Choice Function based Hyper-heuristics for Multi-objective Optimization

Mashael Maashi[1a], Graham Kendall[2], Ender Özcan[3]

Automated Scheduling, Optimisation and Planning Research Group,
University of Nottingham, School of Computer Science, Jubilee Campus,Wollaton Road,
Nottingham, NG8 1BB ,UK.
[1]*email: m.maashi@alumni.nottingham.ac.uk*
[2]*The University of Nottingham Malaysia Campus*
email:graham.kendall@nottingham.edu.my
[3]*email: ender.ozcan@nottingham.ac.uk*

Abstract

A selection hyper-heuristic is a high level search methodology which operates over a fixed set of low level heuristics. During the iterative search process, a heuristic is selected and applied to a candidate solution in hand, producing a new solution which is then accepted or rejected at each step. Selection hyper-heuristics have been increasingly, and successfully, applied to single-objective optimization problems, while work on multi-objective selection hyper-heuristics is limited. This work presents one of the initial studies on selection hyper-heuristics combining a choice function heuristic selection methodology with great deluge and late acceptance as non-deterministic move acceptance methods for multi-objective optimization. A well known hypervolume metric is integrated into the move acceptance methods to enable the approaches to deal with multi-objective problems. The performance of the proposed hyper-heuristics is investigated on the Walking Fish Group test suite which is a common benchmark for multi-objective optimization. Additionally, they are applied to the vehicle crashworthiness design problem as a real-world multi-objective problem. The experimental results demonstrate the effectiveness of the non-

[a]Dr Maashi (m.maashi@gmail.com) works as an Assistant Professor at the Department of Computer Science, Faculty of Computer and Information Technology, University of Tabuk, Saudi Arabia.

deterministic move acceptance, particularly great deluge when used as a component of a choice function based selection hyper-heuristic.

Keywords: Hyper-heuristic, metaheuristic, great deluge, late acceptance, multi-objective optimization

---

1. Introduction

Hyper-heuristics perform a search over the space of heuristics when solving problems. In a hyper-heuristic approach, different heuristics or heuristic components can be selected, generated or combined to solve a given computationally difficult optimization problem in an efficient and effective way. A selection hyper-heuristic, which is the focus of this study, manages a predetermined set of low level heuristics with the goal of choosing the best one at any given time using a performance measure maintained for each low level heuristic. This type of hyper-heuristic comprises two main stages: *heuristic selection* and *move acceptance* strategy. A selection hyper-heuristic is often described as *heuristic selection-move acceptance*. Hyper-heuristics are sufficiently general and modular search methods enabling reuse of their components for solving problems from different domains [1]. The task of heuristic selection, also referred to as the *high level strategy*, is to guide the search intelligently and adapt taking into account the success/failure of the low level heuristics or combinations of heuristic components during the search process.

The low level heuristics in a selection hyper-heuristic framework are in general human designed heuristics which are fixed before the search starts. An initial solution (or a set of initial solutions) is iteratively improved using the low level heuristics until some termination criteria are satisfied. During each iteration, the heuristic selection decides which low level heuristic will be employed next. After the selected heuristic is applied to the current solution(s), a decision is made whether to accept the new solution(s) or not using an acceptance criteria. Usually, in a selection hyper-heuristic framework, there is a clear separation between the high level strategy and the set of low-level heuristics or heuristic components. It is assumed that there is a domain barrier between them [2]. The purpose of domain barrier is increase the level of the generality of hyper-heuristics by being able to apply it to a new of problem without changing the framework. Only a new set of problem-related low-level heuristics

need to be 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 [3]. 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).

Many real-world optimization problems are multi-objective requiring improvement of more than one objective, simultaneously. Often, there is some trade-off between multiple conflicting objectives [4, 5, 6, 7]. Hence, the multi-objective approaches provide a set of improved solutions (not a single solution as in single objective optimization) capturing the trade-off between those objectives for a given problem at the end of the search process. There is a variety of population based approaches for multi-objective optimization in the scientific literature, such as NSGAII [8], SPEA2 [9], and MOGA [10]. However, there are a few studies on multi-objective selection hyper-heuristics. To the best of the authors' knowledge, this paper is one of the first studies that investigate the influence of the move acceptance component on the performance of a selection hyper-heuristic for multi-objective optimization. In this study, we extend our previous work in [11] which describes a HHMO_CF_AM multi-objective hyper-heuristic controlling a set of low level (meta-)heuristics (NSGAII [8], SPEA2 [9], and MOGA [10]). We have adopted the great deluge algorithm (GDA) and late acceptance (LA) separately as a non-deterministic move acceptance component of a selection hyper-heuristic for multi-objective optimization and we have tested the performance of the overall algorithm using the same set of low level heuristics as in our previous study on the well-known Walking Fish Group (WFG) benchmark instances [12]. Moreover, we have applied the proposed selection hyper-heuristics with embedded GDA and LA, on a multi-objective real-world problem of *vehicle crashworthiness* [13] for which a solution aims to provide a vehicle design satisfying multiple objectives reducing different types of injuries as much as possible for the passengers within the vehicle during a crash. The empirical results are aligned with the previous observations for single objective optimization [14] that different combinations of heuristic selection and move acceptance under a selection hyper-heuristic framework yield different performances. Move acceptance components could be extremely influential on the overall performance of a selection hyper-heuristic. Moreover, the proposed

multi-objective hyper-heuristic, embedding GDA, turns out to be an effective, reusable and general approach for multi-objective optimization. The empirical results show that it is the best option as a multi-objective selection hyper-heuristic move acceptance component, outperforming each individual low level (meta-)heuristic run on their own for the WFG instances and NSGA II for the vehicle crashworthiness design problem.

The rest of the paper is organized as follows. In Section 2, a broad overview of the scientific literature on move acceptance methods, in particular the great deluge and late acceptance algorithms, is provided. An overview of existing studies on multi-objective selection hyper-heuristics and a selection hyper-heuristic framework supporting the use of great deluge and late acceptance move acceptance methods for multi-objective optimization are covered in Section 3. The experimental results for the proposed hyper-heuristics to the WFG benchmark and vehicle crashworthiness problem instances are provided in Section 4 and 5, respectively. Finally, the conclusions are presented in Section 6.

2. Move Acceptance Methods

The choice of heuristic selection and move acceptance methods in selection hyper-heuristics influences the performance of a hyper-heuristic [14]. A move acceptance criterion can be *deterministic* or *non-deterministic*. A deterministic move acceptance criterion produces the same result given the same initial solutions. A non-deterministic move acceptance criteria may generate a different result even when the same solutions are used. This could be because the move acceptance criterion depends on time or it might have a stochastic component while making the accept/reject decision. Examples of deterministic move acceptance criteria are All-Moves, Only-Improving and Improving & Equal. In All-Moves, the candidate solution is always accepted whether a move worsens or improves the solution quality. The candidate solution in Only-Improving criteria is accepted only if it improves the solution quality, while in Improving & Equal criteria, the candidate solution is accepted only if it improves or it is equal to the current solution. For a non-deterministic move acceptance criteria, the candidate solution is always accepted if it improves the solution quality, while the worsening solution can be accepted based on an acceptance function some of which include the great deluge algorithm [15], simulated annealing [16] and monte carlo [17].

The *choice function* (CF) is introduced as a heuristic selection method as part of a selection hyper-heuristic in Cowling et al. [18]. The choice function maintains a score for each low level heuristic and chooses the one with the highest score at each decision point during the search process. A low level heuristic's score depends on whether or not the heuristic generates improvement when used individually, when used in cooperation with another heuristic and how much time has been passed since its last invocation. This initial study has been followed by many other studies indicating the success of choice function based hyper-heuristics using different move acceptance methods on different problems. Cowling et al. [19] developed an approach using several proposed hyper-heuristic components in order to solve a real-world scheduling problem; namely project presentations. The approach employed deterministic move acceptance strategies {All-Moves, Only-Improvements} and seven heuristic selection methods {Simple Random, Random Gradient, Random Permutation, Random Permutation-Gradient, Greedy, Reinforcement Learning, Choice Function}. The experimental results show that choice function all-moves performs better than simple random moves over the given problems, and produced better solutions than those produced by humans.

There are a few comparative studies which evaluate the performances of different heuristic selection and move acceptance methods. A set of seven different heuristic selection strategies (Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Greedy, Choice Function, Tabu Search) are combined with a set of five acceptance strategies {All-Moves, Only-Improving, Improving & Equal, Exponential Monte Carlo with Counter, GDA}. The combination set is tested on fourteen benchmark functions against genetic and memetic algorithms. Choice Function-Improving & Equal performs the best [14]. Another study was conducted by Bilgin et al. [20] using a set of eight heuristic selection strategies {Simple Random, Random Gradient, Random Permutation, Random Permutation Gradient, Greedy, Choice Function, Reinforcement Learning, Tabu Search} and five move acceptance strategies {All-Moves, Only-Improving, Improving &Equal, GDA, EMCQ} which were tested on different timetabling benchmark problems. The study showed that there is no one strategy that dominates. In the scientific literature, a wide variety of hyper-heuristics have been proposed that use different heuristic selection and acceptance strategies in different domains: packing, vehicle routing, timetabling, channel assignment, component placement, personnel

scheduling, planning and shelf space allocation (see [21]). The choice function simulated annealing hyper-heuristic performed better than a simple random great deluge hyper-heuristic over a set of examination timetabling problems as presented in [20]. In [22] different heuristic selection methods {Simple Random, Greedy, Reinforcement Learning, Reinforcement, Tabu Search, Choice Function} were combined with a Late Acceptance methodology. The results show that the random heuristic selection with late acceptance obtained the best results on the examination timetabling problem.

All these previous studies focus on single-objective optimization problems. To the best of the authors' knowledge, this paper is one of the first studies that investigates the influence of the move acceptance component of selection hyper-heuristics for multi-objective optimization. In the following subsections we describe the move acceptance methods that we have used in our experiments, as well as related work.

2.1. Great Deluge

The great deluge algorithm (GDA) is a metaheuristic proposed by Dueck [15] using a threshold move acceptance criterion as illustrated in Algorithm 1. This algorithm is adopted from [15], assuming a maximization problem. GDA always accepts improving moves, while a worsening move is accepted only if it is better than a *threshold* (target improvement denoted as $LEVEL$) at a given step. The algorithm starts with an initial water level, which is often taken as the quality of the initial solution (step 4). At each step, a solution in hand (s) is modified and a new solutions ($s*$) is obtained from its neighbourhood using a move operator (step 6). The water level is increased gradually (usually linearly) at each iteration, during the search, according to a predefined rate referred to as *Rain Speed* (UP). A worsening solution ($s*$) is accepted if the quality of the solution (measured by ($f(s*)$) is greater than or equal to the water level (steps 7 and 8) and then the water level is updated (step 9). The algorithm terminates when there is no change in the solution quality within a predefined time or when the maximum number of iterations is exceeded.

