) [1].

Hyper-heuristics are methodologies that operate on a search space of heuristics for solving hard computational problems, with one of the key aims being to raise the level of generality. Hyper-heuristics have a strong link to Operations Research in terms of finding optimal or near-optimal solutions to computational search problems. It is also firmly linked to a branch of Artificial Intelligence in terms of machine learning methodologies [51]. In a hyper-heuristic approach, different heuristics (or heuristic components) can be selected, generated or combined to solve a given optimization problem in an efficient way.

Usually, in a hyper-heuristic framework, there is a clear separation between the high level hyper-heuristic approach (also referred to as strategy) and the set of low-level heuristics or heuristic components. It is assumed that there is a domain barrier between them [52]. The purpose of domain barrier is giving the hyper-heuristics a higher level of abstraction. This is also increase the level of generality of hyper-heuristics by applying a new of problem without changing of the framework only a set of problem-related heuristics are supplied. The barrier allows only problem domain independent information to flow from the low level to the high level, such as the fitness/cost/penalty value (measured by an evaluation function, indicating the quality of a solution) [53]. Low level heuristics or heuristic components are the problem domain specific elements of a hyper-heuristic framework; hence they have access to any relevant information, such as candidate solution(s). The task of the high level strategy is to guide the search intelligently and adapt according to the success/failure of the low level heuristics or combinations of heuristic components during the search process, in order to enable the reuse of the same approach for solving different problems [54]. Thus, the high level strategy does not change while both the low-level heuristics or heuristic components and the evaluation function require changing when tackling a new problem. The high level strategy can be a (meta-)heuristic or a learning mechanism [3].

1) Classification of Hyper-heuristics: Two types of hyper-heuristic methodologies can be identified in the literature [1]: (i) heuristic selection methodologies: (meta-)heuristics to choose (meta-)heuristics, and (ii) heuristic generation methodologies: (meta-)heuristics to generate new (meta-)heuristics from given components. The former type of selection hyper-heuristics based on perturbative meta-heuristics is the focus of this study. More on generation hyper-heuristics can be found in [1], [55], [56]. An orthogonal classification of hyper-heuristics is provided in [51] depending on: (i) the nature of the heuristic search space and (ii) the source of feedback during the search process. Hyper-heuristics can be used to select or generate constructive or perturbative heuristics which determine the nature of the heuristic search space. However, a new research direction of hybrid hyper-heuristics might include a combination of heuristic selection and heuristic generation methodologies, or a combination of constructive and perturbative heuristics. A hyper-heuristic can employ no learning, online learning (getting feedback from the search process while solving an instance), or offline learning (getting feedback via training over a selected set of instances to be utilized for solving unseen instances). A hyper-heuristic which combines simple random heuristic selection with a method of accepting improving and equal quality moves is an example which uses a no learning approach [2]. If a hyper-heuristic incorporates a mechanism to adaptively guide the search process and enable the approach to make informed decisions about selecting or generating a low level heuristic, then it is a learning hyper-heuristic. Machine learning techniques are commonly used in hyper-heuristics. For example, reinforcement learning (based on reward/punishment) is employed as an online learning method for heuristic selection in hyper-heuristics [57]. Genetic programming is frequently used as an offline learning hyper-heuristic which learns via the evolutionary process [56].

2) Selective Hyper-heuristics: Selective hyper-heuristics perform a search using two successive stages [1], [2]: (meta-)heuristic selection and acceptance. An initial solution (a set of initial solutions) is iteratively improved using the low level (meta-)heuristics until some termination criteria is satisfied. During each iteration, the (meta-)heuristic selection decides which low level (meta-)heuristic will be employed next based on some criteria (or randomly). After the selected (meta-)heuristic is applied to the current solution (a set of solutions), a decision is made whether to accept the new solution(s) or not using an acceptance method. The low level (meta-)heuristics in a selective hyper-heuristic framework are in general human designed heuristics which are fixed before the search starts. Most of the existing selective hyper-heuristics are based on perturbative low level heuristics, and favor single point search. Cowling et al. in [57] investigated the performance of different hyper-heuristics, combining different heuristic selection, with different move acceptance methods on a real world scheduling problem. Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Greedy and Choice Function were introduced as heuristic selection methods. The authors utilized the following deterministic acceptance methods: All-Moves accepted and Only Improving moves accepted. The hyper-heuristic combining Choice Function with All-Moves acceptance performed the best. More elaborate acceptance mechanisms have been introduced.
and there is a growing body of comparative studies which evaluate the performances of different heuristic selection and acceptance combinations [1]. In [58], the performance of different hyper-heuristics are compared with different components emphasizing the influence of learning heuristic selection methods for solving a sports scheduling problem. The experiential result shows that the proposed approach is slightly better than the other approaches that use choice function as heuristic selection and great deluge as acceptance for solving a sports scheduling problem. In [59], a greedy heuristic selection strategy was presented which aims to determine low level heuristics with good performance based on the trade-off between the change (improvement) in the solution quality and the number of steps taken. This method performs well with respect to the mock competition hyper-heuristics on four problem domains. Recently, a wide empirical analysis was conducted in [60] to compare many Monte Carlo based hyper-heuristics for examination timetabling. The experimental results show that choice function-simulated annealing with reheating performs well.

In [2], the performance of seven different heuristic selection methods (Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Greedy, Choice Function and Tabu Search) which were combined with five acceptance methods (All-Moves, Only Improving, Improving and Equal, Exponential Monte Carlo with Counter and Great Deluge) were investigated. The resultant hyper-heuristics were tested on fourteen benchmark functions against genetic and memetic algorithms. The empirical results confirmed the success of memetic algorithms over genetic algorithms and the performance of a choice function based hyper-heuristic was comparable to the performance of a memetic algorithm. As described in [57], [61] the choice function heuristic selection method which adaptively ranks the low-level heuristics \((h_i)\) using

\[
f(h_i) = \alpha f_1(h_i) + \beta f_2(h_j, h_i) + \delta f_3(h_i)
\]

where \(f_1\) measures the individual performance of each low level heuristic, \(f_2\) measures the performance of pairs of low level heuristics invoked consecutively, and finally, \(f_3\) is the elapsed CPU time since the heuristic was last called. Both \(f_1\) and \(f_2\) support intensification while \(f_3\) supports diversification. The parameter values for \(\alpha\), \(\beta\) and \(\delta\) are changed adaptively based on a similar idea to reinforcement learning. In [61], the choice function based hyper-heuristic was applied to nurse scheduling and sales summit scheduling. The study shows that the choice function hyper-heuristic is successful in making effective use of low level heuristics, due to its ability of learning the dynamics between the solution space and the low level heuristics to guide the search process towards better quality solutions.

