Applied Soft Computing Journal 97 (2020) 106760

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Hyper-Heuristics based on Reinforcement Learning, Balanced Heuristic Selection and Group Decision Acceptance

Valdivino Alexandre de Santiago Júnior[a,b,*], Ender Özcan[b], Vinicius Renan de Carvalho[c,b]

[a] *Instituto Nacional de Pesquisas Espaciais (INPE), Coordenação de Pesquisa Aplicada e Desenvolvimento Tecnológico (COPDT) - Av. dos Astronautas, 1758, São José dos Campos, SP, 12227-010, Brazil*
[b] *University of Nottingham, Computational Optimisation and Learning (COL) Lab, School of Computer Science, Wollaton Road, Nottingham, NG8 1BB, United Kingdom*
[c] *Universidade de São Paulo (USP), Escola Politécnica (Poli) - Av. Prof. Luciano Gualberto, 380, São Paulo, SP, 05508-010, Brazil*

ARTICLE INFO

Article history:
Received 27 September 2019
Received in revised form 17 August 2020
Accepted 25 September 2020
Available online 1 October 2020

Keywords:
Hyper-heuristic
Reinforcement learning
Balanced heuristic selection
Group decision-making
Multi-objective evolutionary algorithms
Multi-objective optimisation

ABSTRACT

In this paper, we introduce a multi-objective selection hyper-heuristic approach combining Reinforcement Learning, (meta)heuristic selection, and group decision-making as acceptance methods, referred to as Hyper-Heuristic based on Reinforcement LearnIng, Balanced Heuristic Selection and Group Decision AccEptance (HRISE), controlling a set of Multi-Objective Evolutionary Algorithms (MOEAs) as Low-Level (meta)Heuristics (LLHs). Along with the use of multiple MOEAs, we believe that having a robust LLH selection method as well as several move acceptance methods at our disposal would lead to an improved general-purpose method producing most adequate solutions to the problem instances across multiple domains. We present two learning hyper-heuristics based on the HRISE framework for multi-objective optimisation, each embedding a group decision-making acceptance method under a different rule: majority rule (HRISE_M) and responsibility rule (HRISE_R). A third hyper-heuristic is also defined where both a random LLH selection and a random move acceptance strategy are used. We also propose two variants of the late acceptance method and a new quality indicator supporting the initialisation of selection hyper-heuristics using low computational budget. An extensive set of experiments were performed using 39 multi-objective problem instances from various domains where 24 are from four different benchmark function classes, and the remaining 15 instances are from four different real-world problems. The cross-domain search performance of the proposed learning hyper-heuristics indeed turned out to be the best, particularly HRISE_R, when compared to three other selection hyper-heuristics, including a recently proposed one, and all low-level MOEAs each run in isolation.

1. Introduction

The main motivation behind hyper-heuristics is to resolve some of the issues with metaheuristics. Although metaheuristics have long been proved beneficial to solve real-world complex search problems such as scheduling, clustering, educational timetabling, and space allocation, it is still not straightforward to apply them directly to new optimisation problems from different domains with no or minimal change, or even to new instances of the same problem [1,2]. Some of the reasons of such difficulties are the usual high number of parameters or algorithm choices the practitioner must define, and the absence of proper guidelines to select them.

Hence, hyper-heuristics have emerged which represent a class of high-level search techniques whose goal is to raise the level of generality at which search methods work [1,2]. In hyper-heuristics, the search is performed in the space of heuristics (or heuristics components) instead of being performed directly in the decision variable space (space of solutions). Thus, hyper-heuristics are more general and so appropriate to solve different problems rather than a specific problem based on a set of Low-Level (meta)Heuristics (LLHs).

A generic selection hyper-heuristic consists of two key components: heuristic selection and move acceptance methods [3]. As the name implies, in heuristic selection there is a set of LLHs and the strategy must choose the most appropriate one to run at a certain moment of the search process. The move acceptance method decides whether a solution (or a subset of solutions) that is the result of the execution of the selected LLH is accepted

* Corresponding author at: Instituto Nacional de Pesquisas Espaciais (INPE), Coordenação de Pesquisa Aplicada e Desenvolvimento Tecnológico (COPDT) - Av. dos Astronautas, 1758, São José dos Campos, SP, 12227-010, Brazil.
E-mail addresses: valdivino.santiago@inpe.br (V.A. Santiago Júnior), Ender.Ozcan@nottingham.ac.uk (E. Özcan), vrcarvalho@usp.br (V.R. Carvalho).

https://doi.org/10.1016/j.asoc.2020.106760
1568-4946/© 2020 Elsevier B.V. All rights reserved.

or not. As we will explain below, most studies do not exploit the benefits of relying on a robust heuristic selection strategy while simultaneously having a more elaborate and effective move acceptance method for accepting the generated solutions (e.g. in a population).

There are many studies on selection hyper-heuristics based on perturbative LLHs mostly using a single point-based search framework for single objective optimisation [3–8]. However, the use of population-based methods is a recently growing area of research [9] where hyper-heuristics for multi-objective optimisation are employed either for controlling the own components of Multi-Objective Evolutionary Algorithms (MOEAs) [10–12], e.g. mutation and crossover operators as LLHs, or the LLHs being complete MOEAs [13–16].

Despite these efforts and even if there are several proposed selection of LLHs and move acceptance methods, the hyper-heuristics usually consist of a single LLH selection method, with no additional policies, combined with a single move acceptance approach. Few studies have made a contribution where more than one heuristic selection/policy and/or move acceptance method is considered. For instance, Li et al. (2019) [16] proposed a new heuristic selection method, ϵ-RouletteGreedy, which is based on greedy and roulette wheel strategies. But only a single move acceptance method is used. In Kheiri et al. (2016) [17], ensemble move acceptance methods are combined under a group decision-making framework but employing a single point-based search.

We believe that having a set of LLH selection, policies, and move acceptance methods at our disposal can produce a most adequate strategy to solve better the optimisation problems, since we can benefit of the combined strengths of all the approaches. Thus, in this paper we present a novel online selection hyper-heuristic based on perturbative LLHs which aims to solve complex multi-objective problems via a heuristic selection approach based on Reinforcement Learning and a balanced mechanism, and move acceptance methods. The **H**yper-Heuristic based on **R**einforcement Learn**I**ng, Balanced Heuristic **S**election and Group Decision Acc**E**ptance (HRISE) contemplates a single LLH selection method but with additional policies and three move acceptance methods within group decision-making rules. Our hyper-heuristic considers as LLHs complete MOEAs and we indeed propose two variants of it based on group decision-making acceptance: one based on the majority rule (HRISE_M) and another one based on the responsibility rule (HRISE_R). A third hyper-heuristic, called **H**yper-Heuristic based on **R**andom LLH Selection and **R**andom Choice of **M**ove **A**cceptance Methods (HRMA), is also proposed where both a random choice of (meta)heuristic selection is performed combined with a random choice of move acceptance methods.

We compared the performance of HRISE_M, HRISE_R, and HRMA to three other hyper-heuristics, including the Choice Function hyper-heuristic (HH-CF) [13], a random choice (meta) heuristic selection hyper-heuristic with All Moves acceptance (HH-ALL), and a recent one Learning Automata-based Hyper-Heuristic with a Ranking Scheme Initialisation (HH-RILA) [16] which is the state-of-the-art. We also performed the comparison to the following three MOEAs run in isolation which are the LLHs in all hyper-heuristics: Nondominated Sorting Genetic Algorithm-II (NSGA-II) [18], Indicator-Based Evolutionary Algorithm (IBEA) [19], and Strength Pareto Evolutionary Algorithm-2 (SPEA2) [20]. The experiments considered 39 multi-objective[1] problem instances from various domains where 24 are from four different benchmark function classes, and the remaining 15 instances are from four different real-world problems.

The main contributions of this study are:

1. We propose two new selection hyper-heuristics (HRISE_M and HRISE_R) based on an approach embedding a robust heuristic selection strategy, via Reinforcement Learning, and enabling the use of several move acceptance methods under a group-decision framework. Our idea is to rely on a range of methods not only to select a new LLH but also to accept a population of trade-off solutions generated after the execution of a selected LLH. To the best of our knowledge, no previous approach has stressed both the heuristic selection (roulette wheel supported by Reinforcement Learning plus a balanced exploitation/exploration method) and the move acceptance (two-level strategy: Only Improving + group decision-making) mechanisms as we suggest in this work;
2. We propose two new move acceptance methods based on Late Acceptance [21,22]. In Qualified Late Acceptance (QLA), we compare the quality of the current generated solution (population) to the quality of one that has been created but accepted s steps before. In Mean Qualified Late Acceptance (MQLA), we consider the mean value of the quality of the already accepted populations which are stored in the memory. These methods may hence be incorporated into any selection hyper-heuristic;
3. Based on the framework, we also defined a third hyper-heuristic (HRMA) in which we not only randomly select a new LLH but also randomly choose one of the move acceptance methods;
4. We also propose a new quality indicator which does not require a True Pareto Front, Reference Set (reference points) or nondominance of solutions, but only the values of the objective functions. Hence, even for many-objective optimisation problems, it is suitable for supporting a initialisation process considering a few iterations with low computational budget;
5. We devise a dynamic control mechanism where the number of iterations a selected LLH will run varies along the search process;
6. We formulate a real-world problem related to space applications communication which can be used as another benchmark in the future for evaluating multi-objective optimisation algorithms.

This paper is organised as follows. The related work is presented in Section 2. Section 3 introduces the proposed hyper-heuristics and the problems dealt with are provided in Section 4. The experimental results are discussed in Section 5. Finally, the conclusions and future research directions are in Section 6.

2. Related work

In this section we present some relevant studies related to selection hyper-heuristics based on perturbative LLHs. We divide them in three categories: single point search-based, population-based with components of MOEAs, and population-based with complete MOEAs.

Single point search-based selection hyper-heuristics have been dominated the field as presented in [2]. The authors distributed the studies in several classes such as hyper-heuristics using deterministic move acceptance [4,5], hyper-heuristics using heuristic selection with no learning and non-deterministic move acceptance [6], and hyper-heuristics using heuristic selection with online learning and non-deterministic move acceptance [3,7,8].

[1] One problem considered is, in fact, a many-objective one where its instances have four or five objective functions. But, as most of the analysed problem instances have three objectives or less, so we prefer the term multi-objective problem instances.

Although there are several heuristic selection and move acceptance methods addressed, these studies assess the performance of a single selection of LLH method with no additional policies, combined with a single move acceptance approach. In this study, we use roulette wheel supported by Reinforcement Learning together with policies for selecting LLHs, and a two-level (hierarchical) move acceptance method formed by Only Improving and group decision acceptance.

With respect to the studies of selection hyper-heuristics using population-based approaches, here we see strategies which control components (e.g. crossover and mutation operators) of a MOEA [9]. In [23], the Water Distribution Network (WDN) design problem is addressed via a selection hyper-heuristic based on Markov Chains [24] which is incorporated into NSGA-II and SPEA2. They considered a range of mutation and crossover operator-based LLHs. Some of these previous studies are in the context of Search-Based Software Testing (SBST) [25–28] addressing problems such as integration and test order [10], derivation of products for Software Product Line (SPL) testing [11], and Second Order Mutants (SOMs) generation strategies [12]. All these studies only consider a single heuristic selection method (e.g. Choice Function, Upper Confidence Bound, Multi-Armed Bandit [29,30]) with no additional policies to improve the overall approach. Moreover, we believe that if complete MOEAs are used as LLHs, we can benefit of the strengths of each low-level metaheuristic as a whole.

Recent studies fall then in this latter category where instead of operators of a single metaheuristic, complete MOEAs are used as LLHs. In [13], the authors presented a hyper-heuristic based on Choice Function in which NSGA-II, SPEA2, and the Multi-Objective Genetic Algorithm (MOGA) [31] are the LLHs. In [14], researchers used NSGA-II, IBEA, and SPEA2 as LLHs and evaluated several heuristic selection (Fixed Sequence, Choice Function) and acceptance (Great Deluge Acceptance (GDA) [3,32] with D-metric, Best Acceptance) methods for wind farm layout optimisation. The same previous remarks apply here: only one heuristic selection, no additional policies, and only one acceptance method is defined for each hyper-heuristic.

MOABHH is an agent-based hyper-heuristic framework focused on online selection by means of voting techniques [15]. The main idea is to consider MOEAs as candidates and quality indicators as voters in an election. NSGA-II, SPEA2, IBEA, and the Generalised Differential Evolution 3 (GDE3) [33] are the LLHs. They used a voting method to select an LLH based on the Condorcet's principle which is similar to the group decision-making majority rule, but we used this rule as a move acceptor and not as a selector. Moreover, they used only the All Moves acceptance method while we have a set of acceptance methods.

In [16], the authors proposed a learning automata-based selection hyper-heuristic. There are two variants of the hyper-heuristic depending on whether a uniform (HH-LA) or a ranking-based (HH-RILA) initialisation process is chosen. LLHs are NSGA-II, IBEA, and SPEA2. They defined a selection method, named ϵ-RouletteGreedy, in which they apply roulette wheel in the initial stage and, after that, they select between greedy and roulette wheel based on ϵ, the probability of applying the greedy heuristic selection method. Such a probability is increased linearly during the execution. In our case, we defined a light (low demanding in computational terms) initialisation process based on a new quality indicator, and our additional policies are within a robust balanced exploitation/exploration approach. They used only one move acceptance method (Only Improving) while we have a two-level (hierarchical) approach and, moreover, we also defined a nonuniform iterations strategy addressing the number of times an LLH can execute.

3. The HRISE/HRMA selection hyper-heuristics

In this section, we describe both variants of the HRISE hyper-heuristic: HRISE_M and HRISE_R. We designed the HRMA hyper-heuristic under the same general structure of HRISE and hence we address it here too. Fig. 1 presents our hyper-heuristics using an activity diagram while Algorithm 1 is the main HRISE/HRMA procedure where we show the main features of our proposals in an algorithmic format.

We mention some general observations below to show the main differences between the three approaches. An in-depth explanation of the features of our hyper-heuristics are provided later in this section. The LLHs of HRISE are complete MOEAs hence we consider not only their components (e.g. mutation and crossover operations). Note that in Fig. 1 some activities/tasks only exist for HRISE_M and HRISE_R, i.e. balanced exploitation/exploration (*Apply Balanced Exp/Exp*) and the activity related to Reinforcement Learning. Moreover, the heuristic selection is roulette wheel [34] for HRISE_M and HRISE_R and it is a random LLH selection within HRMA. In the group decision move acceptance mechanism, the majority (HRISE_M) or responsibility (HRISE_R) rules are used depending on the hyper-heuristic and, for HRMA, we have a random choice among the available move acceptance methods.

As shown in Algorithm 1, HRISE/HRMA receives as inputs the problem instance (Pr), the version of the hyper-heuristic ($v = M$, R, A for HRISE_M, HRISE_R, HRMA, respectively), the maximum number of iterations (mi), the maximum number of decision points (md), the population size (z), and the sequence[2] of LLHs (LLH). The final resulting population is Pop^f.

The first activity in Fig. 1 is the initialisation where we designed a computationally low demanding procedure (*compLowInitialisationRPO* in Algorithm 1) in which we run each LLH k for very few iterations[3] and only once, and decide the first to be executed based on a new quality indicator which we detail in Section 3.1. We record the previous performance of an LLH k in Per, a sequence whose elements indicate if k had its offspring population accepted in the last time it was executed.

After the execution of the first selected LLH, for HRISE_M and HRISE_R, our approach first selects the next LLH to run based on roulette wheel [34] considering the utility values (UV) associated with the LLHs. The utility value of an LLH k, μ_k, influences the probability of k to be selected and as higher the utility value, the better. After the first decision point, all LLHs have the same utility value and hence a random choice is made. However, the Reinforcement Learning step of our approaches (see Section 3.3) rewards or penalises an LLH according to its performance, and this reflects on the heuristic selection at future decision points.