The main advantage of GDA is that it is simple and easier to implement when compared to many other metaheuristics, such as, simulated annealing [16] and tabu search (TS) [23]. Moreover, better quality solutions could be produced with an increased search time [24]. GDA requires fewer input parameters; in fact it only has one parameter, rain speed (UP)

Algorithm 1 Great Deluge Algorithm

procedure GDA
  Produce an initial solution s
  Choose an initial rain speed $UP > 0$
  Choose an initial water level $LEVEL > 0$ ▷ $LEVEL = f(s)$
  repeat
    Obtain a new solution $s* \in N(s, Q)$ from s using a move operator Q
    if $(f(s*) > LEVEL)$ **then**
      $s = s*$
      $LEVEL = LEVEL + UP$
    end if
  until (termination criteria are satisfied)
end procedure

[25]. The value of UP is usually a small value greater than 0, and less than 0.03 [26]. Dueck [15] provided various recommendations regarding UP. For example, a suggestion is that UP should be on average smaller than 1% of the average distance between the quality of the current solution and the water level. So the water level can be calculated for a solution $s*$ using:

$$LEVEL = LEVEL + UP(LEVEL + f(s*)) \quad (1)$$

The value of UP can also be calculated based on the time allocated for the search and by defining upper/lower bounds of an estimated quality of solution [27]. However, both of those parameters depend on the problem dimensions and can affect the quality of final solution for a given problem [28]. An extended GDA with reheating was proposed by McMullan and McCollum [29]. The idea is similar to the reheating scheme utilized in simulated annealing. The reheating (re-levelling in the GDA context) aims to widen the boundary condition, via improving the rain speed, in order to allow a worsening move to be accepted and avoid becoming trapped in a local optimum. If there is no improvement, water level is reset and re-levelling strategy is applied using a new rain speed value based on the number of total moves in the process.

GDA has been used in many hyper-heuristic approaches as an acceptance move strategy. Özcan et al. [30] proposed a reinforcement learning great deluge hyper-heuristic. It was applied to examination timetabling, producing good quality solutions when compared to some other approaches

in the literature. Kendall and Mohamad [31] presented a variant of a GDA based hyper-heuristic. It was applied to channel assignment benchmarks. The experimental results show that a Simple Random GDA hyper-heuristic produced good results when compared to a constructive heuristic and a genetic algorithm. In addition, a variant of the GDA hyper-heuristic approach including flex deluge, non-linear and extended great deluge is proposed in [32]. These approaches were applied to large scale and highly constrained timetabling problems and tested on exam timetabling benchmark problems. The experimental analysis demonstrates that non-linear great deluge produced the best results compared to other approaches.

In the scientific literature, there are many other studies that investigate GDA and its variants in tackling various optimization problems. However, the majority of them are applied to optimization problems with a single-objective. In fact, there is only one study that proposed the GDA for multi-objective optimization [33] in which weighted sum of the objectives is used for multi-criteria examination timetabling. GDA guides the search process via a trajectory, determined by adaptively changing weights. In this study, we employ a different method rather than reducing the multiple objectives into a single objective.

2.2. Late Acceptance

Late acceptance local search (LALS) is an iterative method, proposed recently by Burke and Bykov [34]. This approach won an international competition to automatically solve the Magic Square problem which was later beaten by a selection hyper-heuristic [35]. It is based on a hill-climbing framework as illustrated in Algorithm 2 (adopted from [34]) which embeds a new move acceptance strategy. The idea is to delay the comparison between the cost of the current solution and the previous solution, hence the move acceptance method is referred to as late acceptance (LA) at the core of the local search algorithm. LA is simple and easy to implement, requiring implementation of a list C of size L. Each item C_l ($0 \leq l < L$) in the list contains the cost (fitness/objective) value of a solution visited in the l^{th} previous iteration. At the beginning of the search, C list is filled by the initial cost value. During the search process, a new solution $s*$ is obtained by applying the move operator to the current solution s. The cost of the new solution $f(s*)$ is compared to the cost of a previously visited solution. This is done via the use of the list C. The last element indicating the cost of a solution visited L steps prior to the current step (i) is maintained

at C_l. Hence, C_l is compared to the cost of the new solution $f(s*)$ for the accept/reject decision. If this cost is better than or equal to the cost of the last element, then the new solution is accepted (s is set to $s*$). The cost of s is inserted into the beginning of the list, while the last element is removed from the list. This process is repeated until a set of stopping criteria is met. In order to avoid shifting the whole list at each iteration and reduce the processing time of LA, a circular queue is employed as suggested and l is calculated using the following formula:

$$l = i \; mod \; L \tag{2}$$

where mod represents the remainder of integer division, i^{th} is the current iteration, L is the length of the cost list C.

Algorithm 2 Late Acceptance Local Search Algorithm

```
1:  procedure LALS
2:      Begin
3:      Produce an initial solution s
4:      Calculate cost of s using f(s)
5:      for ∀l ∈ {0, ..., L − 1} do
6:          C_l = f(s)
7:      end for
8:      Initialize the iteration counter, i = 0
9:      repeat
10:         Create a new solution s* from s using a move operator
11:         Calculate its cost by f(s*)
12:         l = i mod L
13:         if (f(s*) ≤ C_l or f(s*) ≤ f(s)) then
14:             Accept candidate (s = s*)
15:         end if
16:         C_l = f(s)
17:         i = i + 1
18:     until (a chosen stopping condition)
19: end procedure
```

Intuitively, in order to be able to exploit the unique features of LA, L should be set to a value less than the number of iterations and greater than or equal to two. If L is equal to one, LALS becomes a greedy hill-climber [34]. If it is equal to the number of iterations, the search could turn into

random walk depending on the move operator. Since LA is a fairly new methodology, there is only a limited number of studies in the scientific literature and none of those previous studies has dealt with a multi-objective optimization problem. In [22], the late acceptance strategy was combined with different heuristic selection methods (Simple Random, Greedy, Reinforcement Learning, Tabu Search and Choice Function) and applied to examination timetabling problems. The experiments show that the random heuristic selection with late acceptance performs well among other combination methods. In [36], an experimental comparison of LALS with well-known search methods (simulated annealing (SA), threshold accepting (TA) and GDA were carried out on the traveling salesman and exam timetabling problems. The results show the success of LALS when its performance is compared to the others.

3. Proposed Multi-objective Selection Hyper-heuristic Approach

The interest in selection hyper-heuristics has been growing in recent years. However, the majority of research in this area has been limited to single-objective optimization. To date, only a limited number of studies have been identified that address/deal with selection hyper-heuristics for multi-objective problems. Section 3.1 discusses some of those selection hyper-heuristics. Section 3.2 describes the proposed choice function based hyper-heuristic which embeds non-deterministic acceptance methods.

3.1. Related Work

A multi-objective hyper-heuristic based on tabu search was proposed in [37]. The key feature of this study 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 [38] 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 the problem in a flexible way to drive a subset of the population to the desired Pareto front. This approach was evaluated on multi-objective travelling salesman problems using eleven low level heuristics. It was compared to other multi-objective approaches from the literature, revealing that the proposed approach generates good quality results but that

future work is still needed to improve the methodology. A Markov chain based selection hyper-heuristic (MCHH) has been investigated in [39]. The Markov chain guides the selection of heuristics and applies online reinforcement leaning to adapt transition weights between heuristics. MCHH was applied to the DTLZ test problems [40] and compared to a (1+1) evolution strategy, a random hyper-heuristic and a hyper-heuristic presented in [37]. The comparison shows the efficacy of the proposed approach in terms of Pareto convergence and its ability to select good heuristic combinations. MCHH was applied to the Walking Fish Group (WFG) test problems [12]. The experiments show the efficacy of the method but future work is still needed in terms of acceptance strategies to improve the search [39]. MCHH has also been applied to real-world water distribution network design problems and has produced competitive results [41]. In [42], a new hyper-heuristic based on the multi-objective evolutionary algorithm NSGAII [8] is proposed. 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 instances of irregular 2D cutting stock benchmark problems and produced promising results. In [43],[44], a hyper-heuristic-based encoding was 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 hyper-heuristic outperforms individual heuristics. In [45] a multi-objective hyper-heuristic for the design and optimization of a stacked neural network is proposed. The proposed approach is based on NSGAII combined with a local search algorithm (Quasi-Newton algorithm). In [46] a multi-objective hyper-heuristic optimization scheme for engineering system design problems is presented. A genetic algorithm, simulated annealing and particle swarm optimization are used as low-level heuristics.

In [47], a multi-indicator hyper-heuristic for multi-objective optimization is proposed. This approach is based on multiple rank indicators, taken from NSGAII [8], IBEA [48] and SPEA2 [9]. In [49] a multiple neighbourhood hyper-heuristic for two-dimensional shelf space allocation problem is proposed. The proposed hyper-heuristic was based on a simulated annealing algorithm. In [50], a multi-objective hyper-heuristic genetic algorithm (MHypGA) for the solution of Multi-objective Software Module Clustering Problem is presented. In MHypGA, different selection, crossover and mutation operators as the components of a genetic algorithm are uti-

lized as low-level heuristics. A hypervolume-based hyper-heuristic for a dynamic-mapped multi-objective island-based model is proposed in [51]. The proposed method is superior when compared to the contribution-based hyper-heuristic and other standard parallel models over the WFG test problems [12].

Different frameworks have been proposed for mixing a set of existing algorithms. As an example, adaptive multi-method search called AMALGAM is proposed in [52, 53, 54]. This approach employs multiple search algorithms; (NSGAII [8], particle swarm optimization [55], adaptive Metropolis algorithm [56], and differential evolution [57]), simultaneously, based on adaptive offspring creation. AMALGAM is applied to a set of well known multi-objective test problems, and its performance was reported to be superior to other methods [52]. It was also applied to solve a number of water resource problems, yielding high quality solutions [53],[54]. A multi-strategy ensemble multi-objective evolutionary algorithm called MS-MOEA for dynamic optimization is proposed in [58]. It combines different strategies including a memory strategy and genetic and differential operators to adaptively create offspring and achieve fast convergence speed. The experimental results show that MS-MOEA is able to obtain promising results.