3) Selective Hyper-heuristics VS Hybrid Methods:
According to Ke Tang in [62], the idea of combining multiple algorithms is not new at all, and can be traced back to 1980s. In the context of multi-objective optimization and evolutionary computation, many methods are presented utilizing this idea, such as multimeme algorithms and adaptive multi-method/strategy ensemble algorithms (as called in some papers [12], [63]). Multimeme algorithms incorporates local search strategies within hybrid evolutionary algorithms. An example of multimeme approaches can be found in [50]. The adaptive multi-method/strategy ensemble algorithms rely on running multiple algorithms (can be MOEAs or Evolution Strategies) simultaneously and adaptively create the offspring. Multimeme algorithms are different from the adaptive multi-method/strategy ensemble algorithms as they are embedded local search strategies. However, both methods are closely similar to hyper-heuristics for multi-objective optimization problems. Other researchers would argue that the adaptive multi-method/strategy ensemble algorithms are hyper-heuristics methods. According to Burke et al. [1], the hyper-heuristics defined as:

“Hyper-heuristics comprise a set of approaches with the goal of automating, often by the incorporation of machine learning techniques, the process of either (i) selecting and combining simpler heuristics, or (ii) generating new heuristics from components of existing heuristics; in order to solve hard computational search problems.”

It hard to classify the adaptive multi-method/strategy ensemble algorithms as a selective nor generative hyper-heuristics. However, It can not eliminated them from the hyper-heuristics scope, as they combining different heuristics/ meta-heuristics. These methods are closely similar to the selective hyper-heuristic multi-objective methods in term of incorporation different algorithms. However, they are different from selective hyper-heuristic in their concept. The selective hyper-heuristic rely on two concepts selection mechanism and acceptance move strategy. Both concepts are not adopted in the adaptive multi-method/strategy ensemble algorithms. Moreover, multiple heuristic/meta-heuristics run concurrently in the adaptive multi-method/strategy ensemble algorithms. Each heuristic/meta-heuristics produce different population of offspring, then all produced offsprings are evaluated to evolve a new population of offspring by an adaptive creation offspring strategy. In selective hyper-heuristics multi-objective, a sequence of heuristic/meta-heuristic is executed during the search, i.e. one heuristic/meta-heuristic is selected and applied at each stage (iteration/decision point) of the search. The high level strategy in hyper-heuristics evaluates the performance of these heuristic/meta-heuristic in order to improve the population of solutions.

In our study, we propose a new online learning choice
function based selective hyper-heuristic framework which supports multi-point search and cooperative low level meta-heuristics for multi-objective optimization. All-Moves is utilized in this framework as an acceptance strategy. Further details of this hyper-heuristic are discussed in Section III.

D. Test suite for Multi-objective Optimization

Typically, a test suite should include different test problems which consist of a wide range of characteristics and features as mentioned in Section II-A. However, it is impractical to have a test suite that incorporates all possible combinations of features. The test suites most commonly employed as benchmark multi-objective problems in the MO/EA literature are the ZDT test suite [29], the DTLZ test suite [6], the WFG [7] and more recently LZ09 [64]. The LZ09 test suite has nine multimodal constrained test problems with complicate pareto sets and two objectives, except LZ09-F6, which is a tri-objective. As LZ09 is considered a relatively new test suite, no test results are available yet, other than those reported in the original study and [65].

The problem features in ZDT, DTLZ and WFG test suites are presented in Table I.

<table>
<thead>
<tr>
<th>Test Features</th>
<th>ZDT</th>
<th>DTLZ</th>
<th>WFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pareto 1-1</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Pareto M-1</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Flat Reigns</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Modality</td>
<td>Unimodal</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Deception</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Bias</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Pareto Front known</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Separability</td>
<td>Separable</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Scarity</td>
<td>No of Parameters</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Front Geometry</td>
<td>Convex</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Concave</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Disconnected</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Degenerate</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Linear</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Mixed</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

1) ZDT Test Suite: It was introduced in [29] which consists of six test problems. All the problems are separable and complicated enough to enable comparison over a variety of multi-objective evolutionary approaches. They also include some features which make the problems sufficiently difficult for optimizers such as multimodality, nonconvexity and deception. For all problems of ZDT, the global optimum has the same variable values for different decision variables and objectives and the POF is known in advance. In addition, the ZDT test suite has been widely used by many researchers in MO/EAs. Therefore, test results are available and can be easily accessed. However, ZDT has some limitations. In terms of scalability, the number of decision variables and objectives only has one decision variable with two objectives. Moreover, none of its test problems has fitness landscapes with flat regions, a degenerate Pareto front or even nonseparable features. In addition, the only deceptive problem is binary encoded. Also the global optimum for all ZDT problems lies on the lower bound, or in the centre of the search bounds [47].

2) DTLZ Test Suite: It was introduced in [6] that consists of seven different test problems. Similar to ZDT, the global optimum of DTLZ test problems has the same values for decision variables and objectives, all its problems are separable [47], and the POF is exactly known. However, it differs from ZDT in terms of its scalability. DTLZ is scalable to any number of objectives and distance parameters. However, DTLZ has several shortcomings. For all problems, the global optimum is situated in the center of the search range or on the bounds. None of these problems has fitness landscapes with flat regions, deceptive or nonseparable features. Moreover, the number of decision variables is always strongly tied to the number of objectives [7]. In addition, the increase in the number of objectives may cause difficulties for an optimizer to find the Pareto solutions [6], [66].

3) WFG Test Suite: The Walking Fish Group’s test suite (WFG) was created in [7]. It consists of nine test problems (see Table II). The benchmark problems fully satisfy the recommendations set out in Section II-A. The WFG is designed only for real valued parameters with no side constraints which make the problems easy to analyze and implement. The features of the WFG dataset are seen as the common choice for most MO/EA researchers [7]. Unlike most of the multi-objective test suites such as ZDT [29] and DTLZ [6], the WFG test suite [7] has powerful functionality; a number of instances that have features not included in other test suites. The benchmark problems are nonseparable problems, deceptive problems, a truly degenerate problem, and a mixed-shape Pareto front problem. In addition, WFG is scalable to any number of parameters and objectives, and the numbers of both distance- and position-related parameters can be scaled independently [7].