As for HRISE_M and HRISE_R, a balanced exploitation/exploration procedure (*balancedExpExp* in Algorithm 1; more details in Section 3.2) is performed following the LLH selection via roulette wheel. As we have previously mentioned, for HRMA an LHH is randomly selected, and neither the balanced exploitation/exploration mechanism nor the Reinforcement Learning step are used.

Traditionally, the number of iterations a selected LLH will run is fixed, e.g. an LLH runs for 10 iterations [16], 250 iterations [13],

[2] A sequence differs from a set because repetition of elements is allowed and order matters. Even if we do not expect that the user inputs repeated names of metaheuristics, the order that they are defined is relevant because of the index, which is related to the metaheuristic, being used by other sequences of the approach.

[3] We prefer denoting "iteration" rather than "generation" in order to be more general. On the other hand, we use "decision point" in the same sense as other authors use "iteration".

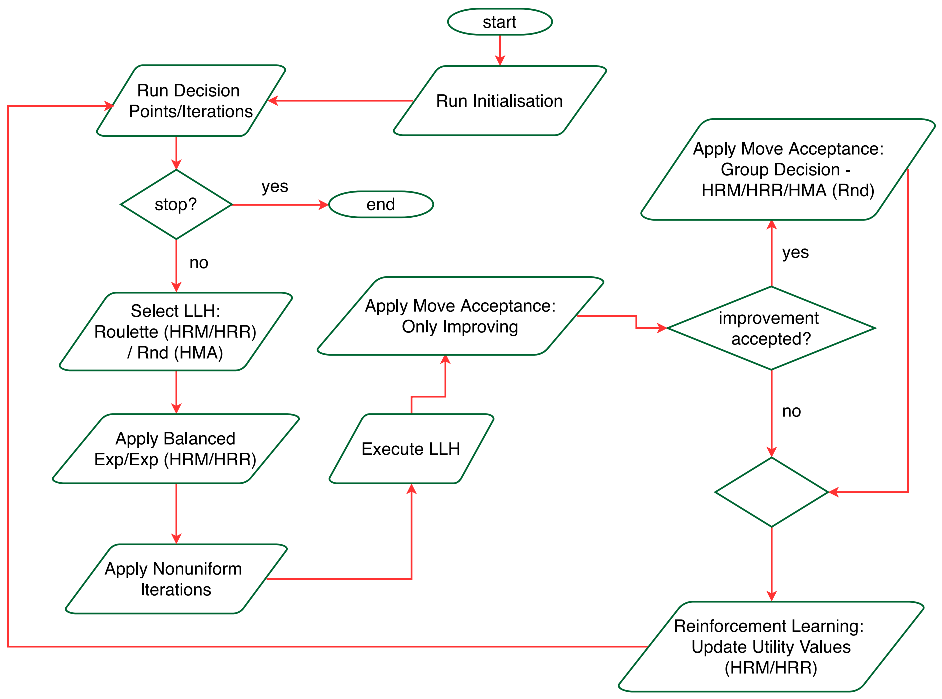

start
Run Initialisation
Run Decision Points/Iterations
stop?
yes
end
no
Select LLH: Roulette (HRM/HRR) / Rnd (HMA)
Apply Balanced Exp/Exp (HRM/HRR)
Apply Nonuniform Iterations
Execute LLH
Apply Move Acceptance: Only Improving
improvement accepted?
yes
Apply Move Acceptance: Group Decision - HRM/HRR/HMA (Rnd)
no
Reinforcement Learning: Update Utility Values (HRM/HRR)

Fig. 1. The HRISE/HRMA selection hyper-heuristics: activity diagram. Caption: HRM = HRISE_M; HRR = HRISE_R; HMA = HRMA; Rnd = Random; Exp/Exp = Exploitation/Exploration.

and so on. We can denote it as a uniform approach regarding the number of iterations/LLH.

But, we have observed that a nonuniform technique is worth being considered where the number of iterations an LLH will execute varies (*nonUniformIterations* in Algorithm 1). We envisage three scenarios as presented in Fig. 2. The *descending* approach is where we give more iterations to a selected LLH to be executed within the very first decision points while providing less iterations in the last decision points. The main reasoning behind this idea is trying to get fitter populations at the beginning by giving the first selected LLHs more iterations to run, but allowing more decision points to exist by providing less iterations later.

Even if we give a higher number of iterations at the beginning, it is possible, for some problem instances, that the populations consist of nondominated solutions but some values of objective functions are extremely high while others are very low. This may cause the hypervolume [35], the quality indicator we consider to drive decisions within our hyper-heuristics, to be extremely low or even 0 at the end of the number of iterations. Hence, by naturally decreasing the number of iterations as in the *descending* approach is not enough. In addition, we chose hypervolume as our key quality indicator since previous studies state that as maximum the hypervolume indicator is the better, because this is equivalent to optimising the overall objective leading to an optimal approximation of the True Pareto Front [36].

The *constant* strategy is then the one where we identify this issue and maintain the number of iterations huge and with the same value up to the termination criterion. The sequence EQ is input to the nonuniform technique and has every value of quality indicators (hypervolumes) due to the offspring populations (Pop_t). In this case, we check whether such hypervolume, h_j, is less than or equal to a threshold, γ, and count the number of these very low values. If this count exceeds 1, hence we can conclude that it is important to keep constant and high the number of iterations/LLH.

The third scenario is called *bathtub*. In other words, we start with a high number of iterations, decrease this number of iterations as in the *descending* strategy but, at some point, we realise that offspring populations have not been accepted and hence we raise again the number of iterations, providing more resources for the later selected LLHs.

Note that it is very possible that the number of decision points, and consequently the number of different LLHs selected, that will be really executed can be, in many situations, less than the value md given as input to our approach. The selected LLH, sel, is then executed (*executeLLH* in Algorithm 1) considering the population size, z, the number of iterations/LLH, n, and the problem instance, Pr. Pop_h is really effective after the execution of the first selected LLH and its function is related to the sharing of populations through the decision points.

Within our acceptance approach (lines from 20 through 39), the population due to the first selected LLH is always accepted, and its respective performance is positive ($Per_k = \top$ (True)). The group decision acceptance (*groupDecAcc* in Algorithm 1) takes place to decide whether the offspring population must be accepted. The input version of the algorithm dictates the acceptance mechanism (more details in Section 3.4). However, regardless of the version (HRISE_M, HRISE_R, HRMA), we defined a two-level (hierarchical) acceptance mechanism where in the first stage we realise whether a quality indicator improvement (*onlyImproving* in Algorithm 1) happens. Considering the hypervolume [35], the hypervolume improvement, $\delta(h_j)$, is accepted if the following condition is satisfied:

$$\delta(h_j) = \frac{h_j - a_{|AQ|}}{a_{|AQ|}} \geq \gamma \tag{1}$$

where h_j is the hypervolume of the current decision point, $a_{|AQ|}$ is the hypervolume of the last accepted population which is stored in the sequence AQ, and γ is a threshold (note that this is the same parameter we have just defined above within the nonuniform strategy). Only if this condition holds is that the group acceptance criterion, i.e. the second level, is applied. Considering this two-level acceptance mechanism (Only Improving and group decision), it is more demanding for a population to be accepted.

If a population is accepted, the current hypervolume (EQ_j) is added to AQ and the last performance of the LLH k is positive ($\top$).

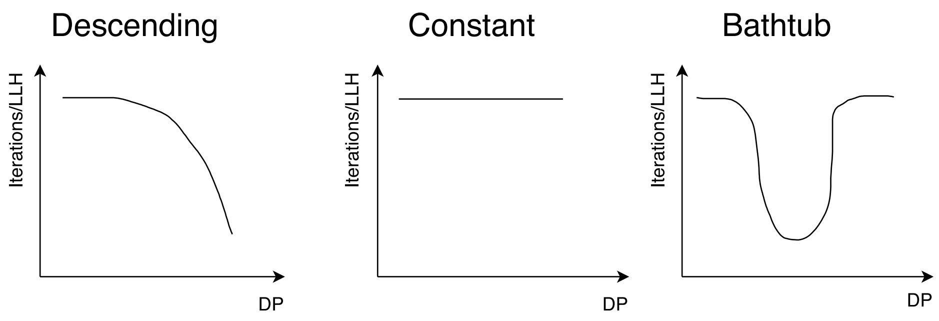


Fig. 2. The nonuniform iterations strategy. Caption: DP = Decision Point.

Algorithm 1 The HRISE/HRMA selection hyper-heuristics: algorithmic format

```
input: $Pr$, $v$, $mi$, $md$, $z$, $LLH$
output: $Pop^f$

1: $i = 1, j = 1, Pop^h = \emptyset, Pop^f = \emptyset, EQ = \emptyset, AQ = \emptyset, IQ = \{IQ_k = 0 \mid k = 1 \cdots |LLH|\}$
2: $Per = \{Per_k = \perp \mid k = 1 \cdots |LLH|\}$
3: $selInit = compLowInitialisationRPO(LLH, z, Pr)$
4: while $(j <= md) \wedge (i <= mi)$ do
5:     if $j = 1$ then
6:         $sel = selInit$
7:     else
8:         if $(v = M) \vee (v = R)$ then
9:             $sel = selectLLH(UV)$
10:            $sel = balancedExpExp(sel, Per, \eta_a, IQ, LLH, AQ)$
11:        else
12:            $sel = getRandom(LLH)$
13:        end if
14:    end if
15:    $Pop^h = adjustPop(Pop^h, z)$
16:    $n = nonUniformIterations(EQ, j, mi, i)$
17:    $i = i + n$
18:    $Pop^t = executeLLH(sel, z, n, Pr, Pop^h)$
19:    $EQ = EQ \cup measureQuality(Pop^t, Pr)$
20:    if $j > 1$ then
21:        $IQ = updateIQ(sel)$
22:        if $onlyImproving(IQ, AQ)$ then
23:            $Pop^h = updatePop(Pop^h, Pop^t)$
24:            if $groupDecAcc(v, EQ_j, j, md, \alpha)$ then
25:                $Per_k = \top$
26:                $AQ = AQ \cup EQ_j$
27:                $Pop^f = updatePop(Pop^f, Pop^h)$
28:            else
29:                $Per_k = \perp$
30:            end if
31:        else
32:            $Per_k = \perp$
33:            $penaliseUV(sel, j, md)$
34:        end if
35:    else
36:        $Per_k = \top$
37:        $AQ = AQ \cup EQ_j$
38:        $Pop^f = updatePop(Pop^f, Pop^h)$
39:    end if
40:    $j = j + 1$
41: end while
42: return $Pop^f$
```

The final population and output of our hyper-heuristics is Pop_f while Pop_h is a population that is considered as the initial from the second decision point onward, and whose value is adjusted before the execution of a new selected LLH.

The performance of an LLH k is negatively recorded ($\perp$, False) as well as it is penalised in accordance with the Reinforcement Learning approach (Section 3.3) in any situation in which the new set of solutions is rejected.

3.1. The Relative Performance Per Objective indicator

Capacity metrics [37], such as Overall Nondominated Vector Generation (ONVG) [38], Overall Nondominated Vector Generation Ratio (ONVGR) [38], and Ratio of Nondominated solutions (RNI) [39], require that populations have nondominated solutions to work out. However, if we run a MOEA for very few iterations considering a problem instance, no solution may dominate another one and hence these indicators are of no value in these situations.

Popular convergence metrics, such as Generation Distance (GD) [40], ϵ and $\epsilon+$ indicators [41], Seven Points Average Distance (SPAD) [42], convergence-diversity metrics, such as hypervolume (s-metric) [35], Inverted Generational Distance (IGD) [43], Modified Inverted Generational Distance (IGD+) [44], and diversity metrics, such as Uniform Distribution (UD) [39], spread (Δ) [18] and generalised spread (Δ^*) [45], demand the existence of a True Pareto Front or Reference Set (reference points) which are not available for real-world problems. Even if there are practical mechanisms to overcome this issue if we think, again, about a initialisation process with very low computational demand (very few iterations) to select an LLH, these metrics may not be suitable.

Hence, we propose the Relative Performance Per Objective (RPO) indicator which does not demand a True Pareto Front, Reference Set (reference points) or nondominance of solutions, and it is suitable for a initialisation process where we run all LLHs only once and for very few iterations. Assuming that all objective functions need to be minimised, RPO is defined as shown in Algorithm 2 where $F_{m\times l}$ is a $m \times l$ matrix containing the maximum values of all m ($1 \leq i \leq m$) objective functions due to the execution of all l LLHs ($1 \leq k \leq l$).

In Algorithm 2, $NF_{m\times l}$ is a matrix with the same dimensions of $F_{m\times l}$ obtained by normalising each element of $F_{m\times l}$ per objective, $NF^T_{l\times m}$ is its transpose, and $rpo_i \in RPO$. The lower the rpo_i, the better the LLH. The main reasoning behind RPO is to calculate the minimum of the maximum values of the objective functions related to a population.

Let us assume that a problem instance has four objective functions ($m = 4$) and three LLHs (MOEAs) are used within our hyper-heuristics ($l = 3$). Suppose that the maximum values of objective functions of a population obtained by running each LLH

Algorithm 2 The RPO indicator

```
input: $F_{m \times l}$
output: $RPO$

1: $RPO = initialiseRPO()$
2: $NF = initialiseMatrix()$
3: for i = 1 to m do
4:     for k = 1 to l do
5:         $max_i = max_i \cup f_{ik}$
6:     end for
7:     $min_i = findMin(max_i)$
8: end for
9: for i = 1 to m do
10:     for k = 1 to l do
11:         if $min_i = f_{ik}$ then
12:             $nf_{ik} = 0$
13:         else
14:             $nf_{ik} = (f_{ik} - min_i)/min_i$
15:         end if
16:     end for
17: end for
18: $NFT = NF^T$
19: for i = 1 to l do
20:     for k = 1 to m do
21:         $rpo_i = rpo_i + nft_{ik}$
22:     end for
23: end for
24: return $RPO$
```

only once and for very few iterations are as shown in matrix $F_{m \times l}$ below:

$$F_{m \times l} = \begin{pmatrix} 1684.22 & 1681.65 & 1692.38 \\ 9.08 & 8.28 & 10.65 \\ 0.26 & 0.24 & 0.23 \\ 8.11 & 8.15 & 7.22 \end{pmatrix}$$

Matrix $NF_{m \times l}$ is then created (lines 9 to 17 in Algorithm 2) by considering the minimum of the maximum values for each objective independently. For instance, let us consider the first objective function (row 1 of matrix $F_{m \times l}$). The minimum value of objective function is due to the LLH 2 = 1681.65. Thus, this becomes 0 in $NF_{m \times l}$, the remaining elements of the matrix are obtained by normalisation (line 14 in Algorithm 2), and the entire $NF_{m \times l}$ is presented below:

$$NF_{m \times l} = \begin{pmatrix} 0.001528 & 0 & 0.006381 \\ 0.096618 & 0 & 0.286232 \\ 0.130435 & 0.043478 & 0 \\ 0.123269 & 0.128809 & 0 \end{pmatrix}$$

Matrix $NF^T_{l \times m}$ is the transpose of $NF_{m \times l}$ as shown below:

$$NF^T_{l \times m} = \begin{pmatrix} 0.001528 & 0.096618 & 0.130435 & 0.123269 \\ 0 & 0 & 0.043478 & 0.128809 \\ 0.006381 & 0.286232 & 0 & 0 \end{pmatrix}$$

Each row of $NF^T_{l \times m}$ has the normalised values of the objective functions due to each LLH. In order to obtain the *rpo* of each LLH, we simply sum the elements of each row. Hence, these are the *rpo* values for this example: $rpo_1 = 0.35185$, $rpo_2 = 0.172287$, $rpo_3 = 0.296213$. Since the smallest value is due to the LLH 2 (rpo_2), then this is the one chosen to be the first to be executed.

3.2. The balanced exploitation/exploration method

We designed additional tasks to have a balanced solution regarding exploitation × exploration as shown in Algorithm 3. These are policies ($P1$, $P2$, $P3$) that may alter the selected LLH due to roulette wheel in the previous step. Note that some input parameters in Algorithm 3 are handled by these three policies and, as we will explain in Section 5.1, η_r is a parameter that we do not need to tune because its value is chosen based on η_a.