Maashi et al. [11] investigated an online learning selection hyper-heuristic, and its hybridization with multi-objective evolutionary algorithms. A choice function all-moves hyper-heuristic for multi-objective optimization problems (HHMO_CF_AM) was proposed in their study. The choice function variant acts as a multi-objective heuristic selection mechanism as part of the high level search strategy. The choice function method adaptively ranks the performance of the low level heuristics and chooses the top ranking heuristic at each decision point. Three multi-objective evolutionary algorithms (NSGAII [8], SPEA2 [9], and MOGA [10]) act as low level heuristics. A ranking scheme is used to rank each low level heuristic against four performance measurements (AE [59], RNI [59], SSC [4], and UD [60]). All-Moves are accepted regardless whether a move improves the quality of the solution or not. The experimental results over the WFG problems [12] and a real world problem [13] show that this approach can benefit from the strengths of the low level heuristics. In the majority of cases, HHMO_CF_AM outperforms the other multi-objective evolutionary algorithms (NSGAII, SPEA2 and MOGA), when used in isolation. Moreover, HHMO_CF_AM is superior to AMALGAM [52] and a random hyper-

heuristic [11]. This study extends our previous work in [11] by introducing a metric enabling the use of non-deterministic move acceptance methods within a selection hyper-heuristic framework.

3.2. Choice Function Multi-objective Hyper-heuristic based on Non-deterministic Move Acceptance

Previous studies show that the use of a different move acceptance method could improve the overall performance of a selection hyper-heuristic in the context of single objective optimization [14]. In this study, we investigate the behavior of the great deluge algorithm (GDA) [15] and late acceptance (LA) [5] as non-deterministic move acceptance strategies within the choice function based hyper-heuristic framework, designed for solving multi-objective optimization problems. We attempt to improve the performance of the original approach (HHMO_CF_AM) [11] by using GDA and LA as an acceptance method instead of All-moves and adopting those methods for multiobjective optimization, allowing acceptance of "worsening" moves at a given step.

The motivation for choosing GDA and LA as an acceptance method is that both are simple and do not require many parameters, requiring less effort in parameter tuning. More importantly, encouraging results have been reported in the literature for single-objective optimization, but there are only a few studies on their application to multi-objective optimization (e.g., [33]). However, GDA and LA are different in the way that they accept worse solutions. GDA uses a time (iteration) varying threshold while accepting a worse solution, while LA uses a delayed acceptance strategy comparing the quality of a current candidate solution to that of a solution visited a number of iterations earlier. Both move acceptance methods require computation of the change in the value of a single-objective at each step and so the D performance metric [4] is proposed for their applicability to the multi-objective optimization problems.

3.2.1. D metric

The D metric [4] is an extended version of the hypervolume, and is also known as the *size of space covered* metric (SSC) [4]. The SSC metric evaluates how much of the objective space is covered by a given front. Hence, SSC cannot be directly used to decide whether a given front dominates another one or not. However, the D metric can compute the space (surface) coverage difference between two non-dominated sets (e.g., initial/current

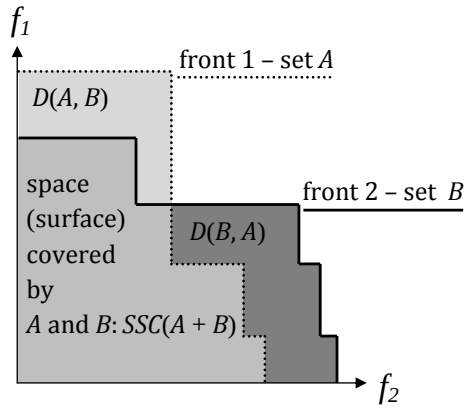

Figure 1: Ilustration of how D metric works for two given fronts 1 (set A) and 2 (set B) [61].

non-dominated set) A and (e.g., new non-dominated set) B with respect to the objective space as illustrated in Fig. 1. $(A + B)$ denotes the union of surfaces covered by A and B. $D(A, B)$ denotes the size of the space dominated by A and not dominated by B while $D(B, A)$ denotes the size of the space dominated by B and not dominated by A:

$$D(A, B) = SSC(A + B) - SSC(B) \tag{3}$$

$$D(B, A) = SSC(A + B) - SSC(A) \tag{4}$$

If $D(A, B) < D(B, A)$ then B dominates A. In other words, the non-dominated set B (front 2) is better than the non-dominated set A (front 1) with respect to the D metric.

In the context of move acceptance criterion, in particular GDA and LA, the quality measure of the current solution and the candidate solution is essential in order to make a decision regarding an acceptance decision. For the single-objective optimization problem, fitness can be used. However, this is not applicable in a multi-objective optimization. In multi-objective problems, the output is a set of solutions (a non-dominated set). We propose the use of the D metric as a way of comparing two non-dominated sets with respect to the objective space. The D metric is usually used in the literature as a performance metric to compare the final solutions ob-

tained from multi-objective optimizers. In this study, for the first time, we integrate the D metric into a move acceptance criterion in order to convert multi-objective optimization to single optimization without having to define weights for each criteria.

3.2.2. Great Deluge and D Metric

GDA is modified based on the D metric to enable its use in multi-objective optimization and comparison of a new front (B) to the current front in hand (A) for acceptance as shown in Algorithm 3. A is a non-dominated front which represents an initial or current solution and B is a front which represents a new solution potentially obtained from the neighborhood. The water level is assigned initially to the value of $D(A, B)$. $D(A, B)$ value will always be less than or equal to $LEVEL$ at any point during the search process. Hence, any improvement (non-dominated front) will be accepted after checking with respect to the inequality of $D(B, A) > LEVEL$. This also allows acceptance of a new front which is worse than (dominated by) A in terms of the D measure. When B is accepted and the water level is increased linearly using a predefined speed rate (UP). This rate is usually fixed as a small positive value in single objective optimization and the same strategy is used for multi-objective optimization.

Algorithm 3 Great Deluge Algorithm with D metric

```
1: procedure GDA(A, B)
2:     if (D(B, A) > LEVEL) then
3:         A = B
4:         LEVEL = LEVEL + UP
5:     end if
6: end procedure
```

3.2.3. Late Acceptance and D Metric

LA is modified to employ the D metric and its pseudo-code is shown in Algorithm 4. LA utilizes a list of length L and this list C contains values obtained using the D metric at each entry C_l, where $0 \leq l < L$. Given two fronts (sets of solutions) A and B representing the current front and new front, respectively, $D(A, B)$ and $D(B, A)$ are measured. The new front is accepted if $D(B, A) > D(A, B)$ (see subsection 3.2.1), since this situation indicates an "improving" front (step 3). If not, B is still accepted if the

value of $D(B, A)$ is better than or equal to the value of C_l which is the D measure of two fronts from L step prior to the current step i (see step 2); $D(B, A) \geq C_l$. After B gets accepted (step 4), the value of $D(B, A)$ is inserted into C_l (step 7) and the old value is removed. If a move is rejected, then C_l is set to the D value of the pair of fronts when A was accepted previously. C_l values are initialized to $D(X, Y)$ when a new front Y is obtained from the first front X at the beginning of the search process.

Algorithm 4 The Late Acceptance with D metric

1: **procedure** LA(A, B) ▷ Given that $dval$ holds the D value of previous two fronts when A was accepted
2: $l = i \; mod \; L$; $i = i + 1$; ▷ i is the current iteration
3: **if** ($D(B, A) \geq C_l$ or $D(B, A) > D(A, B)$) **then**
4: Accept candidate $A = B$
5: $dval = D(B, A)$
6: **end if**
7: Insert D value into the list, $C_l = dval$
8: **end procedure**

3.2.4. Choice Function great deluge/late acceptance based Hyper-heuristic Framework

Maashi et al. [11] introduced a selection hyper-heuristic framework based on a choice function which is modified to deal with multi-objective optimization problems and a mechanism to rank low level heuristics for heuristic selection using equation 5.

$$CF(h) = \alpha f_1(h) + f_2(h) \tag{5}$$

Equation 5 reflects the overall performance of low level heuristics with respect to the performance metrics (AE, RNI, SSC and UD). This measures the resulting non-dominated set in the objective space. f_1 represents the performance of an individual heuristic h based on the highest ranking among the other heuristics. f_1 is used for intensification. f_2 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. α is a large positive value (e.g. 100). It is important to strike a balance between f_1 and f_2 values, so that they are in the same scalar unit.

The pseudo code of the two proposed hyper-heuristics for multi-objective optimization; choice function great deluge based hyper-heuristic (HHMO_CF_GDA) and choice function late acceptance based hyper-heuristic (HHMO_CF_LA) shown in Algorithm 5. Initially, a greedy algorithm is applied to determine the best low level heuristic h to be selected for the first iteration (steps 2-6). All low level heuristics H are executed (step 3). Then, the low level heuristics are ranked based on the ranking scheme (step 4) and their choice function values are computed using equation 5 (step 5). The low level heuristic h with the largest choice function value $CF(h)$ is selected to be applied at the next iteration and it produces the non-dominated front A (a current solution) (steps 6 and 7). Then, for all low level heuristics H, 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 h with the largest choice function value $CF(h)$ is executed and it produces the non-dominated front B (a candidate solution) (steps 11 and 12). In step 14, the acceptance procedure, whether GDA(A,B) or LA(A,B) depends on the hyper-heuristic that we used, is called and applied using the parameters that were obtained from the search (see Sections 3.2.2 and 3.2.3). This process is repeated until the stopping condition is met which is a fixed number of iterations (steps 8-16). Please note the acceptance procedure is applied on the indicator value of the whole candidate population not on the single solution of the population. And 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.

4. Computational Experiments on the Walking Fish Group (WFG) Test Problems

The experiments are conducted over the Walking Fish Group (WFG) benchmark dataset [12] to (i) investigate the influence of using non-deterministic move acceptance strategies; great deluge algorithm and late acceptance on the performance of online learning choice function based selection hyper-heuristic for multi-objective optimization (denoted as HHMO_CF_GDA and HHMO_CF_LA), and (ii) to compare their performance to the original approach; selection hyper-heuristic based on deterministic move acceptance (HHMO_CF_AM) [11].

Algorithm 5 Multi-Objective Hyper-heuristic based on Acceptance Procedure

procedure HH_CF(H) where H is a set of the low level heuristics.
    Initialization
    Run h, $\forall\, h \in H$
    Rank h, $\forall\, h \in H$ based on the ranking scheme
    Get $CF(h)$, $\forall\, h \in H$
    Select h with the largest $CF(h)$ as an initial heuristic
    Execute the selected h and produce a front A
    repeat
        Update the rank of h, $\forall\, h \in H$ based on the ranking scheme
        Update $CF(h)$, $\forall\, h \in H$
        Select h with the largest $CF(h)$, $\forall\, h \in H$
        Execute the selected h and produce a new front B
        Call the acceptance procedure GDA(A,B)/LA(A,B)
    until (termination criteria are satisfied)
end procedure

4.1. WFG Test Problems

The WFG test suit was introduced by Huband et al. [12]. It consists of nine continuous benchmark functions as shown in Table 1. The WFG test suite is designed for real valued optimization with no side constraints which makes it easier to analyze and implement. Due to the distinct features of the problems in the WFG dataset, it is a common choice for most MO/EA researchers [12] for assessing new algorithms.