All WFG test problems are continuous problems that are constructed based on a vector that corresponds to the problem’s fitness space. This vector is derived through a series of transition vectors such as multimodality and nonseparability. The complexity of the problem can be increased according to the number of transition vectors. The main advantage of the WFG test suite is that it is an excellent tool for comparing the performance of EAs over a range of test problems, and it has been shown to have a more comprehensive set of challenges when compared to DTLZ using NSGAII in [7]. Therefore, the
### Table II

<table>
<thead>
<tr>
<th>WFG Test Functions</th>
<th>WFG1</th>
<th>WFG2</th>
<th>WFG3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WFG Test Functions</strong></td>
<td>( h_{M+1} : M = \text{convex}_m )</td>
<td>( h_{M} = \text{mixed}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
<td>( h_{M} = \text{disc}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
</tr>
<tr>
<td>( t_{i-1;k+1} = y_i )</td>
<td>( t_{i-1;k+1} = S_{\text{linear}}(y_i, 0.35) )</td>
<td>( t_{i-1;k+1} = S_{\text{linear}}(y_i, 0.35) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = b_{\text{flat}}(y_i, 0.8, 0.75, 0.85) )</td>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = b_{\text{poly}}(y_i, 0.02) )</td>
<td>( t_{i-1;k+1} = b_{\text{poly}}(y_i, 0.02) )</td>
<td>( t_{i-1;k+1} = b_{\text{poly}}(y_i, 0.02) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = \frac{1}{r_{\text{sum}}(y_i)} \frac{1}{y_i} )</td>
<td>( t_{i-1;k+1} = \frac{1}{r_{\text{sum}}(y_i)} \frac{1}{y_i} )</td>
<td>( t_{i-1;k+1} = \frac{1}{r_{\text{sum}}(y_i)} \frac{1}{y_i} )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td>( t_{i-1;k+1} = \left( \frac{1}{r_{\text{sum}}(y_i)} \frac{1}{y_i} \right) )</td>
<td>( t_{i-1;k+1} = \left( \frac{1}{r_{\text{sum}}(y_i)} \frac{1}{y_i} \right) )</td>
<td></td>
</tr>
<tr>
<td><strong>WFG Test Functions</strong></td>
<td><strong>WFG4</strong></td>
<td><strong>WFG5</strong></td>
<td><strong>WFG6</strong></td>
</tr>
<tr>
<td>( h_{M+1} : M = \text{concave}_m )</td>
<td>( h_{M} = \text{concave}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
<td>( h_{M} = \text{concave}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = S_{\text{linear}}(y_i, 0.35, 10, 0.35) )</td>
<td>( t_{i-1;k+1} = S_{\text{linear}}(y_i, 0.35, 10, 0.35) )</td>
<td>( t_{i-1;k+1} = S_{\text{linear}}(y_i, 0.35, 10, 0.35) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td>( t_{i-1;k+1} = \sum_{i=1}^{M} y_i )</td>
<td></td>
</tr>
<tr>
<td><strong>WFG Test Functions</strong></td>
<td><strong>WFG7</strong></td>
<td><strong>WFG8</strong></td>
<td><strong>WFG9</strong></td>
</tr>
<tr>
<td>( h_{M+1} : M = \text{concave}_m )</td>
<td>( h_{M} = \text{concave}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
<td>( h_{M} = \text{concave}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = b_{\text{param}}(y_i, \sum_{j=1}^{M} y_j, {1, ..., 1}) )</td>
<td>( t_{i-1;k+1} = b_{\text{param}}(y_i, \sum_{j=1}^{M} y_j, {1, ..., 1}) )</td>
<td>( t_{i-1;k+1} = b_{\text{param}}(y_i, \sum_{j=1}^{M} y_j, {1, ..., 1}) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = b_{\text{param}}(y_i, \sum_{j=1}^{M} y_j, {1, ..., 1}) )</td>
<td>( t_{i-1;k+1} = b_{\text{param}}(y_i, \sum_{j=1}^{M} y_j, {1, ..., 1}) )</td>
<td>( t_{i-1;k+1} = b_{\text{param}}(y_i, \sum_{j=1}^{M} y_j, {1, ..., 1}) )</td>
<td></td>
</tr>
<tr>
<td><strong>WFG Test Functions</strong></td>
<td><strong>WFG10</strong></td>
<td><strong>WFG11</strong></td>
<td><strong>WFG12</strong></td>
</tr>
<tr>
<td>( h_{M+1} : M = \text{concave}_m )</td>
<td>( h_{M} = \text{concave}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
<td>( h_{M} = \text{concave}_n \ (\text{with } \alpha = 1 \text{ and } A = 5) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td></td>
</tr>
<tr>
<td>As ( t_i ) from WFG1. (Linear shift.)</td>
<td>As ( t_i ) from WFG1. (Linear shift.)</td>
<td>As ( t_i ) from WFG1. (Linear shift.)</td>
<td></td>
</tr>
<tr>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td>( t_{i-1;k+1} = \text{decept}_n(y_i, 0.35, 0.001, 0.05) )</td>
<td></td>
</tr>
<tr>
<td>As ( t_i ) from WFG4. (weighted sum reduction.)</td>
<td>As ( t_i ) from WFG4. (weighted sum reduction.)</td>
<td>As ( t_i ) from WFG4. (weighted sum reduction.)</td>
<td></td>
</tr>
<tr>
<td>As ( t_i ) from WFG4. (weighted sum reduction.)</td>
<td>As ( t_i ) from WFG4. (weighted sum reduction.)</td>
<td>As ( t_i ) from WFG4. (weighted sum reduction.)</td>
<td></td>
</tr>
<tr>
<td>As ( t_i ) from WFG1. (Linear shift.)</td>
<td>As ( t_i ) from WFG1. (Linear shift.)</td>
<td>As ( t_i ) from WFG4. (weighted sum reduction.)</td>
<td></td>
</tr>
</tbody>
</table>

WFG test suite has been selected to be the benchmark test suite employed in the hyper-heuristic approach that we propose in this paper.

### III. A Multi-objective Hyper-heuristic

#### A. A Hyper-heuristic Framework

In this study, we proposed an online selective choice function based hyper-heuristic for multi-objective optimization problems namely (HHMO_CF) or for short (HH_CF). We utilize a choice function as a selection mechanism and All-Moves as an acceptance strategy. The selection process considers the performances of low level heuristics in order to select a suitable heuristic as the search progresses. The choice function acts as the high level strategy, which adaptively ranks the performance of three low level heuristics deciding which one to call at each decision point. All-Moves is employed as a deterministic acceptance strategy, meaning that we accept the output of each low level heuristic whether it improves the quality of the solution or not. The framework is flexible and could incorporate any meta-heuristic for multi-objective optimization. In this study, we incorporate a three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA) as low level heuristics. The motivation behind choosing these MOEAs is that they incorporate much of the known MOEA theory and their general algorithm complexity is less than other MOEAs [36].