Algorithm 3 The Balanced Exploitation/Exploration Method

```
input: $sel$, $Per$, $\eta_a$, $IQ$, $LLH$, $AQ$
output: $nsel$

1: if $(P1 \vee P2 \vee P3)$ then
2:     $nsel = selectBestPerf(\{LLH \setminus sel\}, AQ)$
3:     if $nsel = null$ then
4:         $nsel = getRandomCheckFailures(\{LLH \setminus sel\}, \eta_r)$
5:     end if
6: else
7:     $nsel = sel$
8: end if
9: if $checkWeakPerfAll(LLH, \eta_r)$ then
10:     $enableLLH(LLH)$
11: end if
12: return $nsel$
```

Since the Reinforcement Learning mechanism (more details in Section 3.3) rewards or penalises the utility values, it is possible that, for a given problem instance, a certain LLH k is selected to run several times and eventually this may be the only selected LLH for this instance resulting in an extreme exploitation situation. This can happen because each time an LLH k has its offspring population accepted, its utility value is rewarded while the others LLHs stay with the same utility value.

In order to avoid this situation, in policy $P1$ we count the number of times (frequency) an LLH k was selected to execute and verify whether the current selected LLH is above a certain limit (η_a), and if this is so, another LLH must be chosen to run. An exception for this is whether the quality improvement of the last time k was run is the best overall compared to the ones of the other LLHs when they were last executed (sequence IQ contains such information). In this case, LLH k is still allowed to run even if its number of executions exceeds η_a.

Policy $P2$ checks whether the last performance of the selected LLH was not good enough so that its population was rejected (check whether $Per_k = \perp$). In policy $P3$, we try to avoid an unnecessary exploration situation, where LLHs that have been continuously failing (i.e. populations not accepted) are ruled out if the number of failures exceed other parameter η_r. However, this elimination is only performed if, at the end, at least two LLHs are enabled for the heuristic selection process in the next decision points.

Any of these three policies being hold is enough to change the previously selected LLH, *sel*. A new LLH, *nsel*, is then chosen based on the best performance up to now (*selectBestPerf* in line 2). In other words, the decision for a certain algorithm, among the ones remaining ($\{LLH \setminus sel\}$), is taken considering the LLH which had its population accepted in highest number of times so far. In the situation where none of the remaining LLHs is superior overall, a random LLH is chosen among them ($\{LLH \setminus sel\}$) but still considering whether the number of failures do not exceed η_r (*getRandomCheckFailures* in line 4).

However, the possible elimination of an LLH of the selection process may not be definitive as presented in other studies [16]. Since we still continue counting the number of times the remaining LLHs poorly performed, it is possible that in a certain moment all the LLHs exceed the established low performance limit (η_r).

Table 1
Some decision points to illustrate the balanced exploitation/exploration method. The * means this is the outcome of the initialisation process.

DP	LLH-RW	f_k	Per_k	IQ_k	$P1$	$P2$	$P3$	LLH-Bal	Pop Acc	$IQ_k - Post$
1	2*	–	–	–	–	–	–	–	YES	–
2	2	1	$\top$	–	F	F	F	2	NO	2×10^{-5}
3	1	1	–	–	F	F	F	1	YES	8×10^{-5}
4	1	2	$\top$	8×10^{-5}	F	F	F	1	NO	1×10^{-5}
5	1	3	$\bot$	1×10^{-5}	T	T	F	2	YES	12×10^{-5}
6	2	3	$\top$	12×10^{-5}	F	F	F	2	NO	0.5×10^{-5}
7	3	1	–	–	F	F	F	3	YES	13×10^{-5}
8	1	1	$\bot$	1×10^{-5}	F	T	F	2	NO	3×10^{-5}
9	2	1	$\bot$	3×10^{-5}	F	T	F	3	YES	15×10^{-5}

Thus, all LLHs are re-enabled (*enableLLH*(*LLH*) in line 10), even those which had already been eliminated, for the next decision points.

Let us assume the same three LLHs (1, 2, 3) as discussed in the last section. Table 1 shows an example of the dynamics considering some decision points. In this table, DP identifies the decision point, LLH-RW means the LLH suggested to run by roulette wheel, f_k means the frequency (number of times) the LLH k has been run, Per_k shows the last performance of the LLH k, IQ_k is the quality improvement of the last time the LLH k was executed, $P1$, $P2$ and $P3$ are the outcomes of the three policies in Algorithm 3, LLH-Bal means the "new" LLH selected to run after the balanced exploitation/exploration mechanism, Pop Acc states if the selected LLH had its population accepted, and $IQ_k - Post$ is the quality improvement of k just after its execution.

We consider $\eta_a = \eta_r = 2$ for this analysis. Firstly note that, in the first decision point, the selected LLH is not the outcome of roulette wheel but rather is the result of the initialisation process previously described and supported by the RPO quality indicator (Section 3.1). We assume that the LLH 2 is the one the initialisation process suggests as shown in the last section. In the second decision point, roulette wheel selects LLH 2, $f_2 = 1$ (column f_k), and the previous performance is True ($Per_2 = \top$; column Per_k) because every population of the first decision point is always accepted. All policies are False and hence there is no change in the selected LLH and LLH 2 is indeed executed (see column *LLH-Bal*). But, we assume that its population is not accepted by our two-level acceptance mechanism. In decision point 3, roulette wheel selects LLH 1, all policies are False, LLH 1 is allowed to run, and its population is accepted. In the next decision point, LLH 1 is again selected by roulette wheel, there is no change in the LLH, but now the population of LLH 1 is not accepted.

In decision point 5, since roulette wheel selects again LLH 1, we see that $f_1 = 3$ (column f_k) is greater than $\eta_a = 2$. But, there is still a possibility to allow LLH 1 to run depending on the quality improvement of the last time LLH 1 was run. But observe that $IQ_1 = 1 \times 10^{-5}$ (column IQ_k) while $IQ_2 = 2 \times 10^{-5}$ (column $IQ_k - Post$). Since the quality improvement of LLH 1 is not better than the one of LLH 2, hence another heuristic must be chosen because policy $P1$ is indeed True. Note that policy $P2$ is also True because of the bad last performance of LLH 1. At this point LLH 2 had 1 population accepted (in the first decision point) and LLH 3 was never run. Hence, the new LLH suggested by the balanced mechanism is LLH 2.

In decision points 6 and 7, LLH 2 and LLH 3 are chosen, respectively, by roulette wheel, all policies are False, there is no change in the suggested LLH, LLH 2 did not succeed in accepting its population, but LLH 3 did succeed. In decision point 8, LLH 1 was selected by roulette wheel and note that its counter is reset since it was above η_a in decision point 5. But, the last performance of LLH 1 is still False (column Per_k) and Policy P2 is satisfied. LLH 2 is then suggested as the new LLH because, at this point, it had two populations accepted (decision points 1 and 5) while LLH 3 had only one (decision point 7).

Finally, in decision point 9, LLH 2 is suggested by roulette wheel and note that its counter is also reset because it was greater than η_a in decision point 6. However, just in the last decision point (8), the population of LLH 2 was not accepted and $Per_2 = \bot$ (column Per_k). To select the new LLH, the balanced strategy realises that both LLH 1 and LLH 3 had one population accepted so far: LLH 1 in decision point 3 and LLH 3 in decision point 7. Therefore, a random choice is made between both heuristics and LLH 3 was chosen to run.

3.3. Reinforcement learning

Reinforcement Learning is a field of Machine Learning which is different from both Supervised and Unsupervised Learning. In Reinforcement Learning, problems involve learning of what should be done (how to map situations to actions) in order to maximise a numerical reward signal [46]. They are closed-loop problems because the application of actions influences the inputs of the system in a later phase.

We use Reinforcement Learning to reward or penalise a certain LLH which is selected to run (action). Hence, if the application of an LLH k results in a population that is accepted by the group decision acceptance methods (see Section 3.4), hence we reward the utility value of the LLH as:

$$\mu'(k) = \mu(k) + \alpha \times \left(1 - \frac{j}{\lfloor 1.5 \times md \rfloor}\right), \tag{2}$$

where $\mu'(k)$ and $\mu(k)$ are the next and the current utility value of the LLH k, respectively, j is the current decision point, md is the maximum number of decision points, $\lfloor . \rfloor$ is the floor function, and α is a parameter in [0, 1].

If the population of a selected LLH is not accepted, it is penalised as follows:

$$\mu'(k) = \mu(k) - \alpha \times \left(1 - \frac{j}{\lfloor 1.5 \times md \rfloor}\right). \tag{3}$$

The utility values impact on the selection of the LLH to run in the next decision point. We only reward or penalise an LLH once per decision point within each hyper-heuristic (HRISE_M and HRISE_R). In Algorithm 1, we see the *penaliseUV* procedure in action (line 33), in case of a negative performance of an LLH. However, within the *groupDecAcc* procedure, *rewardUV* and *penaliseUV* can be called too, depending on whether an LLH must be rewarded or penalised, respectively.

3.4. Group decision-making for acceptance

In group decision-making, several individuals participate in a process where they analyse problems, point out alternative courses of actions, and select from these alternatives one or more solutions [47]. Hence, the idea is to give the responsibility to make decisions to a group of persons rather than a single individual, although this is also an option.

A previous work on selection hyper-heuristic studied ensemble move acceptance methods combining them under a group decision-making framework [17]. Two of the rules discussed in the paper are *responsibility* (also known as authority) and *majority* for making an accept/reject decision using multiple move acceptance methods. The responsibility rule imposes that a single move acceptance method takes the responsibility/authority for the final decision, while the majority rule counts the votes for the accept/reject decision and the majority leads to the final decision. In this study, we consider the majority and responsibility rules to accept or reject the populations generated by the LLHs.

The majority rule is embedded into the HRISE_M hyper-heuristic whereas the responsibility rule derives HRISE_R. In HRMA, we randomly select one of the acceptance methods from the group, and recall that in HRMA we have a random heuristic selection strategy too.

Three move acceptance methods are embedded into HRISE/HRMA where two out of them are new ones created by varying previous solutions. Great Deluge Acceptance (GDA) always accepts improving moves and accepts moves that are worse but which are below a certain threshold which is decreased over time at a linear rate [3,32]. It is an optimisation approach which has been demonstrating good performance in the context of selection hyper-heuristics [48].

We propose a new move acceptance method, called Qualified Late Acceptance (QLA), which is a variation of Late Acceptance (LA) [21,22]. In LA, a comparison is made between a current generated solution and one created s steps before and which is stored in a memory. The parameter s is the memory length and it may influence the performance of the hyper-heuristic. Eventually, if the memory length is high, it is possible that the current generated solution is accepted not because it is too good but because the quality of the solution s steps before may be too low, and hence it is easier to be beaten.

In order to compare to more fitted populations, we store in memory (QA in Algorithm 1) only the quality indicator values of the populations that were accepted in the past. Thus, in the new method QLA, we compare the quality of the current generated solution (population) to the quality of one that has been created and accepted $s \leq j$ steps before, where j refers to decision points. If the current generated solution (population) is no better than the one accepted and created $s \leq j$ steps before, hence we do not take it into account and go on.

We propose another move acceptance method called Mean Qualified Late Acceptance (MQLA). The idea is that a solution (population) is accepted if it is better than the mean value of the quality of the already accepted populations which are stored in the memory.

In HRISE_M, success is where at least two out of the three move acceptance methods agree in accepting the population. In the responsibility version, HRISE_R, only one needs to accept the population. We defined a sequence of applications of the methods in this case as: GDA, QLA, MQLA. Hence, if GDA accepts then the decision of HRISE_R is True. Otherwise, it tries QLA, and a last resource is MQLA. As previously mentioned, in HRMA a random choice is made between GDA, QLA, and MQLA.

Recall that we have a two-level acceptance mechanism where the group decision features (second level) are considered only if the Only Improving criterion (first level) is satisfied. The reasoning behind this strategy is to be more confident when accepting a population and avoid that weak ones to survive further in the process.

4. Problems

All algorithms were evaluated against seven Deb–Thiele–Laumanns–Zitzler (DTLZx, x $=$ 1...7) [49] and nine Walking Fish Group (WFGx, x $=$ 1...9) [50] three-objective benchmark problem instances. The same parameter values for these benchmark functions were used as suggested in [49] and [50] for DTLZ and WFG, respectively.

Still on the benchmark problems, we also assessed three constrained three-objective problem instances as shown in [51]: one Type-1 constrained problem instance (C1-DTLZ1) and two Type-2 constrained problem instances (C2-DTLZ2 and Convex C2-DTLZ2). In Type-1 constrained problem instances, the original True Pareto Front is still optimal, but there is an infeasible barrier in approaching it. Type-2 constrained problems introduce infeasibility to a part of the True Pareto Front and their aim is to test the ability of an algorithm to deal with discontinuities in the True Pareto Front. Moreover, we also considered the first five unconstrained (UFx, x = 1 ... 5) two-objective problem instances from the CEC 2009 Special Session and Competition benchmark [52]. Altogether, we dealt with 24 benchmark problem instances from four problems.

The reasons for choosing the DTLZ and WFG problems and the problem instances above are because they were exactly the same as those used to evaluate HH-RILA [16]. We believe that this option has allowed us to have a more adequate comparison to this recently proposed hyper-heuristic. Furthermore, we selected the DTLZ constrained problem instances because we had already used its unconstrained counterpart, and hence we would like to realise about the performance considering the constrained cases. Problem instances from the CEC 2009 benchmark were considered to increase even more the diversity of case studies.

We also evaluated four different real-world problems as described below. As we will explain, two of these problems are constrained. A total of 15 real-world problem instances from four problems were assessed. Altogether, we evaluated 39 problem instances from eight problems where four problems are real-world applications (unconstrained and constrained). Hence, we have 61.5% of problem instances from benchmark problems and 38.5% of problem instances from real-world applications. This relatively balanced proportion, between benchmark and real-world problems, was one of our goals. In other words, we would not like to have many more problem instances from one type than from another.

4.1. Vehicle crashworthiness problem

Structural optimisation for Vehicle Crashworthiness (VC) criteria is very relevant in the automotive industry [53]. It is difficult to obtain accurate results in these types of problems due to high nonlinearities related to this context. This problem has three objective functions to be minimised: weight (f_1), acceleration characteristics (f_2), and toe-board intrusion (f_3). There are five decision variables and no constraints. The objective functions are shown below.

$$\begin{aligned} f_1(\mathbf{x}) = {} & 1640.2823 + 2.3573285x_1 + 2.3220035x_2 \\ & +4.5688768x_3 + 7.7213633x_4 + 4.4559504x_5 \end{aligned} \tag{4}$$

$$\begin{aligned} f_2(\mathbf{x}) = {} & 6.5856 + 1.15x_1 - 1.0427x_2 + 0.9738x_3 + 0.8364x_4 \\ & -0.3695x_1x_4 + 0.0861x_1x_5 + 0.3628x_2x_4 - 0.1106x_1^2 \\ & -0.3437x_3^2 + 0.1764x_4^2 \end{aligned} \tag{5}$$

$$\begin{aligned} f_3(\mathbf{x}) = {} & -0.0551 + 0.0181x_1 + 0.1024x_2 + 0.0421x_3 \\ & -0.0073x_1x_2 + 0.024x_2x_3 - 0.0118x_2x_4 - 0.0204x_3x_4 \\ & -0.008x_3x_5 - 0.0241x_2^2 + 0.0109x_4^2 \end{aligned} \tag{6}$$

We created four problem instances related to VC considering the original one (three-objective) and by combining pairs of objectives: VC1 $= \{f_1, f_2, f_3\}$; VC2 $= \{f_1, f_2\}$; VC3 $= \{f_1, f_3\}$; and VC4 $= \{f_2, f_3\}$.