Unlike most of the multi-objective test suites such as ZDT [6] and DTLZ [40], the WFG test suite has powerful functionality. It also has a number of features that the other test suites do not include. 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 objectives. The numbers of distance and position related parameters can be scaled independently. The WFG test problems 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 offer a more comprehensive set of challenges when compared to DTLZ using NSGAII [12].

4.2. 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 toward the Pareto optimal front (POF). We use five performance metrics to assess the quality of approximation sets from different aspects: (i) ratio of non-dominated individuals (RNI) [59], (ii) hypervolume (SSC) [4], (iii) uniform distribution of a non-dominated population (UD) [60], (iv) generational distance (GD) [62] and (v) the inverted generational distance (IGD) [7] . A higher value considering one of those performance metrics indicates that non-dominated solutions are of good quality, except for GD and IGD, where a lower value indicates that the approximation non-dominated front is closer to the POF.

We have compared the mean performance of three choice function based multi-objective hyper-heuristics; HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA across multiple trials with respect to the metrics across multiple trials. Moreover, *t*-tests are used as a statistical test for pairwise mean performance comparison. The following notation is used when reporting the results. Given a pair of algorithms, A and B (denoted as A-B), The $+$ $(-)$ indicates that algorithm A performs better (worse) than B on average with respect to the given metric and this performance difference is statistically significant within a confidence interval of 95%. $\pm$ $(\mp)$ indicates that A performs slightly better (worse) than B without any statistical significance. $\approx$ indicates that mean performances of both algorithms are equal.

4.3. Experimental Settings

All experimental parameters are chosen based on those commonly used in the literature for continuous problems (see [6] and [12]). We use the same parameter settings that were used for HHMO_CF_AM in [11] for a fair comparison. NSGAII[8], SPEA2[9], and MOGA[10] act as the low level heuristics within the multi-objective choice function based hyper-heuristics. The nine test problems (WFG1-WFG9) have 24 real parameters

including four position parameters, 20 distance parameters and two objectives. We concur with findings in the literature [6] that two objectives are enough to represent the essential features of multi-objective optimization problems to demonstrate the significance of the proposed approach.

Following the recommendation in [63] and [64], a hyper-heuristic is terminated after 6,250 generations. All hyper-heuristics are run for a total of 25 stages. In each stage, a low level heuristic is chosen and executed for 250 generations with a population size of 100. The secondary population of SPEA2 is set to 100. The execution time takes 10 to 30 minutes depending on a given problem. For the WFG problems, 30 independent trials were run for each method with different random seeds. For all three low level heuristics, the simulated binary crossover (SBX) operator is used for recombination and a polynomial distribution for mutation [65]. 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 and D metric for GDA and LA, the reference points for WFG problems with k objectives was set $r_i = (0, i * 2)$, $i = 1, ..., k$ [12] . The distance sharing σ 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 the hyper-heuristic framework and as a performance measure for the comparison. As for HHMO_CF_GDA, the rain speed (UP) is set to 0.0003 based on the empirical experiments that are presented in the supplementary files. The length of the fitness array (L_{Fa}) in HHMO_CF_LA is set to 5.00 as recommended in [36]. All methods are implemented using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz\2G\250G computer.

4.4. Results

The average values associated with their standard deviations considering the performance metrics, including RNI, SSC, UD, GD and IGD for each WFG problem generated by each hyper-heuristic across 30 trials are provided in Table 2. The pairwise (*t*-test) mean performance comparison of different choice function based selection hyper-heuristics each using a different move acceptance method are provided in Table 3. Overall, HHMO_CF_GDA performs the best. The experimental result show that the selection hyper-heuristic using GDA and LA performs better than the one using AM on average with respect to the ratio of non-dominated solutions (RNI). From this point onward, each hyper-heuristic will be referred

to by its move acceptance method utilized within each hyper-heuristic. The pairwise performance differences of GDA and LA from AM are statistically significant with respect to the measure of RNI for all benchmark functions, except WFG1. GDA and LA perform relatively similar with respect to RNI. With respect to the measure of the hypervolume (SSC), GDA has the best overall mean performance when compared to AM and LA, and this performance difference is statistically significant across all WFG problems, except WFG2. For this instance, GDA performs slightly better than LA. In addition, LA delivers a significantly better performance than AM for all WFG problems, except WFG5. Similarly, GDA delivers a significantly better mean performance when compared to AM and LA with respect to the measure of generational distance (GD) for all benchmark functions, except WFG1 and WFG9. For WFG1, AM performs slightly better than GDA and significantly better than LA, while for WFG9, LA performs significantly better than AM and GDA performs slightly better than AM. With respect to the measure of inverted generational distance (IGD), GDA performs significantly better than AM in all instances except in WFG1. In addition, GDA performs significantly better than LA in four instances of WFG2, WFG4, WFG8 and WFG9 while it performs significantly similar to LA in the rest.

Although non-deterministic move acceptance methods improve the overall mean performance of the hyper-heuristic with respect to RNI, SSC, GD and IGD, AM performs the best with respect to the measure of the uniform distribution of non-dominated solutions (UD). The performance differences from GDA and LA are statistically significant for all problems, except WFG4, for which AM still performs slightly better than LA. GDA and LA have relatively similar performance across all WFG problems except WFG5, WFG8 and WFG9. The success of AM with respect to UD might be due to the use of the D metric within the acceptance procedure. Since D metric is a binary hypervolume measure that is designed to compare two sets of non-dominated solutions with respect of their convergence towards the POF, there is no consideration regarding how uniformly these solutions are distributed along the POF. This might also be a reason why non-deterministic move acceptance produces high quality solutions in terms of the convergence towards the POF.

To further understand how the move acceptance strategies, AM, GDA and LA, are performing and how their performances could affect the quality of the solutions, Fig. 2 illustrates the average number of heuristic invo-

cations of each low level heuristic selected and applied at 25 consecutive decision points during the whole search process over all runs. Each bar in the plot also indicates the average number of accepted and rejected Pareto fronts. A similar pattern for the choice of low level heuristics during the search process has been observed in Fig. 2 on almost all WFG problems considering the three hyper-heuristics. This is highly likely due to the use of the same heuristic selection mechanism (choice function). However, the pattern in the plots for accepted or rejected Pareto fronts produced by the chosen low level heuristic varies for a given problem depending on the move acceptance strategy that the hyper-heuristic employed. NSGAII is always selected more than the other low level heuristics regardless of the move acceptance method, except for WFG5 and WFG9. For WFG5, SPEA2 is the most frequently chosen algorithm regardless of the move acceptance component of the hyper-heuristic during the search process. On the other hand, SPEA2 is frequently chosen when GDA is used as the move acceptance on WFG9. The performance of MOGA is the worst among the three hyper-heuristics on the WFG problems; thus it is invoked relatively less frequently during the search process in all test problems for all methods. NSGAII appears to be a good choice for solving the WFG problems. Our observations are consistent with the result obtained in [66] showing that the best performance is achieved by NSGAII on the WFG test problems with two objectives. As discussed in [11], mixing different metaheuristics under a selection hyper-heuristic framework yields an improved performance.

Fig. 2 shows that there is only one case in which all moves are accepted when a non-deterministic strategy is used, that is GDA for WFG1. The rate of moves rejected for LA is higher than that for GDA on all test problems regardless of the low level metaheuristic employed, except for MOGA, where LA accepts more moves (solutions) than GDA on almost all problems. These observations offer some explanation why the performance of GDA is better than LA in terms of convergence towards the POF: (i) The good moves that are accepted in GDA are rejected in LA, and (ii) as MOGA does not perform well in the WFG test problem and it is invoked relatively less frequently during the search process, LA accepts all MOGA's moves (solutions) while GDA rejects them. LA produces better solutions than AM. So, the non-deterministic acceptance strategies (GDA and LA) beat the deterministic acceptance strategy (AM). In addition, GDA and LA appear to positively affect the performance of the multi-objective choice

function based hyper-heuristic when used as the move acceptance strategy over the WFG test problems.

5. Computational Experiments on the Multi-objective Design of Vehicle Crashworthiness

More experiments are conducted over a multi-objective real-world problem, namely the design of vehicle crashworthiness problem [13], to evaluate the performance of selection hyper-heuristics. The same performance evaluation criteria and algorithms are used as described in the previous section. In this study, the performances of HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA is compared to the NSGAII [8].

5.1. Problem Formulation

In the automotive industry, crashworthiness is a very important issue when designing a vehicle. Crashworthiness design of real-world vehicles involves the optimization of a number of objectives including the head, injury criterion, chest acceleration, chest deflection, etc. However, some of these objectives may be, and usually are, in conflict with each other, i.e. an improvement in one objective value leads to the deterioration in the values of the other objectives.

Liao et al. [13] presented a multi-objective design for the vehicle crashworthiness problem with five real-value parameters and no constraints. The vehicle crashworthiness problem was constructed using the surrogate modelling techniques with latin hypercube sampling and stepwise regression considering the weight, acceleration characteristics and toe-board intrusion as the design objectives. The mass of the vehicle is tackled as the first design objective, while an integration of collision acceleration between $t_1 = 0.05s$ and $t_2 = 0.07s$ in the full frontal crash is considered as the second objective function. The toe-board intrusion in the 40% offset-frontal crash is tackled as the third objective. The second and third objectives (ain and intrusion, respectively) are constructed from the two crash conditions to reflect the extreme crashworthiness and formulated in the quadratic basis functions while the vehicle mass is formulated in a linear function as follows:

$$\begin{aligned}\text{Mass} &= 1640.2823 + 2.3573285t_1 + 2.3220035t_2 \\ &+ 4.5688768t_3 + 7.7213633t_4 + 4.4559504t_5\end{aligned} \tag{6}$$

$$\begin{aligned}\text{Ain} &= 6.5856 + 1.15t_1 - 1.0427t_2 + 0.9738t_3 + 0.8364t_4 \\ &- 0.3695t_1t_4 + 0.0861t_1t_5 + 0.3628t_2t_4 - 0.1106t_1^2 \\ &- 0.3437t_3^2 + 0.1764t_4^2\end{aligned} \tag{7}$$

$$\begin{aligned}\text{Intrusion} &= -0.0551 + 0.0181t_1 + 0.1024t_2 + 0.0421t_3 \\ &- 0.0073t_1t_2 + 0.024t_2t_3 - 0.0118t_2t_4 - 0.0204t_3t_4 \\ &- 0.008t_3t_5 - 0.0241t_2^2 + 0.0109t_4^2\end{aligned} \tag{8}$$

So, the multi-objective design of vehicle crashworthiness problem is formulated as:

$$\begin{aligned}&\text{min F(x)= } [Mass, Ain, Intrusion] \\ &\quad \text{s.t.} \\ &\quad\quad 1mm \leq x \leq 3mm \\ &\text{where x=} (t_1, t_2, t_3, t_4, t_5)^T\end{aligned} \tag{9}$$

The motivation behind applying our three hyper-heuristics multi-objective choice function based approaches to this problem is to see their performance on a real-world problem and measure the level of generality they can achieve. We created three more problem instances beside the original vehicle crashworthiness problem as shown in Table 4 after a private communication with Professor Kalyanmoy Deb. Each instance contains a pair of objectives. NSGAII was applied to the original vehicle crashworthiness problem in [13] and produced reasonable results for the three objective version.