The HH_CF framework is shown in Fig. 1. There is no direct information exchange between low level heuristics. And the high level strategy does not have any knowledge of the problem domain and solutions unless information of how the low level heuristic is perform. This is a separation of domain information known as the domain barrier.

To provide the knowledge of the problem domain to the high level strategy, four performance metrics presented in [21], are selected as a feedback mechanism. The high level strategy selects one low level heuristic at each decision point according to the information obtained from the feedback mechanism. Note that the three low level heuristics operating in encapsulated way. Each heuristic has its own characteristics described in section II-B, but they are sharing the same population.

The four performance metrics that we employ as a feedback mechanism are:

- **AE** (Algorithm effort) [21]: measures the computational effort of an algorithm to obtain the Pareto optimal set. It ranges from \([0, \infty]\). A smaller value of AE indicates better performance.

- **RNI** (Ratio of non-dominated individuals) [21]: evaluates the fraction of non-dominated individuals in the population. It ranges from \([0, 1]\). If RNI = 1, this indicates that all individuals for a given population are non-dominated.

- **SSC** (Size of space covered or so-called as S_metric Hypervolume) [43]: evaluates the size of the
objective functions space covered by the solutions around the POF. It ranges from \([0, \infty)\). A higher value of SSC indicates better performance.

- **UD** (Uniform distribution of a non-dominated population) [67]: evaluates the distribution of non-dominated individuals over the POF. It ranges from \([0,1]\). Higher value of UD indicates better performance.

These metrics are chosen as they have been commonly used for comparison performance of MOEAs to measure different aspects of the final population [21]. In addition, they do not require a prior knowledge of the POF, which make our framework is applicable to tackling a real-world application for the future studies.

**B. The Ranking Scheme**

The four performance metrics (AE, RNI, SSC, and UD) [21] provide information about the performance of the low level heuristics. They also provide an online learning mechanism in order to guide the high level strategy during the search and determine which low level heuristic should be selected next. Since those metrics are not in the same scalar units, it is difficult to determine which is the best heuristic according to the four performance metrics. Therefore, we use a ranking scheme to measure the performance of heuristics. This ranking scheme is simple and flexible to incorporate any number of heuristics and the performance indicators. Unlike the ranking scheme used in [68] which ranks the algorithms based on their probabilities against the performance indicators using mixture experiments, our ranking scheme rely on sorting the heuristics in descending manner based on the highest rank count among the other heuristics. For \(N\) number of heuristics and \(M\) number of performance metrics, \(N\) heuristics are ranked according to their performances against \(M\) metrics. For a particular metric \(m \in M\), a heuristic \(n \in N\) with the best performance among other heuristics assigns the highest rank, which is equal to \(N\). Then another heuristic with the second best performance is ranked as \(N-1\) and so on. If two heuristics have the same performance, both heuristics assign the same rank. This ranking process is applied for all \(M\) metrics. After all heuristics are ranked against all metrics, the frequency of the highest rank for each heuristic is counted. A heuristic with the biggest frequency count of the highest rank is more desirable to select. An example of how the ranking scheme works using the four performance metrics to rank three heuristics is described in Fig. 2.

As we not only look for the heuristic that has the best performance, but also aim to have a large number of non-dominated individuals, the frequency count of the highest rank for a heuristic \(h\) is summed with its RNI rank using:

\[
f_1(h) = \text{freq}_{\text{highest rank}}(h) + RNI_{\text{rank}}(h)
\]

\(f_1(h)\) represents the performance of heuristic \(h\) in the choice function. More details about the choice function are discussed in the next sub-section. In case of two heuristics have the same value of \(f_1(h)\), we consider the heuristic that has more count of the second highest rank \((N-1)\).
The Multi-objective Hyper-heuristic Algorithm

The largest value of \( CF \) are in the same scalar unit. The low level heuristic with a balance between \( f \) is a large positive value (e.g. 100). It is important to strike a balance between \( f \) and the three low level heuristics. The choice function reflects the overall performance of every low level heuristic \( h \). Thus we have

\[
CF(h) = \alpha f_1(h) + f_2(h)
\]  

Equation (3) differs from (1). \( f_2(h) \) in (1) is discarded in (3). Thus in (3), we have \( f_1(h) \), which reflects the performance of each low level heuristic \( h \) against the four metrics (AE, RNI, SSC, and UD) and is computed in (2) using the ranking scheme described earlier. \( f_1(h) \) is the number of CPU seconds elapsed since the heuristic was last called. This provides an element of diversification, by favoring those low level heuristics that have not been called recently. \( \alpha \) is a large positive value (e.g. 100). It is important to strike a balance between \( f_1(h) \) and \( f_2(h) \) values, so that they are in the same scalar unit. The low level heuristic with the largest value of \( CF(h) \) is selected to apply.

C. The Choice Function As A High Level Strategy

The choice function reflects the overall performance of every low level heuristic \( h \). Thus we have

\[
CF(h) = \alpha f_1(h) + f_2(h)
\]  

Equation (3) differs from (1). \( f_2(h) \) in (1) is discarded in (3). Thus in (3), we have \( f_1(h) \), which reflects the performance of each low level heuristic \( h \) against the four metrics (AE, RNI, SSC, and UD) and is computed in (2) using the ranking scheme described earlier. \( f_1(h) \) is the number of CPU seconds elapsed since the heuristic was last called. This provides an element of diversification, by favoring those low level heuristics that have not been called recently. \( \alpha \) is a large positive value (e.g. 100). It is important to strike a balance between \( f_1(h) \) and \( f_2(h) \) values, so that they are in the same scalar unit. The low level heuristic with the largest value of \( CF(h) \) is selected to apply.

D. The Multi-objective Hyper-heuristic Algorithm

Our hyper-heuristic approach is shown in Algorithm 1. Initially, a greedy algorithm is applied to determine the best low level heuristic to be selected for the first iteration (step 2-6). All three low level heuristics are run (step 3). Then, the three low level heuristics are ranked by using (2) and their choice function values are computed by using (3) (steps 4 & 5). The low level heuristic with the largest choice function value is selected (step 6) to be applied as an initial heuristic (step 8). Then, for all low level heuristics, the ranking mechanism is updated (step 9). The choice function values are also computed and updated (step 10). According to the updated choice function values, the low level heuristic with the largest choice function value is selected to apply in the next iteration (step 11). This process is repeated until the stopping condition is met (steps 7-12). Note that the greedy algorithm is applied only once at the beginning of the search, in order to determine which low level heuristic to apply first. Then, only one low level heuristic is selected at each iteration.