4.2. Water resource planning problem

Water Resource (WR) systems are crucial for a proper developed of a nation. However, issues such as inappropriate and/or degraded infrastructure, excessive withdrawals of river flows, pollution from industrial and agricultural activities are still impediments for regions around the world to have even basic drinking water and sanitation needs [54]. This problem has five objective functions that all need to be minimised: the drainage network cost (f_1), the storage facility cost (f_2), the treatment facility cost (f_3), the expected flood damage cost (f_4), and the expected economic loss due to flood (f_5). There are seven constraints and three decision variables [55]. The five objective functions are shown below.

$$f_1(\mathbf{x}) = 106780.37 \times (x_2 + x_3) + 61704.67 \tag{7}$$

$$f_2(\mathbf{x}) = 3000x_1 \tag{8}$$

$$f_3(\mathbf{x}) = \frac{305700 \times 2289x_2}{(0.06 \times 2289)^{0.65}} \tag{9}$$

$$f_4(\mathbf{x}) = 250 \times 2289 \times e^{(-39.75x_2+9.9x_3+2.74)} \tag{10}$$

$$f_5(\mathbf{x}) = 25 \times (\frac{1.39}{x_1x_2} + 4940x_3 - 80) \tag{11}$$

We created five problem instances related to WR considering the original one (five-objective) and by combining four out of the five objectives: WR1 = $\{f_1, f_2, f_3, f_4, f_5\}$; WR2 = $\{f_1, f_2, f_3, f_4\}$; WR3 = $\{f_1, f_2, f_4, f_5\}$; WR4 = $\{f_1, f_3, f_4, f_5\}$; and WR5 = $\{f_2, f_3, f_4, f_5\}$.

4.3. Car side impact problem

Another problem related to the automative industry, Car Side (CS) impact deals with the optimisation of a vehicle side impact crashworthiness, but the formulation of this problem presents 10 constraints [56]. It has three objective functions that need to be minimised: the weight of car ($f1$), the pubic force experienced by a passenger (f_2), and the average velocity of the V-Pillar responsible for withstanding the impact load (f_3). There are seven decision variables and the objective functions are as follows.

$$f_1(\mathbf{x}) = 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 0.00001x_6 + 2.73x_7 \tag{12}$$

$$f_2(\mathbf{x}) = 4.72 - 0.5x_4 - 0.19x_2x_3 \tag{13}$$

$$f_3(\mathbf{x}) = 0.5 \times (27.03 - 0.674x_1x_2 - 0.67275x_2 - 0.489x_3x_7 - 0.843x_5x_6) \tag{14}$$

Three problem instances were created in addition to the original one: CS1 = $\{f_1, f_2, f_3\}$; CS2 = $\{f_1, f_2\}$; and CS3 = $\{f_1, f_3\}$.

4.4. Space applications communication problem

Space Applications (SP), such as satellites, rockets, and balloons, are real-time systems. Data that are gathered by the their various computing subsystems serve not only to make decisions automatically during the operation of the systems but, naturally, they must be sent to ground stations on the Earth's surface so that controllers of such systems can perceive their health and interfere if something is wrong. Furthermore, scientists can make studies based on the observations collected by the applications.

This problem refers to real-time TeleMetry (TM) data transmission from a hard X-ray imaging telescope which is to be launched by a balloon and will operate between 40 to 42 km of altitude [57,58]. The On-Board Data Handling Subsystem (OBDH) is responsible for acquiring, formatting, and transmitting to the Ground Station (GS) all TM data generated by several subsystems (X-ray Camera (XRC), Attitude Control Subsystem (ACS)) of the space segment. The OBDH continuously stores on-board and sends to the GS all TM data it obtains. The main goal is trying to minimise the total time (associated with transmission and propagation delays) of these data that are stored on-board and, at the same time, are visualised on the GS in real-time.

There are three objective functions: minimise the total time related to the delivery of scientific data to the GS (f_1), minimise the total time related to the delivery of housekeeping data (OBDH + ACS) to the GS (f_2), and maximise the total amount of data transmitted to the GS (f_3). This problem has no constraints while it has four decision variables: amount of scientific data TM packet (x_1); amount of housekeeping data TM packet from the OBDH (x_2); amount of housekeeping data TM packet from the ACS (x_3); and amount of event packets generated by the XRC (x_4). The objective functions are described in the sequence where f_3 is formulated as a minimisation problem too.

$$f_1(\mathbf{x}) = x_1 \times (x_4 + \frac{48x_4 + 96}{115200} + \frac{48x_4 + 272}{500000}) + 0.300133 \tag{15}$$

$$f_2(\mathbf{x}) = \frac{115264x_2 + 115328x_3}{115200} + 0.300133 \tag{16}$$

$$f_3(\mathbf{x}) = 300 - (x_1 + x_2 + x_3) \tag{17}$$

Three problem instances were created: the original instance, SP1 = $\{f_1, f_2, f_3\}$; SP2 = $\{f_1, f_3\}$; and SP3 = $\{f_2, f_3\}$.

5. Experimental results

Our hyper-heuristics were implemented based on two frameworks: version 5.6 of jMetal [59] and jMetalHyperHeuristicHelper [60]. As for the experimental evaluation, we considered as LLHs three MOEAs: NSGA-II [18], IBEA [19], and SPEA2 [20]. We chose these MOEAs due to their popularity and also because several other selection hyper-heuristics [13,15,16] have made use of them, and so this is more suitable for the analysis. Selection of individuals was binary tournament, we used Simulated Binary Crossover (SBX) [61] with probability 0.9 and distribution index 20, and Polynomial mutation with probability $1/n$ (n = number of parameters) and distribution index 20. We compared our approaches to three other hyper-heuristics (HH-CF, HH-ALL, and HH-RILA), and the three MOEAs run in isolation.

We executed all approaches for each problem instance for 100,000 evaluations, population size (z) was 100, and trials were 30. Hence, for our hyper-heuristics, HH-CF, HH-ALL, NSGA-II, IBEA, SPEA2, the maximum number of iterations (mi) was 1,000. Within our nonuniform strategy, the first selected LLHs ran for 250 iterations and in the later decision points they ran for 100 iterations (but recall that even later it is possible a huge number of iterations/LLH, i.e. 250). With respect to HH-RILA, each selected LLH ran for 10 iterations as suggested in their study [16]. The maximum number of decision points (md) for our hyper-heuristics was 7, and in HH-CF it was 25 as previously mentioned [13] as well as this is the case for HH-ALL. Table 2 summarises the values of the main parameters used.

The choice of values for the common parameters was driven by other studies where some works use 50,000, 75,000, 100,000 evaluations [16], depending on the problem instance, and others use 100,000 evaluations [62]. We decided to use a fixed default value for all instances equals to 100,000 evaluations to accomplish a more uniform comparison. This choice is not related to any previous analysis of convergence of the algorithms. As well as a population of 100 solutions and 30 trials are quite common values in several other studies in the literature.

The tuning of parameters of our hyper-heuristics are presented in Section 5.1 while the meaning of the parameters of HH-RILA [16] and HH-CF [13], shown in Table 2, are described in their respective articles. Moreover, α in our hyper-heuristics has a different meaning of α in HH-CF. All experiments were performed on an Intel Xeon 3.5 GHz processor with 32 GB of RAM memory and Windows 10 Operating System. All populations (`VAR` files) of all algorithms, the respective values of objective functions (`FUN` files), all True and True Known (see remarks in Section 5.2) Pareto Fronts (`.pf` files) are stored in [63].

Table 2
Experimental evaluation: values of parameters.

Common parameters	
Number of evaluations	100,000
Population size	100
Trials	30
Specific parameters	
HRISE_M, HRISE_R, HRMA	
Iterations/LLH	250 or 100
Maximum iterations	1,000
Maximum decision points	7
α	1.0 (HRISE_M) / 0.1 (HRISE_R)
η_a	2 (HRISE_M) / 3 (HRISE_R)
η_r	2 (HRISE_M) / 1 (HRISE_R)
γ	0.000075
HH-RILA	
Iterations/LLH	10
τ	0.9
m	3.0
K	3
Δ_v	0.0075
HH-CF	
Maximum iterations	1,000
Maximum decision points	25
α	100
HH-ALL	
Maximum iterations	1,000
Maximum decision points	25
NSGA-II, IBEA, SPEA2	
Maximum iterations	1,000
Maximum decision points	1

5.1. Parameter tuning

There are two main parameters to tune within our approaches: α and η_a. The η_r parameter was selected based on η_a:

$$\eta_r = \begin{cases} \eta_a, & \text{if} \quad \eta_a = 2 \text{ (more flexible)}, \\ \eta_a - 2, & \text{if} \quad \eta_a > 2 \text{ (less flexible)} \end{cases}$$

When $\eta_r = \eta_a$, we say that there is a greater flexibility regarding the LLHs which perform worse, while $\eta_r = \eta_a - 2$ means less flexibility because we eliminate an LLH with less number of failures (populations not accepted). Other parameters were selected based on previous studies. For instance, $\gamma = 0.0075/100 = 0.000075$ since 0.0075 was the value suggested in the HH-RILA approach [16]. But in HH-RILA, the comparison is made to the hypervolume of the last population (regardless whether it was accepted or not) while we compare to the last accepted population. Thus, we gave a lower margin to avoid being too rigid to consider a population fitted.

The parameters could take the following values: $\alpha = \{0.1, 0.4, 0.6, 1.0\}$, $\eta_a = \{2, 3, 4\}$. With respect to HRISE_M and HRISE_R, we ran all problem instances from two theoretical benchmarks (DTLZ, WFG) and all problem instances from VC (only) for 30 trials, for every combination $\langle \alpha, \eta_a \rangle$.

We took into account two levels of normalisation of the hypervolume indicator. The front-normalised hypervolume, h, is obtained via the normalisation of the "raw" hypervolume based on the minimum and maximum values of the objective functions of a problem instance. From this point onward, we will denote it simply as *hypervolume*, h, and as higher its value, the better.

As for the cross-domain analysis, where we aim at realising about the generalisation features of hyper-heuristics, we used the normalised (front-normalised) hypervolume, h_N, as defined below [16]:

$$h_N = \frac{h^{max}_{(\forall a,p)} - \overline{h_{(a_i,p)}}}{h^{max}_{(\forall a,p)} - h^{min}_{(\forall a,p)}} \tag{18}$$

where $h^{max}_{(\forall a,p)}$ and $h^{min}_{(\forall a,p)}$ are the maximum and minimum values, respectively, of the hypervolume, h, due to all algorithms a for a problem instance p, and $\overline{h_{(a_i,p)}}$ is the average value of the hypervolume due to algorithm a_i for p. From this point onward, we will denote it simply as *normalised hypervolume*, h_N. However, note that the formulation of h_N is like a maximisation problem (maximise hypervolume) is turned into a minimisation problem. Hence, the lower the value of h_N, the better.

In order to decide the values of parameters for our approaches, we used h_N. Tuning of HRISE_M produced as a result: $\{\alpha = 1.0, \quad \eta_a = 2\}$. The choice for HRISE_R was: $\{\alpha = 0.1, \quad \eta_a = 3\}$.

5.2. Cross-domain performance analysis

One of the main claims of hyper-heuristics supporters is generalisation, i.e. their ability to get better results across several different problems rather than a single problem domain. Hence, the cross-domain performance analysis is a very suitable evaluation tool in this context. Table 3 presents the results of the cross-domain performance analysis using as metric the normalised hypervolume, h_N, for all benchmark problems. In Table 4, we present the results for all real-world and for all (benchmark + real) problems.

In Table 3 and also in some figures later, DTx, WFx identify the DTLZ and WFG problem instances, respectively, C1DT1 is C1-DTLZ1, C2DT2 is C2-DTLZ2, CONVC2DT2 is the Convex C2-DTLZ2, and UFx identifies the CEC 2009 unconstrained problem instances. Moreover, $\overline{h^{DT}_N}$, $\overline{h^{WF}_N}$, $\overline{h^{CDT}_N}$, and $\overline{h^{UF}_N}$ mean the average values of h_N for DTLZ, WFG, Constrained DTLZ, and CEC 2009 problem instances, respectively. Furthermore, in Table 3, $\overline{h^{BEN}_N}$ is the average value considering all 24 benchmark problem instances.

In Table 4 we see the identification of each real-world problem instance, the average value of the problem instances, $\overline{h^{y}_N}$, the mean of all 15 real-world problem instances, $\overline{h^{REA}_N}$, and finally the average value of all 39 (benchmark + real) problem instances, $\overline{h^{APR}_N}$. In grey background we show the top performance approaches, where in bold it is the best algorithm while in italics it is the second best.

Note that the differences are small but this is probably a consequence of the double normalisation approach we took regarding the calculation of the hypervolume. As for the DTLZ problem, we see that HRISE_R is the best overall followed by HH-CF. The good performance of HH-CF, as being even superior to SPEA2, contradicts previous results [16]. One possible explanation for this relies on the fact that HH-CF, in many problem instances including some DTLZ ones, selected SPEA2 as the LLH to run and this may be boosted its results regarding all DTLZ problem instances. As for the WFG problem, we have observed similar results as in previous studies where IBEA presents the best performance followed by HH-RILA. Regarding the Constrained DTLZ problem, HRISE_R is again the top algorithm while HRMA is the second. Regarding UF, NSGA-II gets the top rank and HH-RILA is the second best approach. Across all benchmark problem instances, our hyper-heuristics are the best where HRISE_R is the top algorithm followed by HRISE_M.

Considering the real-world problems, in order to obtain the reference points for hypervolume calculation, we created the so-called *True Known Pareto Front* where, for each problem instance, we joined all final populations of all algorithms after the 30 trials,

Table 3
Cross-domain analysis for the benchmark problem instances: h_N.

	HRISE_M	HRISE_R	HRMA	IBEA	NSGA-II	SPEA2	HH-ALL	HH-CF	HH-RILA
DT1	0.009498327	0.009855810	0.011779675	0.8106395792	0.032925219	0.007965541	0.10512578	0.024426446	0.196691440
DT2	0.012656553	0.008762936	0.008223576	0.0025222595	0.203592440	0.136879273	0.19267656	0.136956350	0.002891709
DT3	0.096110418	0.100118774	0.136844618	0.9980869821	0.095983292	0.022164646	0.31727926	0.063195181	0.479082718
DT4	0.141933089	0.029555748	0.090898478	0.5028700174	0.131192721	0.266326108	0.38443364	0.121693281	0.085819874
DT5	0.002407721	0.002465813	0.002943352	0.0032569910	0.002837598	0.007214540	0.18038798	0.007648043	0.002931086
DT6	0.087432499	0.078917752	0.039584897	0.1198103250	0.072854251	0.068947108	0.15609049	0.022206087	0.080774553
DT7	0.121098815	0.121886602	0.176921627	0.4149621028	0.050817711	0.064805067	0.28064893	0.061841454	0.044582010
$\overline{h_N^{DT}}$	0.067305346	**0.050223348**	0.066742318	0.4074497510	0.084314747	0.082043183	0.23094895	*0.062566692*	0.127539056
WF1	0.285288324	0.347529879	0.284062911	0.1992446777	0.477788012	0.747589962	0.56276940	0.788892232	0.221513702
WF2	0.127902123	0.110695775	0.128248730	0.1115329776	0.080930273	0.080735816	0.23142207	0.220539693	0.063024615
WF3	0.028457138	0.028157958	0.027973307	0.0144763318	0.034144581	0.136818998	0.23622878	0.130049702	0.016318824
WF4	0.043530925	0.048939620	0.084892734	0.0033677056	0.229942212	0.616810382	0.43369575	0.585277348	0.014671879
WF5	0.081245637	0.054177364	0.082759313	0.0031328286	0.348534009	0.789952394	0.47735643	0.800485739	0.020694440
WF6	0.048597034	0.045959387	0.043555438	0.0258769421	0.122126005	0.230140482	0.21113077	0.245983949	0.027588763
WF7	0.020039467	0.023234034	0.027551990	0.0020431587	0.158677383	0.450320014	0.30476881	0.447787727	0.007044674
WF8	0.048436959	0.046163746	0.041767526	0.0031774407	0.109539464	0.217408128	0.20140834	0.220433862	0.014499885
WF9	0.114945835	0.115964342	0.102533787	0.0989629622	0.189197247	0.242370556	0.21015509	0.237506659	0.135830158
$\overline{h_N^{WF}}$	0.088715938	0.091202456	0.091482860	**0.0513127806**	0.194542132	0.390238526	0.31877061	0.408550768	*0.057909660*
C1DT1	0.008509325	0.009095905	0.011947385	0.8252304735	0.034233329	0.007034517	0.09251673	0.022065238	0.473388163
C2DT2	0.016569728	0.005081225	0.007347848	0.0025739009	0.175824202	0.116927954	0.20685660	0.118332161	0.003223532
CONVC2DT2	0.020097068	0.010533541	0.016599613	0.0031356329	0.166692875	0.110234904	0.19716643	0.109982334	0.009079216
$\overline{h_N^{CDT}}$	0.015058707	**0.008236890**	*0.011964949*	0.2769800024	0.125583469	0.078065792	0.16551326	0.083459911	0.161896970
UF1	0.105793909	0.114548708	0.108522859	0.1143688794	0.085169030	0.124261236	0.46938010	0.116029879	0.108326338
UF2	0.045640194	0.032922704	0.044036570	0.0369882745	0.022531491	0.032339746	0.20464743	0.031618316	0.029186501
UF3	0.580816293	0.526752067	0.554291843	0.6082855449	0.293492486	0.335832924	0.63212691	0.379939029	0.287422316
UF4	0.028721956	0.030330975	0.030794960	0.0495172532	0.008504696	0.019580812	0.37418148	0.017101346	0.023916680
UF5	0.431950350	0.389696429	0.491216060	0.4723252253	0.401558306	0.507793659	0.63243150	0.503044967	0.462521909
$\overline{h_N^{UF}}$	0.238584540	0.218850177	0.245772458	0.2562970354	**0.162251202**	0.203961675	0.46255349	0.209546708	*0.182274749*
$\overline{h_N^{BEN}}$	*0.104486654*	**0.095472796**	0.106470796	0.2260995194	0.147045368	0.222518949	0.30395355	0.225543209	0.117126041

Table 4
Cross-domain analysis for the real-world problem instances and for all (benchmark + real) problem instances: h_N.