5.2. Performance Evaluation Criteria and Experimental Settings

We used five performance metrics to assess the quality of approximation sets from different aspects: (i) ratio of non-dominated individuals (RNI) [59], (ii) hypervolume (SSC) [4], (iii) uniform distribution of a non-dominated population (UD) [60], (iv) generational distance (GD) [62] and (v) the inverted generational distance (IGD) [7]. *t*-tests are used as a statistical test for pairwise mean performance comparison of methods using the same notations as those used in Section 4.2.

We performed 30 independent runs for each comparison method using the same parameter settings as provided in [13] with a population size of 30 and running for 50 generations in each iteration. All hyper-heuristic methodologies were run for a total of 75 iterations based on the empirical experiments that are presented in the supplementary files. In order to make a fair comparison, we repeated NSGAII experiments conducted in [13] under our termination conditions over the additional instances. So, all methods terminated after 3,750 generations. The distance sharing σ for the UD metric and MOGA was arbitrarily set to 0.09 in the normalized space. These settings were used for UD as a feedback indicator in the ranking scheme of the hyper-heuristic framework and as a performance measure for the comparison. As the true Pareto front is unknown, we consider the best approximation found by means of combining results of all considered methods and used it instead of a true Pareto front for the metrics of GD and IGD. In the measure of SSC and the D metric for GDA and LA, the reference points in our experiments for k objectives can be set as $r_i = z^{nadir_i} + 0.5(z^{nadir_i} - z^{ideal_i})(0, i*2)$, $i = 1, ..., k$ [67]. Other experimental settings are the same as those used in Section 4.3. All algorithms were implemented with the same common sub-functions using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz/2G/250G computer.

5.3. Performance Comparison of Selection Hyper-heuristics and NSGAII

The mean performance comparison of AM, GDA, LA and NSGAII based on the performance metrics (RNI, SSC, UD, GD and IGD) for solving the vehicle crashworthiness problems is provided in Table 5. Table 5 and the result of the *t*-test in Table 6 show that GDA, LA and NSGAII produce a slightly higher average ratio of non-dominated individuals (RNI) compared to AM for all problems. This means that the comparison methods produce non-dominated solutions that are equal to the given population size and perform very well with respect to this metric. AM performs well

with respect to RNI on Car4, but not for other problem instances. With respect to the hypervolume (SSC), GDA has the highest average value among the other methods for all problem instances. The performance difference of GDA from the other hyper-heuristics is statistically significant for Car1, Car3 and Car4. With respect to the measures of GD and IGD, GDA is superior to the other methods for all problem instances, except Car3, where NSGAII performs the best. This performance difference is statistically significant for Car1, Car2 and Car4. Considering UD, GDA produces solutions that are distributed uniformly along the POF for all problem instances, except Car2, where NSGAII performs the best.

In summary, GDA performs the best considering convergence and diversity, producing solutions that converge towards the POF and that are distributed uniformly along the POF.

Figs. 3 and 4 illustrates the 50% attainment surfaces from the 30 fronts after 3,750 generations produced by each method for each problem instance. GDA appears to generate good convergence for all problem instances. This can be clearly observed for Car2 and Car3, where GDA converges to the best POF with a well spread optimal Pareto front as compared to the other approaches. In contrast, AM generates the poorest solutions in almost all cases. NSGAII and LA have similar convergence for all problem instances, except Car2, where NSGAII has covered a larger proportion of objective space compared to LA. From the above observations, we conclude that GDA outperforms NSGAII and other methods in the majority of cases. The hyper-heuristics for even real world multi-objective problems benefits from the use of a learning heuristic selection method as well as GDA.

6. Conclusion

Studies on selection hyper-heuristics for multi-objective optimization are limited. This is one of the first studies which investigates the influence of the move acceptance component of a selection hyper-heuristic for multi-objective optimization. Two choice function based online learning selection hyper-heuristics are introduced, each embedding a different non-deterministic move acceptance criteria; great deluge algorithm (GDA) and late acceptance (LA) which are adopted for multi-objective optimization using the well-known D metric. The proposed selection hyper-heuristic

framework is highly flexible and its components are reusable. It is built on an interface which allows other researchers to write their own hyper-heuristic components easily. If new and better performing components are found in the future, the software can be easily modified to include those components for testing.

The performance of the two proposed hyper-heuristics, namely HHMO_CF_GDA and HHMO_CF_LA are compared to a previous approach HHMO_CF_AM [11] on the Walking Fish Group test problems (WFG) which is a common benchmark for multi-objective optimization. Additionally, their performance is compared to the well-known multi-objective algorithm, NSGAII on the real-world vehicle crashworthiness multi-objective design problem instances. The experimental results demonstrate the effectiveness of non-deterministic move acceptance strategy based methodology. HHMO_CF_GDA and HHMO_CF_LA outperforms the previously proposed approach of HHMO_CF_AM [11] over the WFG test problems, indicating that the non-deterministic acceptance strategies improve the performance of the multi-objective choice function based selection hyper-heuristic. Moreover, this observation is supported further by empirical evidence obtained from testing those hyper-heuristics against NSGAII over the vehicle crashworthiness problems. HHMO_CF_GDA turns out to be the best choice for solving this problem.

The results from both benchmark test problems (WFG) and the real-world problems (vehicle crashworthiness design) demonstrate the ability and potential of the multi-objective selection hyper-heuristic approaches in solving continuous multi-objective optimization problems. Investigating the performance of selection hyper-heuristics utilizing other heuristic selection or move acceptance which are adopted for multi-objective optimization is a trivial future work. Applying the proposed selection hyper-heuristics on other types of multi-objective problems, such as discrete or dynamic environment problems is also left as future work for testing the level of generality of the proposed approaches further.

Acknowledgement. The authors thank the University of Tabuk and Ministry of Higher Education in Saudi Arabia for funding this work.

References

[1] Qu, R., Burke, E.. Hybridisations within a graph based hyper-heuristic framework for university timetabling problems. Journal of

the Operational Research Society 2009;60:1273–1285.

[2] Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.. Handbook of Meta-Heuristics; chap. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. Kluwer Academic Publishers; 2003, p. 457–474.

[3] Hussin, N.. Tabu Search Based Hyper-heuristic Approaches for Examination Timetabling. Ph.D. thesis; The University of Nottingham, Nottingham, UK; 2005.

[4] Zitzler, E.. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. thesis; ETH Zurich, Switzerland; 1999.

[5] Zitzler, E., Deb, K., Thiele, L.. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 2000;8(2):173–195.

[6] Zitzler, E., Deb, K., Thiele, L.. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 1995;8(2):173195.

[7] Coello Coello, C.A., Cruz Cortès, N.. Solving multiobjective optimization problems using an artificial immune system. Genetic Programming and Evolvable Machines 2005;6(2):163–190.

[8] Deb, K., Goel, T.. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In: Proceedings of Evolutionary Multi Criterion Optimization Conference. 2001, p. 67–81.

[9] Zitzler, E., Laumanns, M., Thiele, L.. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: EUROGEN 2001 Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. 2001, p. 951–100.

[10] Fonseca, C., Fleming, P.. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. in Part I: A Unified Formulation IEEE transactions systems man and cybernetics Part A: Systems and Humans 1998;28(1):26–37.

[11] Maashi, M., Özcan, E., Kendall, G.. A multi-objective hyper-heuristic based on choice function. Expert Systems with Applications 2014;41(9):4475–4493.

[12] Huband, S., Hingston, P., Barone, L., While, L.. A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation 2006;10(5).

[13] Liao, X., Li, Q., Yang, X., Zhang, W., Li, W.. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. Structural and Multidisciplinary Optimization 2008;35:561569.

[14] Özcan, E., Bilgin, B., Korkmaz, E.. A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 2008;12(1):3–23.

[15] Dueck, G.. New optimization heuristics: The great deluge algorithm and the record to record travel. Journal of Computational Physics 1993;104:86–92.

[16] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.. Optimization by simulated annealing. Science 1983;220:671–680.

[17] Glover, F., Laguna, M.. Modern heuristic techniques for combinatorial problems; chap. Tabu search. 1995, p. 70–150.

[18] Cowling, P.I., Kendall, G., Soubeiga, E.. A hyperheuristic approach to scheduling a sales summit. In: Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling III. PATAT '00; London, UK, UK: Springer-Verlag. ISBN 3-540-42421-0; 2001, p. 176–190.

[19] Cowling, P., Kendall, G., Soubeiga, E.. Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. Applications of Evolutionary Computing 2002;2279:1–10.

[20] Bilgin, B., Özcan, E., Korkmaz, E.. An experimental study on hyper-heuristics and exam timetabling. In: Burke, E., Rudov, H., editors. Practice and Theory of Automated Timetabling VI; vol. 3867 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg; 2007, p. 394–412.

[21] Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., et al. Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society 2013;64(12):1695–1724.

[22] Özcan, E., Bykov, Y., Birben, M., Burke, E.K.. Examination timetabling using late acceptance hyper-heuristics. In: the IEEE Congress on Evolutionary Computation. 2009, p. 997–1004.

[23] Glover, F.. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 1986;13(5):533–549.

[24] Burke, E., Bykov, Y., Newall, J., Petrovic, S.. A time-predefined local search approach to exam timetabling problems. IIE Transactions 2004;36(6):509–528.

[25] Petrovic, S., Yang, Y., Dror, M.. Case-based selection of initialisation heuristics for metaheuristic examination timetabling. Expert Systems with Applications 2007;33(3):772 – 785.

[26] Scott, E., Geldenhuysys, G.. A comparison of the tabu search and great deluge methods for graph colouring. In: Proceedings of the 16th IMACS World Congress Scientific Computation Applied Mathematics and Simulation (IMACS 2000). 2000, p. 363.

[27] Sanja Petrovic Yong Yang, M.D.. Case-based selection of initialisation heuristics for metaheuristic examination timetabling. Evolutionary Computation 2000;8(2):173–195.

[28] Telfar, G.. Generally applicable heuristicsfor global optimisation: An investigation of algorithm performance for the euclidean traveling salesman problem. Master's thesis; Institute of Statistics and Operations Research, Victoria University Of Wellington; 1995.

[29] McMullan, P., McCollum, B.. Dynamic job scheduling on the grid environment using the great deluge algorithm. In: Malyshkin, V., editor. Parallel Computing Technologies; vol. 4671 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. ISBN 978-3-540-73939-5; 2007, p. 283–292.

[30] Özcan, E., Misir, M., Ochoa, G., Burke, E.K.. A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. International Journal of Applied Metaheuristic Computing 2010;1(1):39–59.