Algorithm 1 Multi-Objective Hyper-heuristic Algorithm

1: procedure \( HH_{CF}(H) \) where \( H \) is a set of the low level heuristics.
2: Initialization
3: Run \( h, \forall h \in H \)
4: Rank \( h, \forall h \in H \) based on the ranking scheme
5: Get \( CF(h), \forall h \in H \)
6: Select \( h \) with the largest \( CF(h) \) as an initial heuristic
7: repeat
8: Execute the selected \( h \)
9: Update the rank of \( h, \forall h \in H \) based on the ranking scheme
10: Update \( CF(h), \forall h \in H \)
11: Select \( h \) with the largest \( CF(h), \forall h \in H \)
12: until (termination criteria are satisfied)
13: end procedure

IV. The Comparison with Low Level Heuristics

The experiments are conducted for two purposes: 1) to show that \( HH_{CF} \) performs significantly differently from the low level heuristics, that is the MO algorithms (NSGAII, SPEA2, and MOGA) when used in isolation. 2) to gauge the proposed approach using the WFG benchmark dataset, by comparing with the low level heuristics being used isolation. Although NSGAII and SEPA2 have previously been applied to the WFG test suite in [49], we apply them and MOGA under our own experimental settings that are presented later.

A. Performance Evaluation Criteria

The comparison of the quality of solutions for multi-objective optimization is more complex than single-objective problems. The number of non-dominated individuals should be maximized, the distance of the non-dominated front should minimized, i.e. the resulting non-dominated set should be distributed uniformly as much as possible and converge well toward the POF. Because of that, we use three performance metrics RNI, SSC, and UD, to assess the quality of approximation sets in different aspects. In addition, we used the students t-test (t-test) statistic to compare the HH_CF and the three low level heuristics being used in isolation against each other with respect to these performance metrics. So, a series of experiments, with \( m \) associated (t-test) are conducted for HH_CF and the three low level heuristics. Where \( m \) represents the number of the performance metrics.
T-test statically compare if the means of two performance value sets, resulting from the performance of two algorithms with respect a particular metric, are the same. The null hypothesis is as follows:

\[
H_0 \quad \text{two performance value sets have same means} \\
H_1 \quad \text{two performance value sets have different means}
\]

We assume two independent samples, unequal variance and one-tailed distribution at \( \alpha = 0.05 \) (i.e. a 95% confidence level). We aim to reject the null hypothesis and accept the alternative hypothesis and demonstrate the performance of HH_CF is statistically different to the performance of other algorithms.

### B. Experimental Settings

All experimental parameters are chosen accordingly to that commonly used in the literature for the continuous problems [29], [7].

The nine test problems for the WFG suite (WFG1-WFG9) have 24 real parameters including one position parameter, 23 distance parameters and two objectives. We agree with [29] that two objectives are enough to represent the essential features of multi-objective optimization problems to demonstrate the significance of the proposed approach. According to [69], [70] an algorithm could reach better convergence by 6,250 generations. Therefore, the HH_CF was terminated after 6,250 generations. That is, HH_CF runs for a total of 25 iterations. In each iteration, one low level heuristic is applied and this is executed for 250 generations with population size equal to 100. The secondary population of SPEA2 is set to 50. The execution time takes about 10-30 minutes depending on the given problem. In order to make a fair comparison, each low level heuristic that used in isolation was terminated after 6,250 generations.

For the WFG problems, 30 independent trials were run for each algorithm with a different random seed. The crossover and mutation probability were set to 0.9 and 1/24 respectively. The distribution indices for crossover and mutation were set to 10 and 20 respectively. In the measure of SSC, the reference points for WFG problems with \( k \) objectives was set \( r_i = (0, i+2), i = 1, \ldots, k \) [7]. The distance sharing \( \sigma \) for the UD metric and MOGA was set to 0.01 in the normalized space. These settings were used for SSC and UD as a feedback indicator in the ranking scheme of HH_CF and as a performance measure for the comparison. All algorithms were implemented with the same common sub-functions using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz 2G\( \backslash \)250G computer.

### C. Experimental Results and Discussion

The statistical t-test results of comparing our proposed HH_CF and the three low level heuristics that used in isolation (NSGAII, SPEA2, and MOGA) for the three performance metrics (RNI, SSC, and UD) on the nine WFG test problems are given in Table III. The \( +/- \) indicates the two algorithms have different mean which reflect that two algorithms perform differently according to a particular performance metric. The + sign refers to first algorithm is better than the second algorithm. Whereas the – sign refers to the first algorithm is worse than the second algorithm. The ∼ sign indicates the two algorithms have same mean which reflect that they perform similarly according to a particular performance metric. \( n/a \) means the t-test is not applicable for two samples since they are completely equal.

As we can notes from the Table III, the HH_CF and other algorithms are statistically different in the majority cases (i.e. we reject the null hypothesis).

The performance values of HH_CF and other algorithms with respect to the performance metrics (RNI, SSC and UD) on the WFG problems are summarized in Table IV. For each performance metric, the average and standard deviation values are computed. For all performance metrics, a higher value indicates a better performance. In WFG1, HH_CF has higher RNI value than MOGA while it has lower value than NSGAII and SPEA2. HH_CF has the highest value of SSC and UD among the algorithms. We can put WFG5 and WFG6 in this category. In WFG2 and WFG3, HH_CF has a RNI's value similar to MOGA and lower than others. With respect to SSC, HH_CF has higher values than SPEA2 and MOGA and similar to NSGAII. However, HH_CF has the highest value among other methods in the measure of UD. In WFG4 and WFG7, HH_CF has the lowest(worst) RNI value and the highest UD value. It has a higher value than MOGA similar to NSGAII and SPEA2 with respect to SSC metric. In WFG8 and WFG9, the HH_CF has the lowest value with respect to RNI and SSC metrics, and the highest value with respect to UD metric.

These performance results with respect to RNI, SSC and UD are also displayed into box plots in Figs. 3, 4 and 5, in order to provide a clear visualization of the distribution of the simulation data of the 30 independent runs.

In Fig. 3, NSGAII and SPEA2 perform better than other and produce the highest value of RNI for all datasets. However, HH_CF and MOGA produce relatively low values for this metric. This indicates that HH_CF preforms badly according to the metric of RNI and produces a low number of non-dominated solutions than other algorithms except MOGA. In Fig. 4, it can be seen that HH_CF has the highest uniform distribution’s value(UD) in all test problems. This indicates that HH_CF is superior to the other algorithms on all WFG problems in terms of the distribution of non-dominated individuals over the POF.