	HRISE_M	HRISE_R	HRMA	IBEA	NSGA-II	SPEA2	HH-ALL	HH-CF	HH-RILA
VC1	0.015012307	0.013628537	0.017481842	0.0569451460	0.008349051	0.026181470	0.27573145	0.056483475	0.015936118
VC2	0.001319560	0.001357898	0.001250924	0.0046165526	0.001882698	0.002537431	0.11803921	0.002339636	0.001428423
VC3	0.055187005	0.067916339	0.083305384	0.1289743811	0.025055766	0.052064056	0.14929741	0.127610995	0.025298341
VC4	0.007708097	0.010273151	0.013475085	0.0224313831	0.004521357	0.012924844	0.22817754	0.011177095	0.003216150
$\overline{h_N^{VC}}$	0.019806742	0.023293981	0.028878309	0.0532418657	**0.009952218**	0.023426950	0.19281140	0.049402800	*0.011469758*
WR1	0.027937434	0.029167885	0.026731115	0.2425430335	0.037294335	0.041923049	0.21545727	0.038285723	0.052779993
WR2	0.064559746	0.072697526	0.064185063	0.0490614413	0.240510363	0.131559224	0.17207845	0.136286873	0.054281490
WR3	0.024679457	0.017986256	0.030043657	0.2769798906	0.021755999	0.095107382	0.16970055	0.059714725	0.022207198
WR4	0.005607416	0.004176212	0.006604715	0.0147733811	0.068454363	0.004697338	0.07354753	0.004568826	0.005187584
WR5	0.023693619	0.024568723	0.031057522	0.2799361234	0.023248492	0.057375523	0.15574944	0.043900160	0.026801098
$\overline{h_N^{WR}}$	**0.029295534**	*0.029719320*	0.031724414	0.1726587740	0.078252711	0.066132503	0.15730665	0.056551261	0.032251473
CS1	0.026356768	0.025725774	0.031364345	0.0248840790	0.077695569	0.056646031	0.25161217	0.055379358	0.027199804
CS2	0.022463122	0.017201529	0.021148784	0.0020043514	0.031447442	0.047567388	0.16108329	0.045894507	0.008558193
CS3	0.018558693	0.016649183	0.017549538	0.0009053537	0.014937509	0.023561818	0.16865857	0.023240025	0.007479879
$\overline{h_N^{CS}}$	0.022459528	0.019858829	0.023354222	**0.0092645947**	0.041360173	0.042591746	0.19378468	0.041504630	*0.014412625*
SP1	0.018759884	0.020190636	0.019735483	0.1014069934	0.032734573	0.031436401	0.11038464	0.030378643	0.036287694
SP2	0.005107621	0.004783941	0.005350405	0.0009710367	0.009412362	0.007053115	0.11392940	0.006984647	0.002001008
SP3	0.002009001	0.001665776	0.002182971	0.0015161120	0.006987187	0.001623675	0.09318753	0.001628722	0.001956825
$\overline{h_N^{SP}}$	**0.008625502**	*0.008880118*	0.009089620	0.0346313807	0.016378041	0.013371063	0.10583385	0.012997337	0.013415176
$\overline{h_N^{REA}}$	*0.021263982*	0.021865958	0.024764456	0.0805299506	0.040285804	0.039483916	0.16377563	0.042924894	**0.019374653**
$\overline{h_N^{APR}}$	*0.072477934*	**0.067162473**	0.075045280	0.1701112237	0.105983997	0.152120859	0.25003897	0.155305396	0.079529353

Table 5
Statistical comparison considering all (benchmark + real) problem instances: h. Caption: Comp = Comparison; Res = Result; p = p-value.

Comp	Res	p
HRISE_M × HRISE_R	$\sim$	8.903E−01
HRISE_M × HRMA	$\sim$	9.180E−01
HRISE_M × IBEA	$>$	1.105E−09
HRISE_M × NSGA-II	$\sim$	2.937E−01
HRISE_M × SPEA2	$>$	9.811E−03
HRISE_M × HH-ALL	$>$	3.996E−14
HRISE_M × HH-CF	$>$	2.812E−03
HRISE_M × HH-RILA	$\sim$	2.772E−01
HRISE_R × HRMA	$\sim$	8.297E−01
HRISE_R × IBEA	$>$	3.499E−10
HRISE_R × NSGA-II	$\sim$	2.200E−01
HRISE_R × SPEA2	$>$	4.948E−03
HRISE_R × HH-ALL	$>$	7.992E−15
HRISE_R × HH-CF	$>$	1.264E−03
HRISE_R × HH-RILA	$\sim$	2.144E−01
HRMA × IBEA	$>$	1.777E−09
HRMA × NSGA-II	$\sim$	3.209E−01
HRMA × SPEA2	$>$	1.289E−02
HRMA × HH-ALL	$>$	6.128E−14
HRMA × HH-CF	$>$	3.846E−03
HRMA × HH-RILA	$\sim$	2.958E−01
IBEA × NSGA-II	$<$	2.819E−07
IBEA × SPEA2	$<$	1.955E−04
IBEA × HH-ALL	$\sim$	2.937E−01
IBEA × HH-CF	$<$	5.678E−04
IBEA × HH-RILA	$<$	1.196E−06
NSGA-II × SPEA2	$\sim$	1.637E−01
NSGA-II × HH-ALL	$>$	3.685E−11
NSGA-II × HH-CF	$\sim$	7.056E−02
NSGA-II × HH-RILA	$\sim$	9.180E−01
SPEA2 × HH-ALL	$>$	1.605E−07
SPEA2 × HH-CF	$\sim$	7.826E−01
SPEA2 × HH-RILA	$\sim$	2.200E−01
HH-ALL × HH-CF	$<$	5.645E−07
HH-ALL × HH-RILA	$<$	3.520E−10
HH-CF × HH-RILA	$\sim$	1.118E−01

obtained the nondominated solutions, and removed the repeated ones. Results for the VC problem are in line with previous studies [16] where NSGA-II is in the top position followed by HH-RILA. In two problems, WR and SP, HRISE_M and HRISE_R presents the best results (HRISE_M is the best in both problems). For all real-world problem instances, HH-RILA is the best being slightly superior than HRISE_M. However, considering all 39 (benchmark + real) problem instances ($\overline{h_N^{APR}}$), HRISE_R is the best approach followed by HRISE_M, and HRMA.

In Table 5, we present a statistical evaluation considering the 39 problem instances altogether but we used the hypervolume, h, at this time where the higher the value, the better. We applied a two-tailed permutation test (conditional inference procedure) [64] for multi-group comparison with significance level equal to 0.05. In Table 5, "$>$" means the leftmost algorithm is significantly better than the rightmost one, "$<$" means the leftmost algorithm is significantly worse, "$\sim$" means no significant difference, and E in the p-values (p) represents the scientific notation (standard form).

In accordance with the statistical outcomes, the top three approaches ranked based on the cross-domain analysis, i.e. our three hyper-heuristics, are not better than HH-RILA and NSGA-II. However, HRISE_R, HRISE_M, and HRMA are superior than all remaining algorithms, including SPEA2 and HH-CF, while there is a tie if we make a pairwise comparison between HH-RILA, NSGA-II, SPEA2, and HH-CF. In other words, between these last four algorithms in a pairwise manner (i.e. NSGA-II × SPEA2, NSGA-II × HH-CF, NSGA-II × HH-RILA, SPEA2 × HH-CF, SPEA2 × HH-RILA, HH-CF × HH-RILA), there is no statistical difference considering h as a metric. Thus, from the statistical point of view, we may not claim that our hyper-heuristics are truly better than HH-RILA and NSGA-II, but we might say that they are at least slightly better because of the remarks we have just presented above. We also compared all algorithms via the normalised ϵ indicator (ϵ_N) which shares the main reasoning behind the normalised hypervolume. However, ϵ_N is calculated in the traditional manner in the context of minimisation problems since as lower ϵ, and also ϵ_N, the better. In Table 6, we show a summarised version of the results presenting only the average values (ϵ_N^y). As in the case of hypervolume, for the real-world problems, we used the True Known Pareto Front obtained along the execution of all algorithms.

Again HH-CF performs well for the DTLZ problem where it obtains the top place followed by HRISE_R. With respect to WFG, we have the same pattern as in the case of h_N while for the Constrained DTLZ problem we note a swap between the top contenders: HRMA is the best followed by HRISE_R. As for the UF problem we notice the same classification as in the hypervolume case, and overall, for the benchmark problem instances, HH-RILA is slightly better than HRISE_R. Considering the real-world problems, we might say that our approaches are even better than in the previous analysis, where in three problems (WR, CS, and SP) they are the two best solutions, as well as HRISE_M presents the best results for all real-world problems followed by HRISE_R. Considering all 39 (benchmark + real) problem instances, HRISE_R is the best strategy followed by HH-RILA, HRISE_M, and HRMA.

The conclusion of all these results is that our hyper-heuristics obtained the best results when comparing to HH-CF, HH-ALL, and the MOEAs run in isolation. Regarding the comparison to the recently proposed HH-RILA, even if we saw no statistical significant difference when evaluating the hypervolume, h, we may say that our hyper-heuristics were at least slightly better than HH-RILA taking into account the pairwise comparison between HH-RILA and some other approaches as we have previously pointed out. Overall, taking into account both quality indicators, HRISE_R was the best of all algorithms based on the results of the cross-domain evaluations.

However, we should mention that we still need to perform more experiments considering other benchmark and real-world problems to definitely conclude on the best performance of the approaches. In other words, more problems are required to conclude about generalisation. This is known as a threat to external validity related to the population in the context of controlled experiments [65,66].

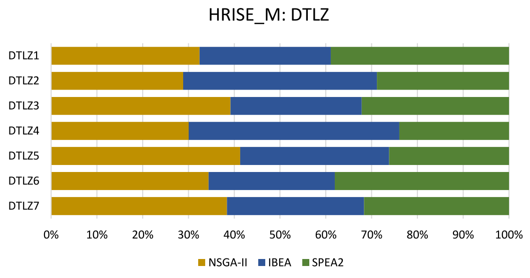
5.3. Evaluation of the heuristic selection mechanisms: Utilisation rate

In this section, we aim at evaluating the heuristic selection mechanisms implemented within our three hyper-heuristics and HH-RILA, the most recent and which got the closest performance in comparison to our proposals. We use the utilisation rate as our main metric over all 30 trials. Figs. 3 and 4 present the results regarding the benchmark problems DTLZ and WFG, respectively. At first, it seems that HH-RILA eventually possesses more "intelligence" than our solutions since we clearly see that it favours much more some MOEAs in certain problem instances which may be a good approach. But, the initialisation process of HH-RILA has a mechanism that may rule out an LLH, depending on its performance, for all the remaining iterations. As we will mention, sometimes this is interesting and in other situations it is not.

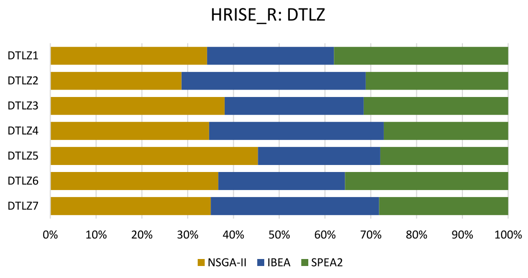
Let us consider the DTLZ6 problem instance. HH-RILA completely eliminated NSGA-II and all the iterations were split between IBEA and SPEA2. But, in Fig. 5(a), we see that the mean

Table 6
Cross-domain analysis for all (benchmark + real) problem instances: ϵ_N.

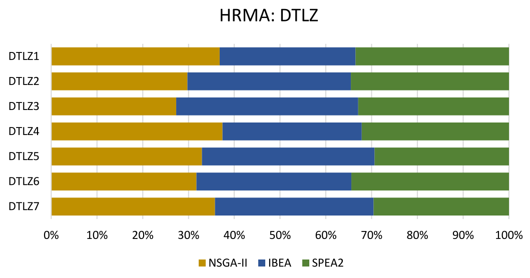
	HRISE_M	HRISE_R	HRMA	IBEA	NSGA-II	SPEA2	HH-ALL	HH-CF	HH-RILA
$\overline{\epsilon_N^{DT}}$	0.060932882	*0.040354858*	0.058350169	0.259914166	0.043035902	0.056226919	0.18275938	**0.030316965**	0.050506416
$\overline{\epsilon_N^{WF}}$	0.067613460	0.068323852	0.069926535	**0.050738515**	0.172394917	0.273261452	0.25869282	0.286147075	*0.052807338*
$\overline{\epsilon_N^{CDT}}$	0.020761323	*0.017383621*	**0.017088303**	0.225705532	0.069568751	0.040420909	0.14444991	0.041710692	0.137120840
$\overline{\epsilon_N^{UF}}$	0.236468439	0.214543273	0.224203583	0.245104048	**0.130358635**	0.162392234	0.49241086	0.189553516	*0.153678809*
$\overline{\epsilon_N^{BEN}}$	0.094986561	*0.084261079*	0.092086368	0.174111776	0.113054374	0.157756892	0.27095646	0.160851754	**0.083690313**
$\overline{\epsilon_N^{VC}}$	0.072032179	0.085378983	0.097708036	0.170663040	**0.051386684**	0.066223695	0.27745860	0.144490275	*0.059446475*
$\overline{\epsilon_N^{WR}}$	**0.061375688**	*0.064445890*	0.070449603	0.282405124	0.135166963	0.096526192	0.19483680	0.091871970	0.092797546
$\overline{\epsilon_N^{CS}}$	0.036901571	**0.027387574**	*0.029413080*	0.047980472	0.030080473	0.031345462	0.21765911	0.029852972	0.042643926
$\overline{\epsilon_N^{SP}}$	0.024417463	*0.022879396*	**0.022720465**	0.063517357	0.035932933	0.025066251	0.11091780	0.024946384	0.028292059
$\overline{\epsilon_N^{REA}}$	**0.051930951**	*0.054303086*	0.059965386	0.161944751	0.071961451	0.061117392	0.20464994	0.080114601	0.060972106
$\overline{\epsilon_N^{APR}}$	0.078426711	**0.072738774**	0.079732144	0.169432151	0.097249404	0.120587853	0.24545395	0.129799003	*0.074952541*



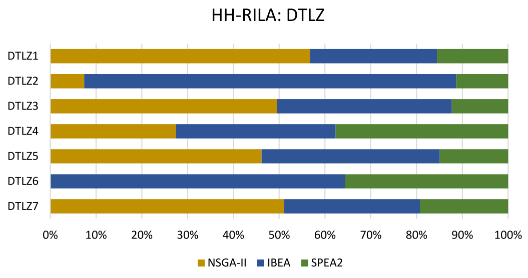
(a) Utilisation Rate: HRISE_M



(b) Utilisation Rate: HRISE_R

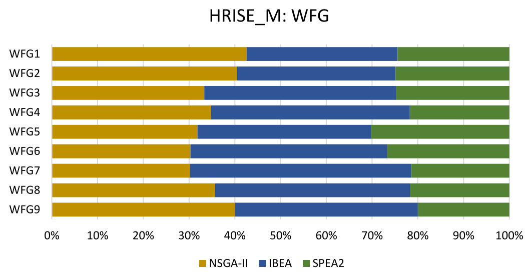


(c) Utilisation Rate: HRMA

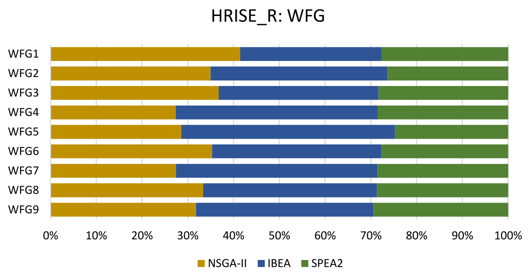


(d) Utilisation Rate: HH-RILA

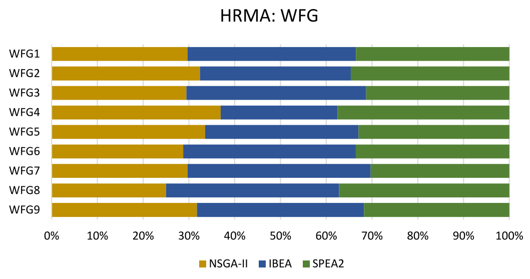
Fig. 3. Utilisation Rate: DTLZ problem instances.