[31] Kendall, G., Mohamad, M.. Channel assignment in cellular communication using a great deluge hyperheuristic. In: IEEE International Conference on Network. 2004, p. 769–773.

[32] Sin, E.S., Kham, N.S.M.. Hyper heuristic based on great deluge and its variants for exam timetabling problem. CoRR 2012;abs/1202.1891.

[33] Petrovic, S., Bykov, Y.. A multiobjective optimisation technique for exam timetabling based on trajectories. In: Burke, E., De Causmaecker, P., editors. Practice and Theory of Automated Timetabling IV; vol. 2740 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg; 2003, p. 181–194.

[34] Burke, E.K., Bykov, Y.. A late acceptance strategy in hill-climbing for exam timetabling problems. In: Proceeding of the 7th International Conference on the Practice and Theory of Automated Timetabling. 2008, p. 1–7.

[35] Kheiri, A., Özcan, E.. Constructing constrained-version of magic squares using selection hyper-heuristics. The Computer Journal 2014;57(3):469–479.

[36] Burke, E., Bykov, Y.. The late acceptance hill-climbing heuristic. Tech. Rep. Technical Report CSM-192; Computing Science and Mathematics, University of Stirling; Stirling, UK; 2012.

[37] Burke, E.K., Landa-Silva, J.D., Soubeiga, E.. Multi-objective hyper-heuristic approaches for space allocation and timetabling. In: Ibaraki, T., Nonobe, K., Yagiura, M., editors. Meta-heuristics: Progress as Real Problem Solvers. The 5th Metaheuristics International Conference (MIC 2003); Springer; 2005, p. 129–158.

[38] Veerapen, N., Landa-Silva, D., Gandibleux, X. (ORO). Hyperheuristic as component of a multi-objective metaheuristic. In: Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search

Algorithms (SLS-DS 2009). Bruxelles, Belgium; 2009,Technical Report TR/IRIDIA/2009-024, IRIDIA, Universite Libre de Bruxelles, Brussels, Belgium.

[39] McClymont, K., Keedwell, E.C.. Markov chain hyper-heuristic (mchh): An online selective hyper-heuristic for multi-objective continuous problems. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. GECCO '11; ACM. ISBN 978-1-4503-0557-0; 2011, p. 2003–2010.

[40] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.. Scalable multiobjective optimization test problems. In: Proceedings of Congress on Evolutionary Computation. 2002, p. 825–830.

[41] McClymont, K., Keedwell, E., Savic, D., Randall-Smith, M.. A general multi-objective hyper-heuristic for water distribution network design with discolouration risk. Journal of Hydroinformatics 2013;15(3):700–716.

[42] Gomez, J., Terashima-Marín, H.. Approximating multi-objective hyper-heuristics for solving 2d irregular cutting stock problems. In: Advances in Soft Computing. Lecture Notes in Computer Science; 2010, p. 349–360.

[43] Miranda, G., de Armas, J., Segura, C., León, C.. Hyperheuristic codification for the multi-objective 2d guillotine strip packing problem. In: Proceedings of IEEE Congress on Evolutionary Computation. 2010, p. 1–8.

[44] de Armas, J., Miranda, G., León, C.. Hyperheuristic encoding scheme for multi-objective guillotine cutting problems. In: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference. 2011, p. 1683–1690.

[45] Furtuna, R., Curteanu, S., Leon, F.. Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic. Applied Soft Computing 2012;12(1).

[46] Rafique, A.F.. Multiobjective hyper heuristic scheme for system design and optimization. In: Proceedings of the 9th International Con-

ference on Mathematical Problems in Engineering, Aerospace and Science, AIP Conference; vol. 1493. 2012, p. 764–769.

[47] Vázquez-Rodríguez, J., Petrovic, S.. A mixture experiments multi-objective hyper-heuristics. Journal of the Operational Research Society 2013;64(11):1664–1675.

[48] Zitzler, E., Künzli, S.. Indicator-based selection in multiobjective search. In: Lecture Notes in Computer Science. Parallel Problem Solving from Nature (PPSN VIII); Springer; 2004, p. 832842.

[49] Bai, R., van Woensel, T., Kendall, G., Burke, E.K.. A new model and a hyper-heuristic approach for two-dimensional shelf space allocation. Journal Operational Research 2013;11:31–55.

[50] Kumari, A., Srinivas, K., Gupta, M.. Software module clustering using a hyper-heuristic based multi-objective genetic algorithm. In: Advance Computing Conference (IACC), 2013 IEEE 3rd International. 2013, p. 813–818.

[51] Len, C., Miranda, G., Segura, C.. Hyperheuristics for a dynamic-mapped multi-objective island-based model. In: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living; vol. 5518 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2009, p. 41–49.

[52] Vrugt, J., Robinson, B.. Improved evolutionary optimization from genetically adaptive multimethod search. Proceedings of the National Academy of Sciences 2007;104(3):708–711.

[53] Raad, D., Sinkse, A., Vuuren, J.. Multiobjective optimization for water distribution systemdesign using a hyperheuristic. Journal of Water Resources Management 2000;136(5):592–596.

[54] Zhang, X., Srinivasan, R., Liew, M.V.. On the use of multi-algorithm, genetically adaptive multi-objective method for multi-site calibration of the swat model. Hydrological Processes 2010;24(8):955–1094.

[55] Kennedy, J., Eberhart, R., Shi, Y.. Swarm Intelligence. Morgan Kaufmann; 2001.

[56] Haario, H., Saksman, E., Tamminen, J.. An adaptive metropolis algorithm. Bernoulli 2001;7:223–242.

[57] Storn, R., Price, K.. Differential evolution: A simple and efficient heuristic for global optimization over continuous. Journal of Global Optimization 1997;11:341–359.

[58] Wang, Y., Li, B.. Multi-strategy ensemble evolutionary optimization for dynamic multi-objective optimization. Memetic Computing 2010;2:3–24.

[59] Tan, K.C., Lee, T.H., Khor, E.F.. Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. Artificial Intelligence Review 2002;17:253–290.

[60] Srinivas, N., Deb, K.. Multiobjective optimization using non-dominated sorting in genetic algorithms. Evolutionary Computation 1994;2(3):221–248.

[61] Grosan, C., Oltean, M., Dumitrescu, D.. Performance metrics for multiobjective optimization evolutionary algorithms. In: Proceedings of Conference on Applied and Industrial Mathematics. 2003,.

[62] Van Veldhuizen, D.A., Lamont, G.B.. Evolutionary computation and convergence to a pareto front. In: In Late Breaking Papers at the Genetic Programming 1998 Conference. 1998, p. 221–228.

[63] Chow, J., Regan, A.. A surrogate-based multiobjective metaheuristic and network degradation simulation model for for robust toll pricing. 2012. Civil Engineering Working Papers.

[64] Voutchkov, I., Keane, A.. Computational Intelligence in Optimization: Adaptation, Learning, and Optimization; chap. Multi-objective optimization using surrogates. 2010, p. 155–175.

[65] Deb, K., Agrawal, R.B.. Simulated binary crossover for continuous search space. Complex Systems 1995;9:115–148.

[66] Bradstreet, L., Barone, L., While, L., Huband, S., Hingston, P.. Use of the wfg toolkit and pisa for comparison of multi-objective evolutionary algorithms. In: Proceedings of IEEE Symposium on Compu-

tational Intelligence in Multi-criteria Decision-making. 2007, p. 382–389.

[67] Li, H., Landa-Silva, D.. An adaptive evolutionary multi-objective approach based on simulated annealing. Evolutionary Computation 2011;19(4):561–595.

Table 1: The WFG Test Functions

WFG1	$h_{M=1} : M = convex_m$ $h_M = mixed_M\ (with\ \alpha = 1\ and\ A = 5)$ $t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n} = S\_linear(y_i, 0.35)$ $t^2_{i=1:k} = y_i$ $t^2_{i=k+1:n} = b\_flat(y_i, 0.8, 0.75, 0.85)$ $t^3_{i=1:n} = b\_poly(y_i, 0.02)$ $t^4_{i=1:M-1} = r\_sum(\{y_{(i-1)k/(M-1)} + 1, ..., y_{ik/(M-1)}\}, \{2((i-1k/(M-1, ..., 2ik/(M-1)\})$ $t^4_M = r\_sum(\{y_{k+1}, ..., y_n\}, \{2(k+1), ..., 2n\})$
WFG2	$h_{M=1} : M = convex_m$ $h_M = disc_M\ (with\ \alpha = \beta = 1\ and\ A = 5)$ $As\ t^1\ from\ WFG1.\ (Linear\ shift.)$ $t^2_{i=1:k} = y_i$ $t^2_{i=k+1:k+l/2} = r\_nonsep(\{y_{k+2(i-k)-1}, y_{k+2(i-k)}\}, 2)$ $t^3_{i=1:M-1} = r\_sum(\{y_{(i-1)k/(M-1)} + 1, ..., y_{ik/(M-1)}\}, \{1, ..., 1\})$ $t^3_M = r\_sum(\{y_{k+}, ..., y_{k+l/2}\}, \{1, ..., 1\})$
WFG3	$h_{M=1} : M = linear_m(degenerate)$ $As\ t^{1:3}\ from\ WFG2.(Linear\ shift, non\ separable\ reduction$ $and\ weighted\ sum\ reduction.)$
WFG4	$h_{M=1} : M = concave_m$ $t^1_{i=1:n} = S\_multi(y_i, 30, 10, 0.35)$ $t_(i = 1 : M-1)^2 = r\_sum(\{y_{(i-1)k/(M-1)} + 1, ..., y_{ik/(M-1)}\}, \{1, ..., 1\})$ $t^2_M = r\_sum(\{y_{k+1}, ..., y_n\}, \{1, ..., 1\})$
WFG5	$h_{M=1} : M = concave_m$ $t^1_{i=1:n} = S\_decept(y_i, 0.35, 0.001, 0.05)$ $As\ t^2\ from\ WFG4.\ (weighted\ sum\ reduction.)$
WFG6	$h_{M=1} : M = concave_m$ $As\ t^1\ from\ WFG1.\ (Linear\ shift.)$ $t^2_{i=1:M-1} = r\_nonsep(\{y_{(i-1)k/(M-1)} + 1, ..., y_{ik/(M-1)}\}, k/(M-1))$ $t^2_M = r\_nonsep(\{y_{k+1}, ..., y_n\}, l)$
WFG7	$h_{M=1} : M = concave_m$ $t^2_{i=1:k} = b\_param(y_i, r\_sum(\{y_{(i-1)}, ..., y_n\}, \{1, ..., 1\}), 0.98/49.98, 0.02, 50)$ $t^2_{i=k+1:n} = y_i$ $As\ t^1\ from\ WFG1.\ (Linear\ shift.)$ $As\ t^2\ from\ WFG4.\ (weighted\ sum\ reduction.)$
WFG8	$h_{M=1} : M = concave_m$ $t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n} = b\_param(y_i, r\_sum(\{y_1, ..., y_{i-1}\}, \{1, ..., 1\}), 0.98/49.98, 0.02, 50)$ $Ast^1 fromWFG1.(Linearshift.)$ $As\ t^2\ from\ WFG4.\ (weighted\ sum\ reduction.)$
WFG9	$h_{M=1} : M = concave_m$ $t^1_{i=1:n-1} = b\_param(y_i, r\_sum(\{y_{i+1}, ..., y_n\}, \{1, ..., 1\}, 0.98/49.98, 0.02, 50)$ $t^1_n = y_n$ $t^2_{i=1:k} = S\_decept(y_i, 0.35, 0.001, 0.05)$ $t^2_{i=k+1:n} = S\_multi(y_i, 30, 95, 0.35)$ $As\ t^2\ from\ WFG6.\ (non\ separable\ reduction.)$