In Fig. 5, the performance of HH_CF for SSC relatively better than SPEA2 and MOGA in all test problems. It is also better than NSGA2 in WFG1,WFG5 and WFG6. However, It is moderate and similar to NSGAII in WFG2, WFG3, WFG4, and WFG7. In WFG8 and
Dominated solutions that distribute uniformly well over the POF comparing the other methods. HH_CF also performs better than the others in the most WFG problems and produces non-dominated solutions with high diversity that covering a larger proportion of objective space, except in WFG8 and WFG9 where it failed to converge towards the POF. As WFG8 and WFG9 has a significant bias feature, HH_CF might face a difficulty coping with this bias.

Generally, HH_CF could produces competitive results in the most WFG test problems with respect to two performance metrics SSC and UD out of the three metrics. Although HH_CF obtains low number of solutions, it produces very good solutions in terms of diversity and convergence when comparing to the low level heuristics that used in isolation. HH_CF can benefit from the strengths of the low level heuristics. Moreover, it has the capability to intelligently adapt to calling combinations of low level heuristics. To understand how the HH_CF could obtain these results, we analyze the behavior of the low level heuristics in the next subsection.

### D. Analysis the Behavior of Low Level Heuristics

We compute the average heuristic utilization rate which indicates how frequently a given low level heuristic is chosen and applied during the whole search process.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Methods</th>
<th>RNI</th>
<th>SSC</th>
<th>UD</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFG1</td>
<td>HH_CF-SPEA2</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>WFG1</td>
<td>MOGA</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

WFG9, HH_CF do not perform well comparing to the other methods except MOGA.

We can note from all above results that HH_CF performs worse than the low level heuristics that used in isolation and produces low number of non-dominated solutions in the most of WFG problems. However, HH_CF performs very well and produces non-dominated solutions that distribute uniformly well over the POF comparing the other methods. HH_CF also performs better than the others in the most WFG problems and produces non-dominated solutions with high diversity that covering a larger proportion of objective space, except in WFG8 and WFG9 where it failed to converge towards the POF. As WFG8 and WFG9 has a significant bias feature, HH_CF might face a difficulty coping with this bias.

Generally, HH_CF could produces competitive results in the most WFG test problems with respect to two performance metrics SSC and UD out of the three metrics. Although HH_CF obtains low number of solutions, it produces very good solutions in terms of diversity and convergence when comparing to the low level heuristics that used in isolation. HH_CF can benefit from the strengths of the low level heuristics. Moreover, it has the capability to intelligently adapt to calling combinations of low level heuristics. To understand how the HH_CF could obtain these results, we analyze the behavior of the low level heuristics in the next subsection.

### D. Analysis the Behavior of Low Level Heuristics

We compute the average heuristic utilization rate which indicates how frequently a given low level heuristic is chosen and applied during the whole search process.
Fig. 3. Box plots of NSGAII, SPEA2, MOGA and HH_CF for the measure of ratio of non-dominated individuals (RNI) on the WFG test functions.

Fig. 4. Box plots of NSGAII, SPEA2, MOGA and HH_CF for the measure of hypervolume (SSC) on the WFG test functions.
across all runs in order to see which low level heuristic is used more frequently. The results are presented in Fig. 6. The average heuristic utilization rate of NSGAII is at least 44% and is the highest among all low level heuristics for each problem, except WFG5 for which SPEA2 is chosen most frequently with a utilization rate of 55.72% during the search process. This is explain why, HH_CF has either a similar or relatively better convergence to the POF for most of the test problems when compared with NSGAII. This is indicates that NSGAII performs best among other low level heuristics in most of the WFG problems. The authors theorize that HH_CF, therefore, prefers NSGAII and it becomes preferable to be chosen more frequently than the other low level heuristics. Our result is consistent with the result in [49] that show the best performance is achieved by NSGAII on the WFG test suite. The performance of MOGA is not that good on the WFG test, thus it is invoked relatively less frequently during the search process because of the diversification factor $f_2$ in (3). However, MOGA still influences the performance of HH_CF, negatively, in particular with respect to the RNI metric. This is due to the fact that MOGA does not have any archive mechanism or preserving strategy to maintain the non-dominated solutions during the search. The average utilization rate of MOGA is the highest for WFG8 (10.16%) and WFG9 (22.40%) among other WFG problems. This utilization rate explains why the performance of HH_CF is the worst performing approach in terms of RNI. HH_CF also faces some difficulty while solving WFG8 and WFG9 in terms of convergence as well.

In order to see the effectiveness of each chosen low level heuristic on the performance of HH_CF, we looked into the performance of low level heuristics with respect to the RNI, SSC and UD metrics at twenty five decision points during the whole search process. We observe that some problems are following a specific pattern while the low level heuristics during the search. Each problem has its own pattern. For example, for WFG3, NSGAII is invoked and executed for the first seven consecutive decision points. Then SPEA2 is invoked for the next four decision points, followed by one iteration for MOGA. Then NSGAII is chosen for the rest of the search. More of these patterns are illustrated in Fig. 7.

In order to analyze these results easily, we divide WFG problems into four categories based on the performance of HH_CF compring to the three low level heuristics being used in isolation with respect to RNI, SSC and UD as listed below:

1) WFG1, WFG5 and WFG6:
   - RNI: Better performance than MOGA and worse than NSGAII and SPEA2
   - SSC: The best performance among NSGAII, SPEA2 and MOGA
   - UD: The best performance among NSGAII, SPEA2 and MOGA
2) WFG2 and WFG3:
   - RNI: Similar performance to MOGA and worse than NSGAII and SPEA2
   - SSC: Better performance than SPEA2 and MOGA and similar to NSGAII
   - UD: The best performance among NSGAII, SPEA2 and MOGA

3) WFG4 and WFG7:
   - RNI: The worst performance among NSGAII, SPEA2 and MOGA
   - SSC: Better performance than SPEA2 and MOGA and similar to NSGAII
   - UD: The best performance among NSGAII, SPEA2 and MOGA

4) WFG8 and WFG9:
   - RNI: The worst performance among NSGAII, SPEA2 and MOGA
   - SSC: The worst performance among NSGAII, SPEA2 and MOGA
   - UD: The best performance among NSGAII, SPEA2 and MOGA

For each category described above, except the last one, we have selected a sample problem to visualize the low level call patterns. WFG5 for the first category, WFG3 for the second category and WFG4 for the third category. For the last category, no specific pattern has been observed. The selected three problems have different problems features in terms of separability and modality (Huband et al., 2006). The average of RNI, SSC and UD values versus decision point plots across selected benchmark problems (WFG3, WFG4 and WFG5) are shown in Fig. 7. Each step in the plot is associated with the most frequently selected low level heuristics across 30 trials. Since we employed All-Moves as an acceptance strategy, some moves are accepted even if it worsens the solution quality.