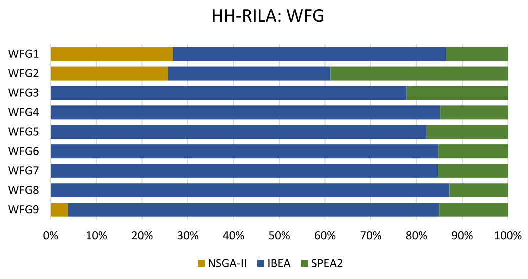
(a) Utilisation Rate: HRISE_M



(b) Utilisation Rate: HRISE_R

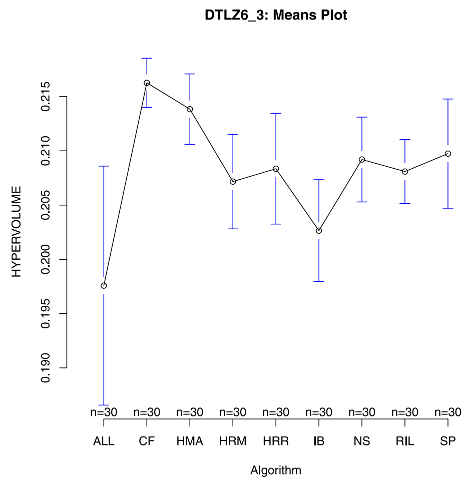


(c) Utilisation Rate: HRMA

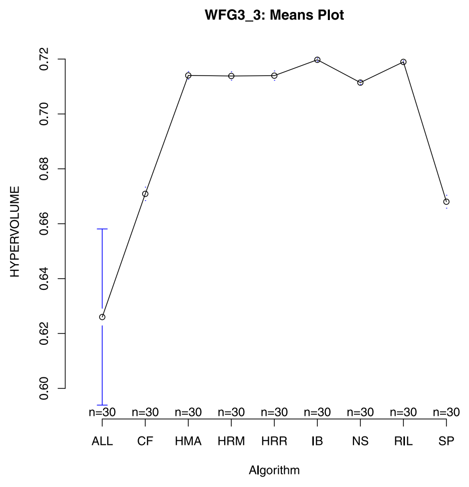


(d) Utilisation Rate: HH-RILA

Fig. 4. Utilisation Rate: WFG problem instances.

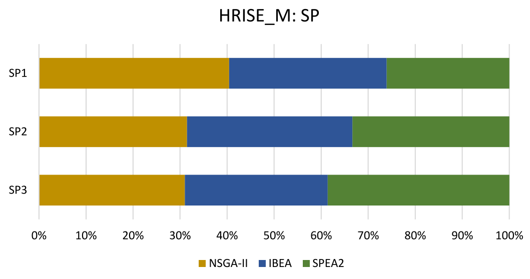


(a) DTLZ6

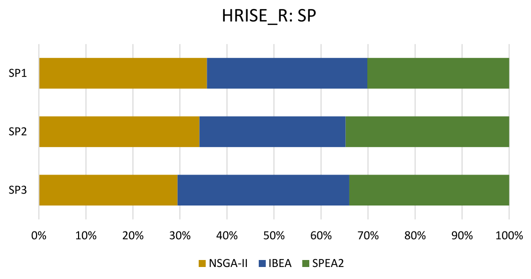


(b) WFG3

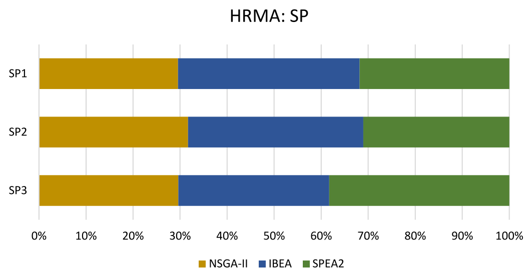
Fig. 5. Means plots of the hypervolume (h) over 30 trials: DTLZ6 and WFG3. The number after "_" in the problem instance nomenclature is the number of objective functions, and the circle is the mean value. Caption: ALL = HH-ALL; CF = HH-CF; HMA = HRMA; HRM = HRISE_M; HRR = HRISE_R; IB = IBEA; NS = NSGA-II, RIL = HH-RILA; SP = SPEA2.



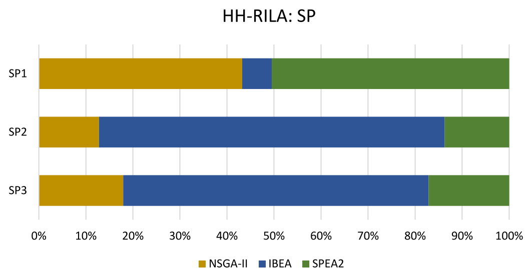
(a) Utilisation Rate: HRISE_M



(b) Utilisation Rate: HRISE_R

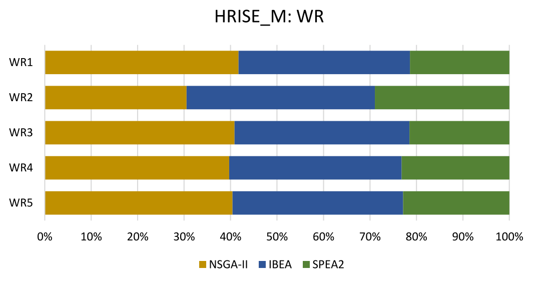


(c) Utilisation Rate: HRMA

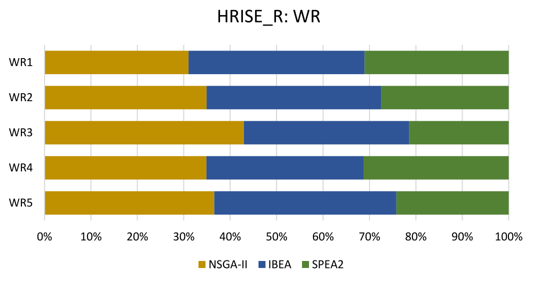


(d) Utilisation Rate: HH-RILA

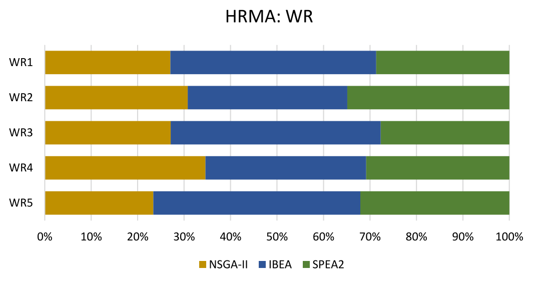
Fig. 6. Utilisation Rate: SP problem instances.



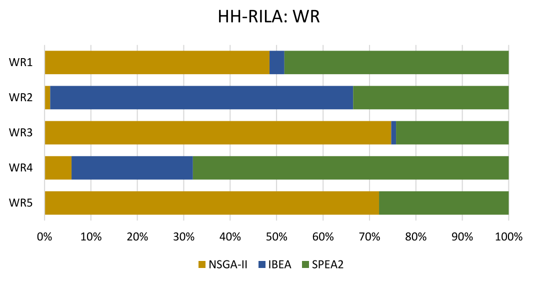
(a) Utilisation Rate: HRISE_M



(b) Utilisation Rate: HRISE_R

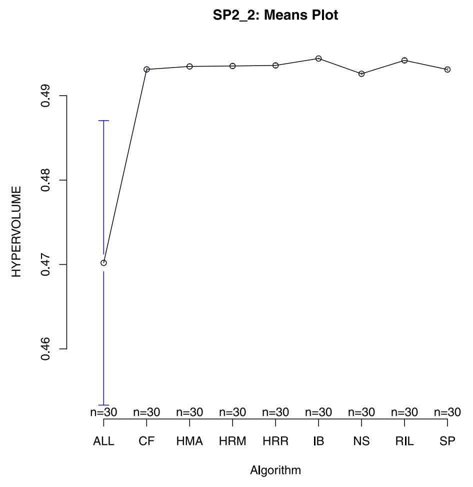


(c) Utilisation Rate: HRMA

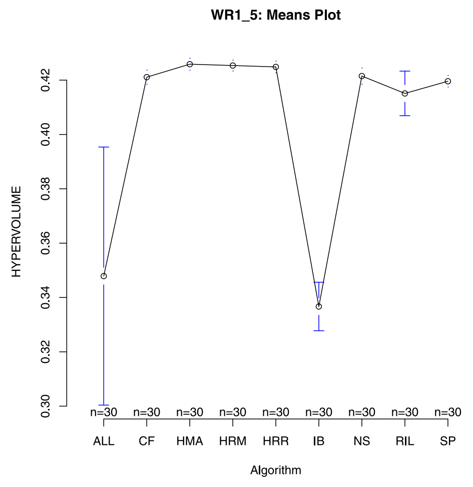


(d) Utilisation Rate: HH-RILA

Fig. 7. Utilisation Rate: WR problem instances.



(a) SP2



(b) WR1

Fig. 8. Means plots of the hypervolume (h) over 30 trials: SP2 and WR1. The number after "_" in the problem instance nomenclature is the number of objective functions, and the circle is the mean value. Caption: ALL = HH-ALL; CF = HH-CF; HMA = HRMA; HRM = HRISE_M; HRR = HRISE_R; IB = IBEA; NS = NSGA-II, RIL = HH-RILA; SP = SPEA2.

hypervolume (h) of NSGA-II over 30 trials is the second best if we compare only the three MOEAs in isolation. It is better than IBEA. Hence, the radical measure to get rid off an LLH k at the beginning was not a suitable measure in this case. As we have previously mentioned, in HRISE_R and HRISE_M we can eliminate an LLH k but such action may be temporary because there is a possibility it comes back again, depending on the performance of the remaining MOEAs. This explains a more uniform utilisation rate in our hyper-heuristics.

As for WFG, HH-RILA was even more radical excluding NSGA-II from WFG3 up WFG8 problem instances and selecting many more times IBEA. This was a good decision since the very good outcomes produced by IBEA when applied to solve the WFG problem instances as shown in Section 5.2. However, note again that the mean hypervolume of NSGA-II was better than SPEA2 for the WFG3 problem instance (Fig. 5(b)).

In Figs. 6 and 7, we show the utilisation rates for two real-world problems, respectively, SP and WR. As in the case of the benchmark problems, HH-RILA presented a very nonuniform LLH selection behaviour even if now only in WR5 an LLH (IBEA) was excluded definitely. As earlier, this can be suitable or not. If we consider SP2, HH-RILA selected proportionally many more times IBEA even if the mean hypervolume between the three MOEAs are very close (see Fig. 8(a)). On the other hand, in WR1, HH-RILA selected a few times IBEA and it seems suitable since IBEA presented the smallest of all average hypervolume values for such a problem instance (see Fig. 8(b)).

It is not totally clear whether eliminating an LLH k and avoiding it to be executed at all, as HH-RILA does, is overall a worse or better solution compared to a more permissive approach where other chances are given to an LLH k that was previously ruled out, as our hyper-heuristics accomplish. This is a point that remains an open question within the context of hyper-heuristics.

We now show some Solution Fronts[4] where our hyper-heuristics had better and worse performances compared to HH-RILA, considering the benchmark and real-world problem instances. We also show the True Pareto Front or True Known Pareto Front as references. Regarding the benchmark problem instances, DTLZ3 was one where HRISE_R and HRISE_M got better results (Fig. 9). Note that HH-RILA struggled here where many points in the objective space present very low objective values given the sensation that we have only few points (Fig. 9(d)). At first, we observed this behaviour in our hyper-heuristics too, but the nonuniform iterations strategy was crucial to overcome this issue. In Fig. 10, we see a better performance of HH-RILA particularly when compared to HRISE_R for the WFG1 problem instance.

Regarding the real-world problem instances, Figs. 11 and 12 show one of the best and worst performances, respectively, compared to HH-RILA. It is evident that in SP1 (three objectives) our approaches outperformed HH-RILA and the opposite happened in VC3 (two objectives).

Finally, we would like to emphasise the diversity of different problem instances, unconstrained and constrained, that we addressed in the evaluation. As we know, there is a huge interest from the community in assessing complicated real-world problems [52]. We believe we approached this adequately by addressing four different real-world problems with problem instances with up to five objective functions. To highlight this point, we show the True Known Pareto Front of the real-world problem instance VC1 in Fig. 13. Looking at this front, we see that there is no clear pattern or characteristic that defines it and, moreover, it presents discontinuities posing difficulties for the algorithms to solve such problem instance.

[4] The final population obtained by the algorithms is also known as the Solution Set. The Solution Front lies in the objective space, and it is composed of the values of the objective functions of the elements (solutions) in the Solution Set.

Table 7
Correlation analysis: selected LLHs × accepted populations.

Problem	Algorithm	τ	Conclusion
DTLZ			
	HRISE_R	0.704	Strong+
	HRISE_M	0.66	Strong+
	HRMA	0.133	Weak+
WFG			
	HRISE_R	0.799	Strong+
	HRISE_M	0.754	Strong+
	HRMA	0.383	Medium+
VC			
	HRISE_R	0.779	Strong+
	HRISE_M	0.646	Strong+
	HRMA	−0.156	Weak−
WR			
	HRISE_R	0.475	Medium+
	HRISE_M	0.654	Strong+
	HRMA	0.262	Weak+
CS			
	HRISE_R	0.706	Strong+
	HRISE_M	0.366	Medium+
	HRMA	−0.114	No
SP			
	HRISE_R	0.514	Strong+
	HRISE_M	0.725	Strong+
	HRMA	0.171	Weak+

5.4. Evaluation of the Heuristic selection mechanisms: Accepted populations

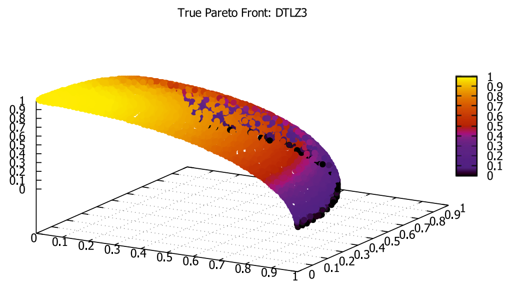
Since HRMA, our random heuristic selection approach, presented results compatible with HRISE_R and HRISE_M, we wondered whether the LLH selection method we designed, roulette wheel supported by Reinforcement Learning plus balanced exploitation/exploration, was really effective. In other words, what is the correlation between the selected LLHs due to HRISE_R and HRISE_M and the number of accepted populations (by our two-level mechanism)? And is this correlation really better than a simple random heuristic selection technique? Recall again that, in HRMA, Reinforcement Learning and the balanced exploitation/exploration mechanism are not used.