Table 2: The performance of comparison methods on the WFG test problems with respect to four metrics RNI, SSC, UD, GD and IGD

		RNI		SSC		UD		GD		IGD	
Problem	Methods	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
WFG1	AM	0.8800	0.2539	12.1386	0.9101	**0.4428**	0.1007	**7.740E-03**	1.11E-02	**7.30E-04**	6.50E-04
	GDA	0.9357	0.1821	**12.9388**	1.2517	0.3941	0.0698	8.24E-03	1.11E-02	1.02E-03	1.17E-03
	LA	**0.9950**	0.0292	12.1867	0.9967	0.3117	0.0521	1.53E-02	7.26E-03	2.40E-03	7.20E-04
WFG2	AM	0.2293	0.0545	11.0219	0.3042	**0.7278**	0.0661	1.46E-03	4.90E-04	4.40E-04	6.00E-05
	GDA	**1.0000**	0.0000	**11.8148**	0.0146	0.3729	0.0064	**4.500E-04**	8.00E-05	**3.50E-04**	0.00E+00
	LA	**1.0000**	0.0000	11.8139	0.1567	0.3716	0.0156	7.00E-04	7.30E-04	3.70E-04	8.00E-05
WFG3	AM	0.6027	0.0445	11.894	0.0853	**0.545**	0.0289	6.80E-04	4.50E-04	6.85E-04	7.17E-06
	GDA	**1.0000**	0.0000	**11.9197**	0.0064	0.4252	0.0120	**2.000E-04**	5.00E-05	**6.84E-04**	2.50E-07
	LA	**1.0000**	0.0000	11.9093	0.0162	0.4222	0.0094	4.10E-04	7.00E-05	6.84E-04	1.18E-06
WFG4	AM	0.5443	0.0452	9.6588	0.0176	**0.5596**	0.0361	9.70E-04	1.90E-04	2.56E-04	3.38E-05
	GDA	**1.0000**	0.0000	**9.6642**	0.0100	0.4145	0.0112	**4.700E-04**	4.00E-05	**1.30E-04**	1.03E-05
	LA	**1.0000**	0.0000	9.6512	0.0141	0.415	0.0143	6.10E-04	5.00E-05	1.48E-04	1.10E-05
WFG5	AM	0.8537	0.1723	9.2899	0.5744	**0.4779**	0.0468	2.73E-03	3.20E-04	5.44E-04	2.25E-05
	GDA	**1.0000**	0.0000	**9.2964**	0.4023	0.4395	0.0086	**2.450E-03**	1.00E-05	**5.29E-04**	6.67E-07
	LA	**1.0000**	0.0000	9.2772	0.0080	0.4170	0.0213	2.51E-03	3.00E-05	5.39E-04	8.86E-06
WFG6	AM	0.4720	0.0412	9.3687	0.0542	**0.5962**	0.0363	2.25E-03	5.62E-04	5.52E-04	6.75E-05
	GDA	**1.0000**	0.0000	**9.3745**	0.0628	0.4128	0.0083	**2.000E-03**	3.50E-04	**4.44E-04**	7.68E-05
	LA	**1.0000**	0.0000	9.3711	0.0474	0.4136	0.0129	2.05E-03	2.70E-04	4.47E-04	5.60E-05
WFG7	AM	0.6173	0.0653	9.6606	0.0926	**0.5289**	0.0416	4.70E-04	2.50E-04	2.21E-04	5.03E-05
	GDA	1.0000	0.0000	**9.6650**	0.0028	0.4085	0.0151	**3.300E-04**	4.00E-05	**1.19E-04**	7.97E-06
	LA	1.0000	0.0000	9.6641	0.0100	0.4112	0.0133	4.10E-04	4.00E-05	1.32E-04	1.19E-05
WFG8	AM	0.2627	0.0454	8.3033	0.1224	**0.7886**	0.1245	4.42E-03	4.30E-04	6.20E-04	7.77E-05
	GDA	**1.0000**	0.0000	**8.7279**	0.0120	0.4248	0.0341	**3.890E-03**	3.80E-04	**3.63E-04**	1.40E-05
	LA	**1.0000**	0.0000	8.4859	0.0754	0.4128	0.0136	4.41E-03	1.50E-04	4.21E-04	1.37E-05
WFG9	AM	0.6410	0.0896	8.6132	0.2236	**0.5142**	0.0525	5.28E-03	1.45E-03	9.55E-04	2.44E-04
	GDA	0.9893	0.4193	**8.7689**	0.3054	0.4111	0.0210	**3.640E-03**	1.95E-03	**7.88E-04**	3.91E-04
	LA	**0.9973**	0.0146	8.7132	0.2518	0.3953	0.0144	3.77E-03	1.69E-03	8.31E-04	3.54E-04

Table 3: Pairwise t-test results for the choice function based multi-objective hyper-heuristics using AM, GDA and LA as the move acceptance component on each WFG test problem instance with respect to different metrics; the ratio of non-dominated individuals (RNI), the hypervolume (SSC), the uniform distribution (UD), the generational distance (GD) and the inverted generational distance (IGD).

Problem	Methods	Metrics				
		RNI	SSC	UD	GD	IGD
WFG1	AM-GDA	$\mp$	-	+	$\pm$	+
	AM-LA	-	-	+	+	+
	GDA-LA	$\mp$	+	$\pm$	+	+
WFG2	AM-GDA	-	-	+	-	-
	AM-LA	-	-	+	-	-
	GDA-LA	$\approx$	$\pm$	$\pm$	+	+
WFG3	AM-GDA	-	-	+	-	$\mp$
	AM-LA	-	-	+	-	$\mp$
	GDA-LA	$\approx$	+	$\pm$	+	$\pm$
WFG4	AM-GDA	-	-	+	-	-
	AM-LA	-	-	$\pm$	-	-
	GDA-LA	$\approx$	+	$\mp$	+	+
WFG5	AM-GDA	-	-	+	-	-
	AM-LA	-	$\pm$	+	-	$\mp$
	GDA-LA	$\approx$	+	+	+	$\pm$
WFG6	AM-GDA	-	-	+	-	-
	AM-LA	-	-	+	-	-
	GDA-LA	$\approx$	+	$\mp$	+	$\pm$
WFG7	AM-GDA	-	-	+	-	-
	AM-LA	-	-	+	-	-
	GDA-LA	$\approx$	+	$\mp$	+	$\pm$
WFG8	AM-GDA	-	-	+	-	-
	AM-LA	-	-	+	-	-
	GDA-LA	$\approx$	+	+	+	+
WFG9	AM-GDA	-	-	+	-	-
	AM-LA	-	-	+	-	-
	GDA-LA	$\mp$	+	+	$\pm$	+

Table 4: The vehicle crashworthiness problems

Problem Name	Objective Functions
Car1	Mass and Ain
Car2	Mass and Intrusion
Car3	Ain and Intrusion
Car4	Mass, Ain and Intrusion

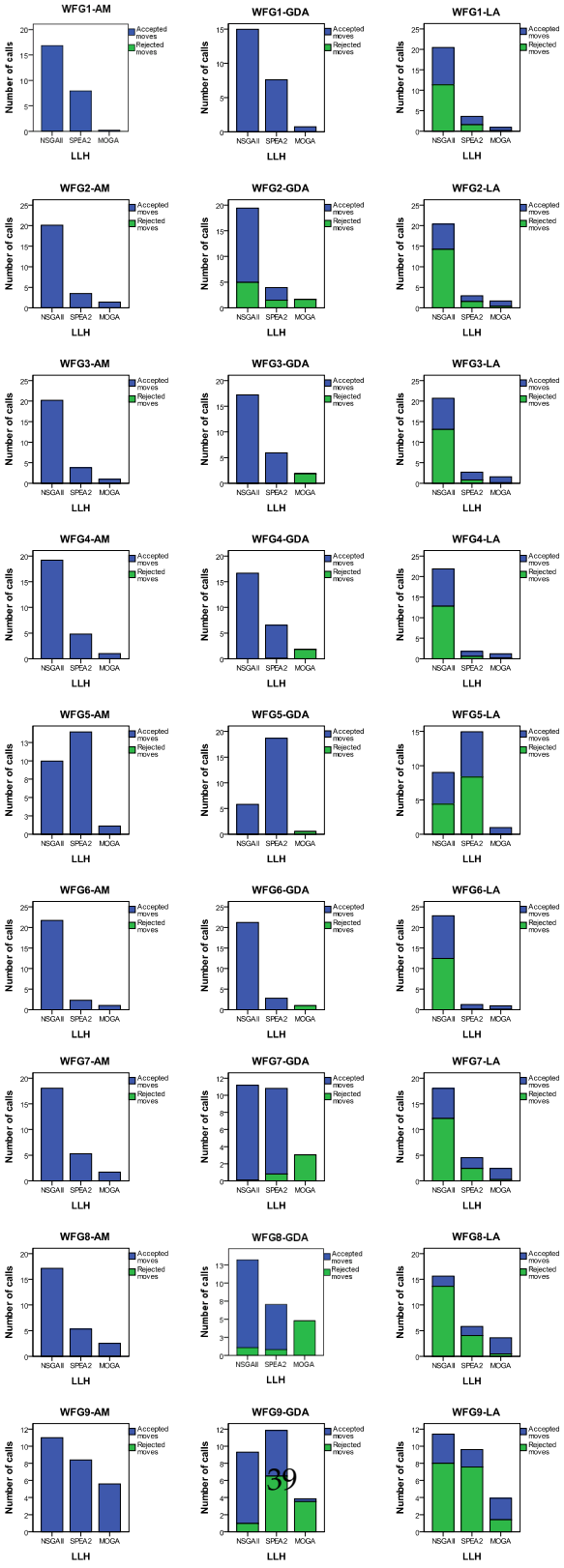

Figure 2: The average number of low level metaheuristic invocations (NSGAII, SPEA2 and MOGA) and accepted/rejected moves produced by selection hyper-heuristics using AM, GDA and LA over the WFG test problems.