From Fig. 7, it is clear that MOGA, during the search, produces a worse solution with respect to RNI, and this solution is accepted which affects the performance of HH_CF. However, some worsened moves are able to produce better solution. This can be noted in the performance of UD. SPEA2 produce low quality solutions but this helps it to escape from the local optimum and obtain better solutions at the end. This is also true with respect to the SSC performance indicator. In addition, we can note that HH_CF has an advantage over MOGA and outperforms the three MOEAs methods with respect to the distribution of non-dominated individuals over the Pareto optimal front. It also has an advantage over NSGAII in terms of convergence, that it performs better than all methods in some problems while performing better or similar to NSGAII on the other problems. However, HH_CF does not have an advantage over NSGAII and SPEA2 with respect to the non-dominated individuals in the population. HH_CF performs poorly because of MOGA’s effect. This could be avoided by employing another acceptance move strategy instead of All-Moves. A non-deterministic acceptance strategy could accept worsening moves within a limited degree and help improve the quality of the solutions.

It can be concluded that our choice function hyper-heuristic can benefit from the strengths of the low level heuristics. And it can avoid the weaknesses of them (partially), as the poor performance of MOGA affects the performance of HH_CF badly in the metric of RNI by producing low number of non-dominated solutions. We can avoid this by employing another acceptance move strategy instead of All-Moves. However, HH_CF has the capability to intelligently adapt to calling combinations of low level heuristics.

The tentative research questions that arise from these observations are whether there is any relation between the ratio of non-dominated individuals (RNI) of the obtained front and the uniform distribution (UD) of this front over the Pareto optimal front, and how the ratio of non-dominated individuals (RNI) would affect the other performance metrics in particular uniform distribution (UD).

V. THE COMPARISON WITH MULTI-OBJECTIVE HYPER-HEURISTICS

The experiments are conducted to examine the performance of our proposed HH_CF comparing with two multi-objective hyper-heuristics; a random hyper-heuristics (HH_RAND) and the adaptive multi-method search (AMALGAM) [12]. In a random hyper-heuristics (HH_RAND), we employed a simple random selection instead of the choice function selection that used in HH_CF. No ranking scheme is embedded into HH_RAND. In HH_RAND, we used the same three low level heuristics that used in HH_CF.

A. Performance Evaluation Criteria

The hypervolume (SSC) [43] and the generational distance (GD) [71] metrics were used to compare the performance of multi-objective hyper-heuristics. The GD measure the distance (convergence) between the approximation non-dominated front and the POF. A smaller value of GD is more desirable and it indicates that the approximation non-dominated front is more closer to the POF.

B. Experimental Settings

All experimental parameters are chosen accordingly to that commonly used in the literature for the continuous problems [29] and [7]. All methods were applied to the nine WFG test problems with 24 real values and two objectives. For each problem, the comparison methods ran 30 independent trials. In each run, they were executed for 25,000 evaluation functions with 100 population size
Fig. 6. The average heuristic utilization rate for the low level heuristics (NSGAII, SPEA2 and MOGA) in HH_CF on the WFG test suite

Fig. 7. The average of RNL, SSC and UD values versus decision point steps plots across selected benchmark problems (the WFG3, WFG4 and WFG5). Each step in the plot is associated with the most frequently selected low level heuristics across 30 trials.
and 250 generations. Both HH_CF and HH_RAND are executed 2500 evaluation functions at each iteration. In this experiment, we decreased the computational cost due to the resource limitation. Depending on the given problem, the execution time of HH_CF and HH_RAND for one run takes about 5-12 minutes while it takes about 20-50 minutes for AMALGAM. Other parameter settings of AMALGAM are identical to those used in [12]. HH_CF and HH_RAND have been implemented with the same common sub-functions using Microsoft Visual C++ 2008. As for AMALGAM, we used its Matlab's code that obtained from the author upon a request. We implemented a C++ interface between AMALGAM and the WFG test suite's C++ code. All methods were run on an Intel Core2 Duo 3GHz\2G\250G computer. Other parameter settings are similar to those that presented in Section IV-B.

C. Experimental Results and Discussion

The performance values of HH_CF and the other hyper-heuristics methods with respect to the performance metrics SSC and GD on the WFG problems are summarized in Table V. For each performance metric, the average and standard deviation values are computed. From these results, it is clear that HH_CF outperforms others methods in terms of both performance metrics on the most of WFG problems. As expected, HH_CF achieves better coverage and diversity than HH_RAND according to the both metrics. This due to the learning mechanism that used in HH_CF which adaptively guide the search toward the true Pareto front. Interestingly, HH_RAND preforms better than AMALGAM according to the hypervolume metric except in WFG9. However, It preforms worse than AMALGAM according to the GD metric in the all problems.

Comparing to AMALGAM, HH_CF preforms better with respect to the convergence and diversity in the most of the WFG problems. According to SSC metric, HH_CF produced non-dominated solutions that covering a larger proportion of objective space than AMALGAM in all WFG problems except in WFG9. The superiority of HH_CF in SSC metric is due to the stronger selection mechanism and the effective ranking scheme that rely on choosing the right heuristic with the best SSC value in the right time (decision point) to guide the search to move toward more spaces around the POF. This result is more reliable as shown in Fig. 8, where these performance results with respect to SSC is displayed into box plots in order to provide a clear visualization of the distribution of the simulation data of the 30 independent runs.

According to the metric of GD, HH_CF is superior to AMALGAM in the most of WFG problems (5 out 9) WFG1, WFG2, WFG5, WFG6, and WFG7 as reported in Table V and displayed into box plots in Fig. 9. Again, this result because of the effective online-learning selection mechanism and the ranking scheme in HH_CF. The ranking scheme maintains the past performance of low level heuristics using a set of performance indicators that measure different aspects of the solutions. During the search process, the ranking scheme creating a kind of balance between choosing the low level heuristics and their performances according to a particular metric. This balance enhances the algorithm performance to yield better solutions that converge well toward the POF as well as distribute uniformly well along the POF. However, AMALGAM is slightly preform better than HH_CF in the rest four WFG problems for GD. This might because of nature of the problems that make some difficulties for HH_CF to converge toward the POF or might slow down the convergence speed such as the bias in WFG8, WFG9 and the multimodality of WFG4. It is good to report that AMALGAM has better performance according to the both metrics; SSC and GD in WFG9.

For each problem, we computed the 50% attainment surface for each algorithm, from the 30 fronts after 25,000 evaluation functions. In Fig. 10, we have plotted the POF and the 50% attainment surface of the algorithms. HH_CF shows good convergence and well uniform distribution for most datasets. It seems clear that HH_CF has quickly converged on the POF in WFG1 and WFG2 compared to other algorithms. Moreover, HH_CF produced solutions that covered larger proportion of objective space comparing with other algorithms. AMALAGAM has poor convergence in the most problems. It has few number of solutions with poor convergence in WFG2. And it has no solutions in the middle-lower segments of the POF in WFG3, WFG5, WFG6, WFG7, and WFG8 and no solutions in the upper-middle segments of the POF in WFG4.