In order to answer these questions, we relied on the Kendall rank correlation coefficient, τ, and Table 7 presents the results considering two benchmark problems, DTLZ and WFG, and all real-word problems. As shown in the table, HRISE_R and HRISE_M get basically strong positive correlations with a few medium positive relationships. On the other hand, HRMA has mostly weak correlations with even a negative one in the VC problem.

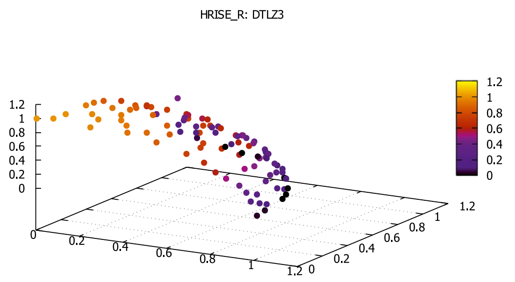
We then conclude that the heuristic selection mechanism designed within HRISE_R and HRISE_M matters and it is more suitable than a simple random approach.

6. Conclusions

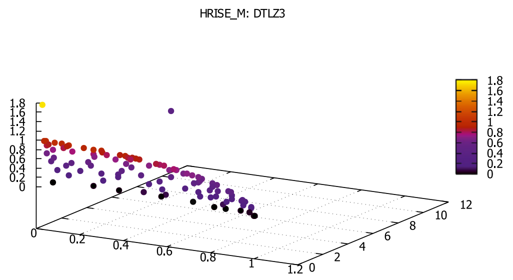
While the studies on selection hyper-heuristics have rapidly been increasing based on single point-based search for single objective optimisation, the studies on multi-objective hyper-heuristics are still missing in the scientific literature, possibly because it is not straightforward to design a hyper-heuristic combining an appropriate heuristic selection method with a robust



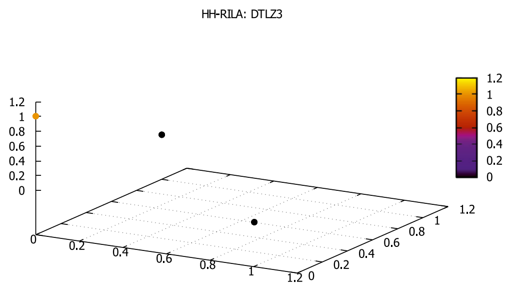
(a) True Pareto Front



(b) Solution Front: HRISE_R

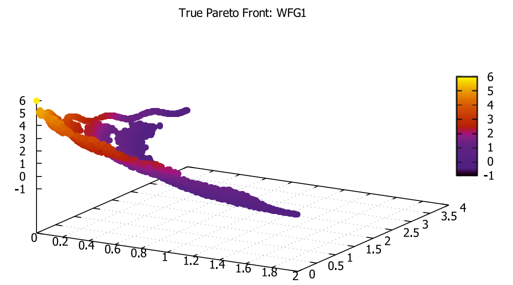


(c) Solution Front: HRISE_M

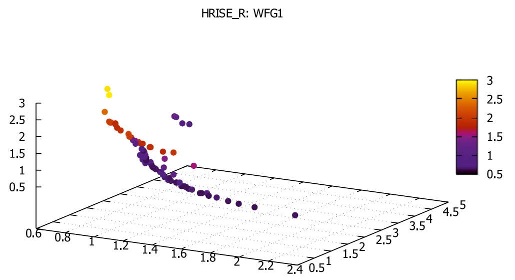


(d) Solution Front: HH-RILA

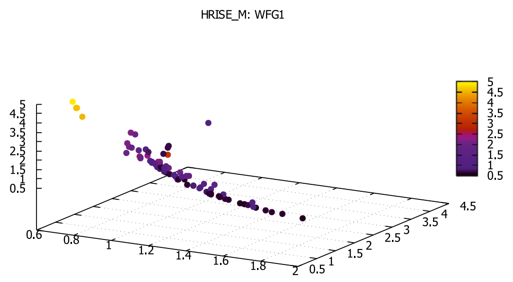
Fig. 9. One of the best performances of our approaches: DTLZ3 (benchmark).



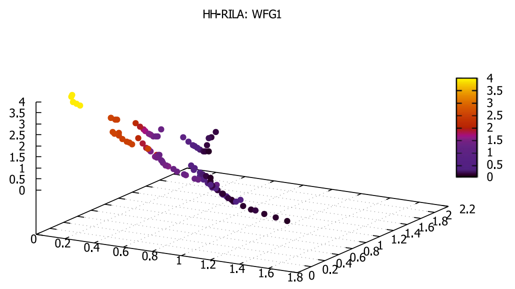
(a) True Pareto Front



(b) Solution Front: HRISE_R



(c) Solution Front: HRISE_M



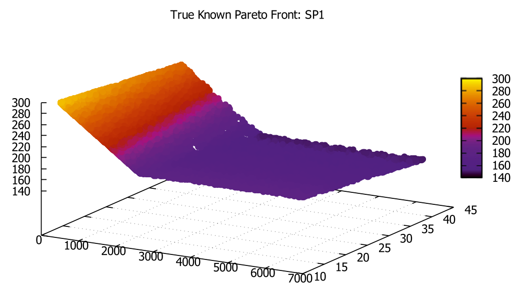
(d) Solution Front: HH-RILA

Fig. 10. One of the worst performances of our approaches: WFG1 (benchmark).

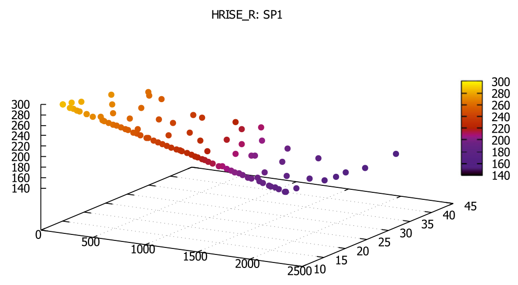
acceptance method, and the use of population-based approaches as Low-Level (meta)Heuristics introduces additional complexities.

In this study, we presented the HRISE selection hyper-heuristics for multi-objective optimisation managing a set of low-level MOEAs and embedding the following novel algorithmic components:

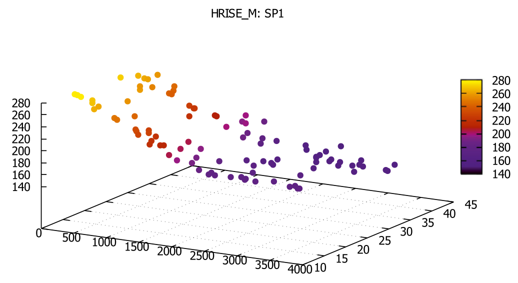
1. Reinforcement Learning-based metaheuristic selection method complemented by a balanced exploitation/exploration approach;
2. Dynamic control mechanism (nonuniform iterations strategy) deciding how long (for how many iterations) a selected LLH will run;



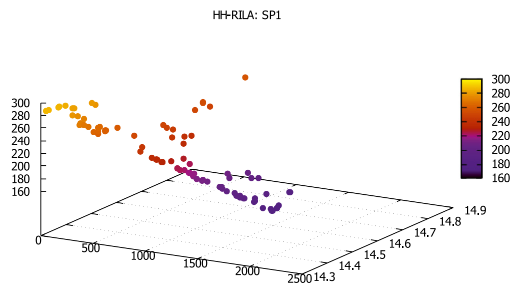
(a) True Known Pareto Front



(b) Solution Front: HRISE_R

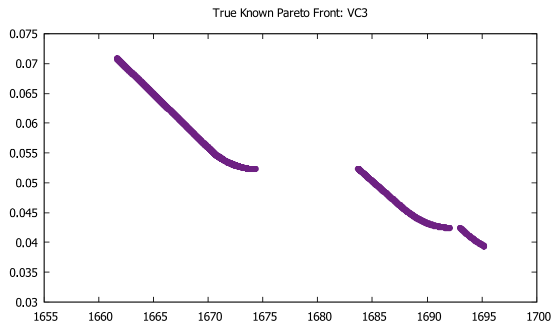


(c) Solution Front: HRISE_M

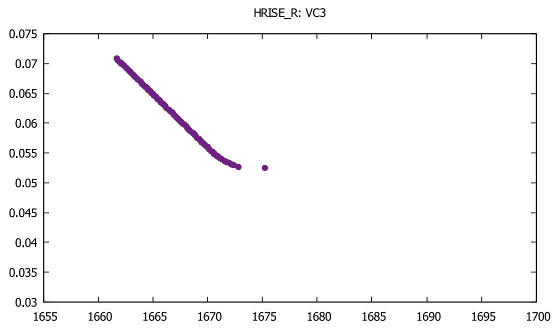


(d) Solution Front: HH-RILA

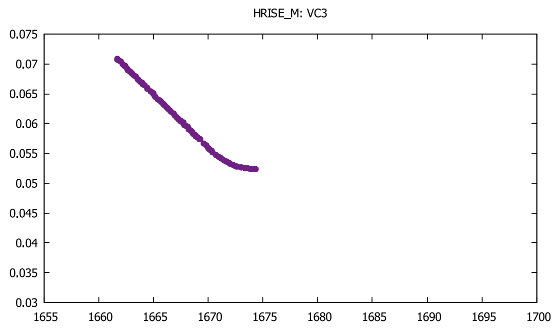
Fig. 11. One of the best performances of our approaches: SP1 (real).



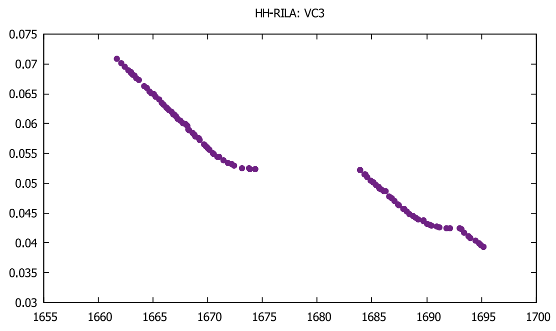
(a) True Known Pareto Front



(b) Solution Front: HRISE_R



(c) Solution Front: HRISE_M



(d) Solution Front: HH-RILA

Fig. 12. One of the worst performances of our approaches: VC3 (real).

3. Two-level (hierarchical) acceptance approach, Only Improving and group decision-making, where in the latter we have the responsibility and majority rules (denoted as hyper-heuristics HRISE_R and HRISE_M, respectively); and
4. QLA and MQLA acceptance methods proposed to be used within the group along with GDA.

A third proposed hyper-heuristic is HRMA where LLH selection is random but we still have the nonuniform strategy, and we make a random selection of all available move acceptance methods. Previous works in selection hyper-heuristics for single objective optimisation show the importance of learning in heuristic selection [67] as well as how crucial the choice of move

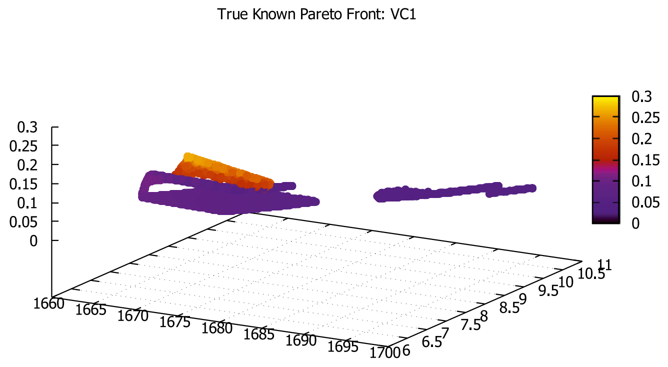


Fig. 13. True Known Pareto Front: VC1 (real).

acceptance is in combination with heuristic selection [17,68]. This study confirms the same observations for multi-objective optimisation and additionally emphasises that an improved overall performance is possible using multiple acceptance methods under a group decision-making scheme.

The empirical results, across 39 problem instances from four classes of benchmark functions and four real-world problems, where we evaluated both unconstrained and constrained problems, illustrate that our hyper-heuristics, with the proposed algorithmic components, performed the best when compared to HH-CF, HH-ALL, and each low-level MOEA run in isolation in terms of hypervolume. Still regarding the hypervolume, both HRISE hyper-heuristics and HRMA delivered a slightly better performance than the recently proposed state-of-the-art selection hyper-heuristic HH-RILA. Although this performance variation was not statistically significant, we may conclude that, considering the problem instances we selected, our hyper-heuristics were better than HH-RILA taking into account the pairwise comparison between HH-RILA and some other approaches where HH-RILA was not better statistically speaking but our hyper-heuristics were. Considering the ϵ indicator, HRISE_R was the best followed by HH-RILA. Overall, considering both quality indicators, HRISE_R presented the best performance of all algorithms.

A natural future research direction would be evaluating and analysing the whole approach under various configurations, for example, including different set of low-level metaheuristics and/or adding more move acceptance methods, such as the Iteration Limited Threshold Accepting (ILTA) [68] and Adaptive Simulated Annealing (ASA) [69] for multi-objective optimisation. Additionally, the proposed framework can be evaluated for many-objective optimisation based on other relevant metrics, such as fast computation hypervolume [70] along with the RPO indicator proposed in this article, and relying on other relevant approaches as low-level metaheuristics, such as NSGA-III [71] and the Many Objective Metaheuristic Based on the R2 indicator-II (MOMBI-II) [72]. Another natural extension to the current work would be applying this general-purpose approach to several additional unseen real-world multi/many-objective problems to evaluate its performance, aiming to conclude definitively in terms of generalisation.

CRediT authorship contribution statement

Valdivino Alexandre de Santiago Júnior: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Project administration, Funding acquisition. **Ender Özcan:** Writing - review & editing, Supervision. **Vinicius Renan de Carvalho:** Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by *Fundação de Amparo à Pesquisa do Estado de São Paulo* (FAPESP), Brazil, process number: 2018/08372-8.