Table 5: The performance comparison of hyper-heuristic methods embedding AM, GDA and LA, and NSGAII on the vehicle crashworthiness problems

		RNI		SSC		UD		GD		IGD	
vehicle problems	Methods	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Car1	NSGAII	1.000	0.000	2.30E+04	1.40E-01	0.450	0.021	8.10E-04	4.00E-04	4.66E-04	3.11E-05
	AM	0.980	0.050	2.11E+04	5.05E+03	0.430	0.067	7.50E-04	4.70E-04	5.87E-03	5.99E-03
	GDA	**1.000**	0.000	**2.298E+04**	3.88E+03	**0.453**	0.020	**4.500E-04**	2.00E-04	**4.28E-04**	5.76E-05
	LA	0.990	0.000	2.17E+04	3.98E+03	0.452	0.031	8.40E-04	6.00E-04	6.91E-04	1.36E-03
Car2	NSGAII	1.000	0.000	3.93E+04	1.63E+04	**0.461**	0.032	2.45E-03	3.28E-03	3.17E-03	2.89E-03
	AM	0.950	0.090	3.77E+04	1.71E+04	0.427	0.095	2.30E-03	3.12E-03	4.97E-03	3.53E-03
	GDA	**1.000**	0.000	**3.953E+04**	1.69E+04	0.451	0.020	**1.860E-03**	2.12E-03	**3.13E-03**	3.63E-03
	LA	1.000	0.000	2.11E+03	1.51E+04	0.450	0.021	2.50E-03	3.34E-03	4.18E-03	2.88E-03
Car3	NSGAII	1.000	0.000	4.17E+01	8.82E+00	0.464	0.022	**1.008E-01**	4.02E-03	**9.93E-02**	5.07E-02
	AM	0.980	0.080	4.06E+01	1.02E+01	0.478	0.031	1.03E-01	3.83E-03	1.65E-01	6.29E-02
	GDA	**1.000**	0.000	**4.175E+01**	9.98E+00	**0.480**	0.021	1.03E-01	7.53E-03	1.26E-01	6.47E-02
	LA	**1.000**	0.000	4.15E+01	9.68E+00	0.463	0.033	1.03E-01	4.66E-03	1.42E-01	5.74E-02
Car4	NSGAII	**1.000**	0.000	7.94E+07	1.60E+07	0.592	0.045	2.48E-03	9.10E-04	4.16E-03	3.86E-03
	AM	**1.000**	0.000	7.38E+07	1.46E+07	0.585	0.050	2.71E-03	7.90E-04	4.38E-03	4.17E-03
	GDA	**1.000**	0.000	**8.289E+07**	1.95E+07	**0.613**	0.034	**2.110E-03**	7.10E-04	**3.55E-03**	3.08E-03
	LA	**1.000**	0.000	7.54E+07	1.47E+07	0.582	0.062	3.32E-03	1.33E-03	3.60E-03	2.58E-03

Table 6: Pairwise t-test results for NSGAII and choice function based multi-objective hyper-heuristics using AM, GDA and LA as the move acceptance component on the vehicle crashworthiness problems with respect to different metrics; the ratio of non-dominated individuals (RNI), the hypervolume (SSC), the uniform distribution (UD), the generational distance (GD) and the inverted generational distance (IGD)

Problem	Methods	Metrics				
		RNI	SSC	UD	GD	IGD
Car1	NSGAII-AM	$\pm$	+	+	$\mp$	+
	NSGAII-GDA	$\approx$	-	-	-	$\mp$
	NSGAII-LA	$\pm$	+	-	$\pm$	$\pm$
	AM-GDA	$\mp$	-	+	-	-
	AM-LA	$\mp$	$\mp$	-	-	-
	GDA-LA	$\pm$	+	$\pm$	+	+
Car2	NSGAII-AM	$\pm$	+	+	$\mp$	$\pm$
	NSGAII-GDA	$\approx$	$\mp$	$\pm$	-	$\mp$
	NSGAII-LA	$\approx$	+	+	$\mp$	+
	AM-GDA	$\mp$	-	-	-	-
	AM-LA	$\mp$	+	-	$\pm$	$\mp$
	GDA-LA	$\approx$	+	$\pm$	+	+
Car3	NSGAII-AM	$\pm$	+	+	$\pm$	+
	NSGAII-GDA	$\approx$	$\mp$	-	$\pm$	+
	NSGAII-LA	$\approx$	$\pm$	$\pm$	$\pm$	+
	AM-GDA	$\mp$	-	$\mp$	$\mp$	-
	AM-LA	$\mp$	-	+	$\pm$	-
	GDA-LA	$\approx$	$\pm$	+	$\pm$	+
Car4	NSGAII-AM	$\approx$	+	$\pm$	$\pm$	-
	NSGAII-GDA	$\approx$	-	+	$\mp$	-
	NSGAII-LA	$\approx$	+	+	+	-
	AM-GDA	$\approx$	-	-	$\mp$	-
	AM-LA	$\approx$	-	$\pm$	+	-
	GDA-LA	$\approx$	+	+	+	$\mp$

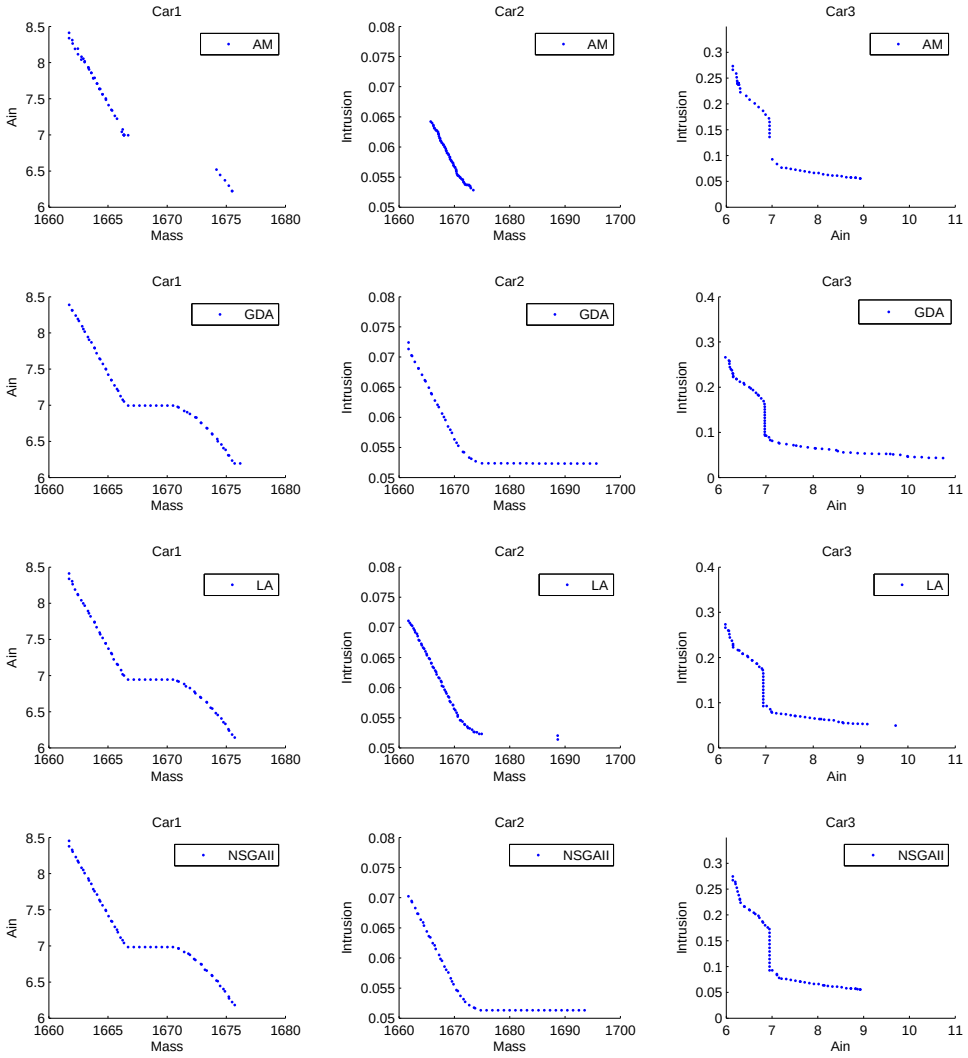

Figure 3: The 50% attainment surfaces for NSGAII and choice function based hyper-heuristics embedding AM, GDA and LA, which are averaged over 30 fronts obtained after 3750 generations on Car1, Car2 and Car3.

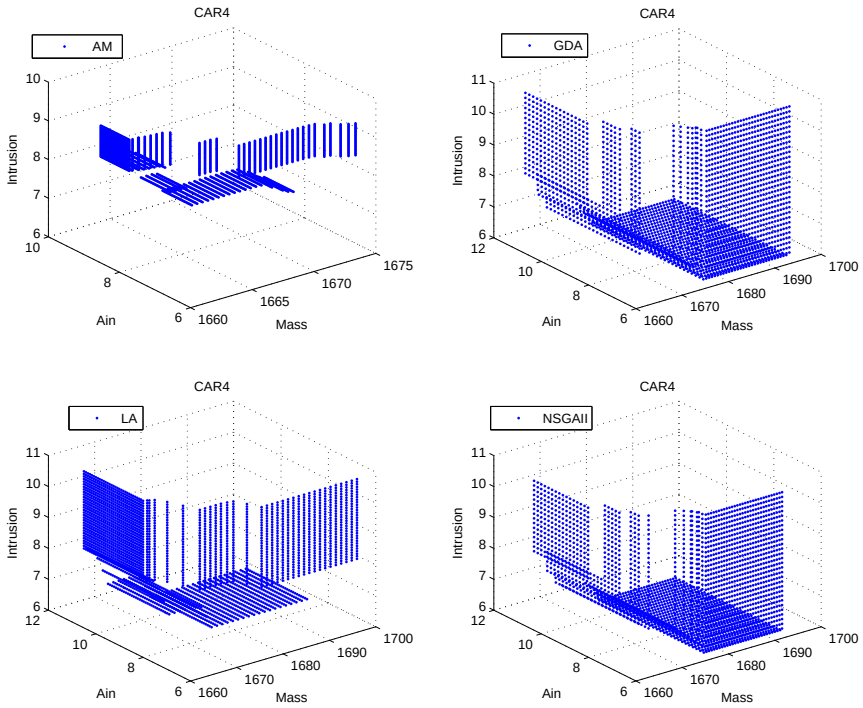

Figure 4: The 50% attainment surfaces for NSGAII and choice function based hyper-heuristics embedding AM, GDA and LA, which are averaged over 30 fronts obtained after 3750 generations on Car4.