It can concluded that all above results demonstrate the effectiveness of this hyper-heuristic approach HH_CF in teams of its capability to intelligently adapt to calling combinations of low level heuristics and outperforming other hyper-heuristics for multi-objective optimization for solving this kind of problems.

VI. CONCLUSION

Hyper-heuristics have drawn increasing attention from the research community in recent years, although their roots can be traced back to the 1960's. They perform a search over the space of heuristics rather than searching over the solution space directly. Research attention has focussed on two types of hyper-heuristics: selection and generation. A selective hyper-heuristic manages a set of low level heuristics and aims to choose the best heuristic at any given time using historic performance to make this decision, along with the need to diversify the search at certain times. In this study, we present an online selective hyper-heuristic choice function based for multi-objective optimization
(HHMO_CF) or (HH_CF) for short, that controls and combines the strengths of three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA), which are utilized as the low level heuristics.
A choice function acts as the high level strategy, which adaptively ranks the performance of three low-level heuristics, deciding which one to call at each decision point. All-Moves is employed as an acceptance strategy, meaning that we accept the output of each low level heuristic whether it improves the quality of the solution or not. Four performance metrics (Algorithm effort (AE), Ratio of non-dominated individuals (RNI), Size of space covered (SSC) and Uniform distribution of a non-dominated population (UD)) act as an online learning mechanism to provide knowledge of the problem domain to the high level strategy.

We have conducted a number of experiments to analyze our approach HH_CF and compared its performance to the low level heuristics (NSGAIL, SPEA2 and MOGA), when used in isolation over the WFG test suite which utilized as our benchmark instances. The experimental results shows that the choice function based hyper-heuristic can benefit from the strengths of the low level heuristics. Moreover, it has the capability to intelligently adapt to calling combinations of low level heuristics. Our hyper-heuristic performs well with respect to the non-dominated individuals in the population (RNI). Another acceptance strategy instead of All-Moves can be employed to avoid this and improve the quality of solutions.

We have also conducted a number of experiments to examine the performance of our proposed HH_CF comparing with two multi-objective hyper-heuristics; a random hyper-heuristics HH_RAND and the adaptive multi-method search AMALGAM. The experimental results demonstrate the effectiveness of this hyper-heuristic approach. HH_CF outperforms other hyper-heuristics for multi-objective optimization according to the performance metrics SSC and GD in the most of the WFG problems.

The framework in which HH_CF is used for managing a set of multi-objective metaheuristics offers interesting potential research directions in multi-objective optimization. There is strong empirical evidence showing that different combinations of heuristic selection and acceptance methods in a selection hyper-heuristic framework yield different performances in single objective optimization [60]. We have adapted choice function heuristic selection for multi-objective optimization in this study. More heuristic selection and even move acceptance methods can be adapted from previous research in single objective optimization and used for multi-objective optimization. This process is not a trivial process requiring elaboration of existing methods and their usefulness in a multi-objective setting. Our aim is to test the level of generality of our framework further on some other domains including real-world problems.

### Table V

<table>
<thead>
<tr>
<th>Methods</th>
<th>SSC AVG</th>
<th>SSC STD</th>
<th>GD AVG</th>
<th>GD STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFG1</td>
<td>HH_CF</td>
<td>12.0044</td>
<td>0.5801</td>
<td>0.00744</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>7.0258</td>
<td>0.7977</td>
<td>0.0242</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>7.7902</td>
<td>0.1941</td>
<td>0.0292</td>
</tr>
<tr>
<td>WFG2</td>
<td>HH_CF</td>
<td>11.0102</td>
<td>0.2535</td>
<td>0.00466</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>9.7547</td>
<td>0.5078</td>
<td>0.0168</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>1.7582</td>
<td>0.8210</td>
<td>0.0010</td>
</tr>
<tr>
<td>WFG3</td>
<td>HH_CF</td>
<td>11.1250</td>
<td>0.0743</td>
<td>0.0007</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>11.0220</td>
<td>0.1490</td>
<td>0.0038</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>6.6890</td>
<td>0.0049</td>
<td>0.00036</td>
</tr>
<tr>
<td>WFG4</td>
<td>HH_CF</td>
<td>9.5610</td>
<td>0.0143</td>
<td>0.0010</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>9.2052</td>
<td>0.0145</td>
<td>0.0041</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>3.5687</td>
<td>0.0075</td>
<td>0.00081</td>
</tr>
<tr>
<td>WFG5</td>
<td>HH_CF</td>
<td>9.2701</td>
<td>0.5343</td>
<td>0.00253</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>9.2577</td>
<td>0.0556</td>
<td>0.0028</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>6.3554</td>
<td>0.0323</td>
<td>0.0028</td>
</tr>
<tr>
<td>WFG6</td>
<td>HH_CF</td>
<td>9.3579</td>
<td>0.0530</td>
<td>0.00225</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>9.3119</td>
<td>0.0501</td>
<td>0.0033</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>6.3554</td>
<td>0.0323</td>
<td>0.0030</td>
</tr>
<tr>
<td>WFG7</td>
<td>HH_CF</td>
<td>9.6498</td>
<td>0.0901</td>
<td>0.00474</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>9.1184</td>
<td>0.3473</td>
<td>0.0043</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>3.9171</td>
<td>0.0035</td>
<td>0.0007</td>
</tr>
<tr>
<td>WFG8</td>
<td>HH_CF</td>
<td>8.2643</td>
<td>0.1851</td>
<td>0.0044</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>8.1089</td>
<td>0.3867</td>
<td>0.0114</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>3.9405</td>
<td>0.0213</td>
<td>0.00241</td>
</tr>
<tr>
<td>WFG9</td>
<td>HH_CF</td>
<td>8.5981</td>
<td>0.2143</td>
<td>0.0053</td>
</tr>
<tr>
<td></td>
<td>HH_RAND</td>
<td>8.4697</td>
<td>0.3059</td>
<td>0.0060</td>
</tr>
<tr>
<td></td>
<td>AMALGAM</td>
<td>9.0676</td>
<td>0.1140</td>
<td>0.00113</td>
</tr>
</tbody>
</table>

### VII. Acknowledgement

The first author would like to thank Shahriar Asta for his help in implementing a C++ interface between AMALGAM and the WFG test suite.

### REFERENCES


Fig. 10. Pareto optimal front and 50% attainment surfaces for AMALGAM, HH_RAND and HH_CF on the WFG test functions