References

[1] E.K. Burke, M.R. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, A classification of hyper-heuristic approaches: Revisited, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Springer International Publishing, Cham, 2019, pp. 453–477, http://dx.doi.org/10.1007/978-3-319-91086-4_14.
[2] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: A survey of the state of the art, J. Oper. Res. Soc. 64 (12) (2013) 1695–1724, http://dx.doi.org/10.1057/jors.2013.71.
[3] E. Özcan, M. Misir, G. Ochoa, E.K. Burke, A reinforcement learning-great-deluge hyper-heuristic for examination timetabling, Int. J. Appl. Metaheuristic Comput. 1 (1) (2010) 39–59, http://dx.doi.org/10.4018/jamc.2010102603.
[4] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: E. Burke, W. Erben (Eds.), Practice and Theory of Automated Timetabling III, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 176–190.
[5] P. Cowling, G. Kendall, E. Soubeiga, Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation, in: S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, G.R. Raidl (Eds.), Applications of Evolutionary Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 1–10.
[6] M. Ayob, G. Kendall, A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine, in: Placement Machine, INTECH'03 Thailand, 2003, pp. 132–141.
[7] D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems, Comput. Oper. Res. 34 (8) (2007) 2403–2435, http://dx.doi.org/10.1016/j.cor.2005.09.012, URL http://www.sciencedirect.com/science/article/pii/S0305054805003023.
[8] E. Ozcan, Y. Bykov, M. Birben, E.K. Burke, Examination timetabling using late acceptance hyper-heuristics, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 997–1004, http://dx.doi.org/10.1109/CEC.2009.4983054.
[9] J.H. Drake, A. Kheiri, E. Özcan, E.K. Burke, Recent advances in selection hyper-heuristics, European J. Oper. Res. 285 (2) (2020) 405–428, http://dx.doi.org/10.1016/j.ejor.2019.07.073, URL http://www.sciencedirect.com/science/article/pii/S0377221719306526.
[10] G. Guizzo, S.R. Vergilio, A.T.R. Pozo, G.M. Fritsche, A multi-objective and evolutionary hyper-heuristic applied to the Integration and Test Order Problem, Appl. Soft Comput. 56 (2017) 331–344, http://dx.doi.org/10.1016/j.asoc.2017.03.012, URL http://www.sciencedirect.com/science/article/pii/S1568494617301357.
[11] T.N. Ferreira, J.A.P. Lima, A. Strickler, J.N. Kuk, S.R. Vergilio, A. Pozo, Hyper-heuristic based product selection for software product line testing, IEEE Comput. Intell. Mag. 12 (2) (2017) 34–45, http://dx.doi.org/10.1109/MCI.2017.2670461.
[12] J.A.P. Lima, S.R. Vergilio, A multi-objective optimization approach for selection of second order mutant generation strategies, in: Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing, SAST, ACM, New York, NY, USA, 2017, pp. 6:1–6:10, http://dx.doi.org/10.1145/3128473.3128479, URL http://doi.acm.org/10.1145/3128473.3128479.
[13] M. Maashi, E. Özcan, G. Kendall, A multi-objective hyper-heuristic based on choice function, Expert Syst. Appl. 41 (9) (2014) 4475–4493, http://dx.doi.org/10.1016/j.eswa.2013.12.050, URL http://www.sciencedirect.com/science/article/pii/S095741741400013X.
[14] W. Li, E. Özcan, R. John, Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation, Renew. Energy 105 (2017) 473–482, http://dx.doi.org/10.1016/j.renene.2016.12.022, URL http://www.sciencedirect.com/science/article/pii/S0960148116310709.
[15] V.R. Carvalho, K. Larson, A.A.F. Brandão, J.S. Sichman, Applying social choice theory to solve engineering multi-objective optimization problems, J. Control Autom. Electr. Syst. 31 (2020) 119–128, http://dx.doi.org/10.1007/s40313-019-00526-2.
[16] W. Li, E. Özcan, R. John, A learning automata-based multiobjective hyper-heuristic, IEEE Trans. Evol. Comput. 23 (1) (2019) 59–73, http://dx.doi.org/10.1109/TEVC.2017.2785346.

[17] A. Kheiri, M. Mısır, E. Özcan, Ensemble move acceptance in selection hyper-heuristics, in: T. Czachórski, E. Gelenbe, K. Grochla, R. Lent (Eds.), Computer and Information Sciences, Springer International Publishing, Cham, 2016, pp. 21–29.

[18] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197, http://dx.doi.org/10.1109/4235.996017.

[19] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: X. Yao, E.K. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J.E. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature, PPSN VIII, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 832–842.

[20] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Performance of the Strength Pareto Evolutionary Algorithm, Tech. Rep., Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 2001.

[21] E. K.B.urke, Y. Bykov, A late acceptance strategy in hill-climbing for examination timetabling problems, in: Proceedings of the International Conference on the Practice and Theory of Automated Timetabling, PATAT, 2008, Extended Abstract.

[22] W.G. Jackson, E. Özcan, J.H. Drake, Late acceptance-based selection hyper-heuristics for cross-domain heuristic search, in: 2013 13th UK Workshop on Computational Intelligence, UKCI, 2013, pp. 228–235, http://dx.doi.org/10.1109/UKCI.2013.6651310.

[23] K. McClymont, E. Keedwell, D. Savić, M. Randall-Smith, A general multi-objective hyper-heuristic for water distribution network design with discolouration risk, J. Hydroinform. 15 (3) (2012) 700–716, http://dx.doi.org/10.2166/hydro.2012.022.

[24] K. McClymont, E.C. Keedwell, 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, New York, NY, USA, 2011, pp. 2003–2010, http://dx.doi.org/10.1145/2001576.2001845, URL http://doi.acm.org/10.1145/2001576.2001845.

[25] M. Harman, Y. Jia, Y. Zhang, Achievements, open problems and challenges for search based software testing, in: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST, 2015, pp. 1–12, http://dx.doi.org/10.1109/ICST.2015.7102580.

[26] A. Saeed, S.H.A. Hamid, M.B. Mustafa, The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review, Appl. Soft Comput. 49 (2016) 1094–1117, http://dx.doi.org/10.1016/j.asoc.2016.08.030, URL http://www.sciencedirect.com/science/article/pii/S1568494616304240.

[27] J.M. Balera, V.A. Santiago Júnior, A systematic mapping addressing Hyper-Heuristics within Search-based Software Testing, Inf. Softw. Technol. 114 (2019) 176–189, http://dx.doi.org/10.1016/j.infsof.2019.06.012, URL http://www.sciencedirect.com/science/article/pii/S0950584919301430.

[28] M. Khari, P. Kumar, An extensive evaluation of search-based software testing: A review, Soft Comput. 23 (6) (2019) 1933–1946, http://dx.doi.org/10.1007/s00500-017-2906-y.

[29] J.-Y. Audibert, R. Munos, C. Szepesvári, Exploration–exploitation tradeoff using variance estimates in multi-armed bandits, Theoret. Comput. Sci. 410 (19) (2009) 1876–1902, http://dx.doi.org/10.1016/j.tcs.2009.01.016, Algorithmic Learning Theory, URL http://www.sciencedirect.com/science/article/pii/S030439750900067X.

[30] S. Jain, S. Gujar, S. Bhat, O. Zoeter, Y. Narahari, A quality assuring, cost optimal multi-armed bandit mechanism for expertsourcing, Artificial Intelligence 254 (2018) 44–63, http://dx.doi.org/10.1016/j.artint.2017.10.001, URL http://www.sciencedirect.com/science/article/pii/S000437021730125X.

[31] C.M. Fonseca, P.J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation, IEEE Trans. Syst. Man Cybern. A 28 (1) (1998) 26–37, http://dx.doi.org/10.1109/3468.650319.

[32] G. Dueck, New optimization heuristics: The great deluge algorithm and the record-to-record travel, J. Comput. Phys. 104 (1) (1993) 86–92, http://dx.doi.org/10.1006/jcph.1993.1010, URL http://www.sciencedirect.com/science/article/pii/S0021999183710107.

[33] S. Kukkonen, J. Lampinen, GDE3: The third evolution step of generalized differential evolution, in: 2005 IEEE Congress on Evolutionary Computation, Vol. 1, 2005, pp. 443–450, http://dx.doi.org/10.1109/CEC.2005.1554717.

[34] E. Shojaedini, M. Majd, R. Safabakhsh, Novel adaptive genetic algorithm sample consensus, Appl. Soft Comput. 77 (2019) 635–642, http://dx.doi.org/10.1016/j.asoc.2019.01.052, URL http://www.sciencedirect.com/science/article/pii/S1568494619300651.

[35] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271, http://dx.doi.org/10.1109/4235.797969.

[36] K. Bringmann, T. Friedrich, The maximum hypervolume set yields near-optimal approximation, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10, ACM, New York, NY, USA, 2010, pp. 511–518, http://dx.doi.org/10.1145/1830483.1830576, URL http://doi.acm.org/10.1145/1830483.1830576.

[37] S. Jiang, Y. Ong, J. Zhang, L. Feng, Consistencies and contradictions of performance metrics in multiobjective optimization, IEEE Trans. Cybern. 44 (12) (2014) 2391–2404, http://dx.doi.org/10.1109/TCYB.2014.2307319.

[38] D.A. Van Veldhuizen, G.B. Lamont, On measuring multiobjective evolutionary algorithm performance, in: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), Vol. 1, 2000, pp. 204–211, http://dx.doi.org/10.1109/CEC.2000.870296.

[39] K. Tan, T. Lee, E. Khor, Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons, Artif. Intell. Rev. 17 (4) (2002) 251–290, http://dx.doi.org/10.1023/A:1015516501242.

[40] D.A. Van Veldhuizen, G.B. Lamont, Multiobjective evolutionary algorithm test suites, in: Proceedings of the 1999 ACM Symposium on Applied Computing, SAC '99, ACM, New York, NY, USA, 1999, pp. 351–357, http://dx.doi.org/10.1145/298151.298382, URL http://doi.acm.org/10.1145/298151.298382.

[41] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132, http://dx.doi.org/10.1109/TEVC.2003.810758.

[42] J.R. Schott, Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization (Ph.D. thesis), Massachusetts Institute of Technology (MIT), Dept. of Aeronautics and Astronautics, 1995.

[43] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302, http://dx.doi.org/10.1109/TEVC.2008.925798.

[44] H. Ishibuchi, H. Masuda, Y. Nojima, A study on performance evaluation ability of a modified inverted generational distance indicator, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15, ACM, New York, NY, USA, 2015, pp. 695–702, http://dx.doi.org/10.1145/2739480.2754792, URL http://doi.acm.org/10.1145/2739480.2754792.

[45] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 892–899, http://dx.doi.org/10.1109/CEC.2006.1688406.

[46] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, second ed., in: Adaptive Computation and Machine Learning series, A Bradford Book, 2018.

[47] K. Kimura, H. Sawada, J. Katayama, Outcome evaluations in group decision-making using authority rule: An electrophysiological study, Neuropsychologia 119 (2018) 271–279, http://dx.doi.org/10.1016/j.neuropsychologia.2018.08.031, URL http://www.sciencedirect.com/science/article/pii/S0028393218305451.

[48] B. Kiraz, A.Ş. Etaner-Uyar, E. Özcan, Selection hyper-heuristics in dynamic environments, J. Oper. Res. Soc. 64 (12) (2013) 1753–1769, http://dx.doi.org/10.1057/jors.2013.24.

[49] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multiobjective optimization, in: A. Abraham, L. Jain, R. Goldberg (Eds.), Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, Springer London, London, 2005, pp. 105–145, http://dx.doi.org/10.1007/1-84628-137-7_6.

[50] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477–506, http://dx.doi.org/10.1109/TEVC.2005.861417.

[51] H. Jain, K. Deb, An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach, IEEE Trans. Evol. Comput. 18 (4) (2014) 602–622, http://dx.doi.org/10.1109/TEVC.2013.2281534.

[52] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization test instances for the CEC 2009 special session and competition, 2009, Available from: https://bit.ly/35M9818, (Access in: 7 Jan 2020).

[53] X. Liao, Q. Li, X. Yang, W. Zhang, W. Li, Multiobjective optimization for crash safety design of vehicles using stepwise regression model, Struct. Multidiscip. Optim. 35 (6) (2008) 561–569, http://dx.doi.org/10.1007/s00158-007-0163-x.

[54] D.P. Loucks, E. van Beek, Water resources planning and management: An overview, in: Water Resource Systems Planning and Management: An Introduction To Methods, Models, and Applications, Springer International Publishing, Cham, 2017, pp. 1–49, http://dx.doi.org/10.1007/978-3-319-44234-1_1.

[55] T. Ray, K. Tai, K.C. Seow, Multiobjective design optimization by an evolutionary algorithm, Eng. Optim. 33 (4) (2001) 399–424, http://dx.doi.org/10.1080/03052150108940926.

[56] A. Sinha, D.K. Saxena, K. Deb, A. Tiwari, Using objective reduction and interactive procedure to handle many-objective optimization problems, Appl. Soft Comput. 13 (1) (2013) 415–427, http://dx.doi.org/10.1016/j.asoc.2012.08.030.

[57] V.A. Santiago Júnior, S. Tahar, Time performance formal evaluation of complex systems, in: M. Cornélio, B. Roscoe (Eds.), Formal Methods: Foundations and Applications, Springer International Publishing, Cham, 2016, pp. 162–177.

[58] J. Braga, F. D'Amico, M.A.C. Avila, A.V. Penacchioni, J.R. Sacahui, V.A. Santiago Júnior, F. Mattiello-Francisco, C. Strauss, M.A.A. Fialho, The protoMIRAX hard X-ray imaging balloon experiment, Astron. Astrophys. 580 (2015) A108.

[59] J.J. Durillo, A.J. Nebro, jMetal: A java framework for multi-objective optimization, Adv. Eng. Softw. 42 (10) (2011) 760–771, http://dx.doi.org/10.1016/j.advengsoft.2011.05.014, URL http://www.sciencedirect.com/science/article/pii/S0965997811001219.

[60] V.R. Carvalho, Jmetalhyperheuristichelper web site, 2019, Available from: https://bit.ly/2G11gyx, (Access in: 07 Aug 2019).

[61] K. Deb, R.B. Agrawal, Simulated binary crossover for continuous search space, Complex Syst. 9 (2) (1995) 115–148.

[62] M. Wagner, K. Bringmann, T. Friedrich, F. Neumann, Efficient optimization of many objectives by approximation-guided evolution, European J. Oper. Res. 243 (2) (2015) 465–479, http://dx.doi.org/10.1016/j.ejor.2014.11.032, URL http://www.sciencedirect.com/science/article/pii/S0377221714009552.

[63] V.A. Santiago Júnior, E. Özcan, V.R. Carvalho, Experimental evaluation data: Applied Soft Computing Journal, 2020, Available from: https://bit.ly/2kuPAgc, (Access in: 12 Jan 2020).

[64] T. Hothorn, K. Hornik, M. van de Wiel, A. Zeileis, Implementing a class of permutation tests: The coin package, J. Stat. Softw. 28 (8) (2008) 1–23, http://dx.doi.org/10.18637/jss.v028.i08, https://www.jstatsoft.org/v028/i08.

[65] J.M. Balera, V.A. Santiago Júnior, An algorithm for combinatorial interaction testing: definitions and rigorous evaluations, J. Softw. Eng. Res. Dev. 5 (1) (2017) 10, http://dx.doi.org/10.1186/s40411-017-0043-z.

[66] L.B.R. Santos, V.A. Santiago Júnior, L.V. Povoa, A.V. Freitas, C.C. Mario, Software inspections: comparing a formal method based with a classical reading methodology, Int. J. Comput. Appl. Technol. 59 (4) (2019) 296–317, http://dx.doi.org/10.1504/IJCAT.2019.099198.

[67] E. Özcan, B. Bilgin, E.E. Korkmaz, A comprehensive analysis of hyper-heuristics, Intell. Data Anal. 12 (1) (2008) 3–23, URL http://dl.acm.org/citation.cfm?id=1368027.1368029.

[68] M. Misir, T. Wauters, K. Verbeeck, G.V. Berghe, A new learning hyper-heuristic for the traveling tournament problem, in: Proceedings of the 8th Metaheuristic International Conference, MIC'09, 2009, pp. id1–id10.

[69] X. Geng, Z. Chen, W. Yang, D. Shi, K. Zhao, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, Appl. Soft Comput. 11 (4) (2011) 3680–3689, http://dx.doi.org/10.1016/j.asoc.2011.01.039, URL http://www.sciencedirect.com/science/article/pii/S1568494611000573.

[70] S. Rostami, F. Neri, A fast hypervolume driven selection mechanism for many-objective optimisation problems, Swarm Evol. Comput. 34 (2017) 50–67, http://dx.doi.org/10.1016/j.swevo.2016.12.002, URL http://www.sciencedirect.com/science/article/pii/S2210650216301328.

[71] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints, IEEE Trans. Evol. Comput. 18 (4) (2014) 577–601, http://dx.doi.org/10.1109/TEVC.2013.2281535.

[72] R. H. Gómez, C.A. C. Coello, Improved metaheuristic based on the R2 indicator for many-objective optimization, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15, ACM, New York, NY, USA, 2015, pp. 679–686, http://dx.doi.org/10.1145/2739480.2754776, URL http://doi.acm.org/10.1145/2739480.2754776.