Manuscript Details

Manuscript number	SWEVO_2017_337_R2
Title	Analysis of Selection Hyper-heuristics for Population-based Meta-heuristics in Real-valued Dynamic Optimization
Short title	Analysis of Selection Hyper-heuristics for Population-based Meta-heuristics in Real-valued Dynamic Optimization
Article type	Full Length Article

Abstract

Dynamic optimization problems provide a challenge in that optima have to be tracked as the environment changes. The complexity of a dynamic optimization problem is determined by the severity and frequency of changes, as well as the behavior of the values and trajectory of optima. While many efficient algorithms have been developed to solve these types of problems, the choice of the best algorithm is highly dependent on the type of change present in the environment. This paper analyses the ability of popular selection operators used in a hyper-heuristic framework to continuously select the most appropriate optimization method over time. Empirical studies examine the behavioral differences between various hyper-heuristic selection operators to better understand their mode of operation. The results show that these hyper-heuristic approaches can yield higher performance more consistently across difference types of environments.

Keywords	hyper-heuristics; dynamic optimization; evolutionary computation; swarm intelligence.
Corresponding Author	Stefan van der Stockt
Corresponding Author's Institution	University of Pretoria
Order of Authors	Stefan van der Stockt, Andries P. Engelbrecht
Suggested reviewers	Katherine Malan

Submission Files Included in this PDF

File Name [File Type]

SWEVO_revision_notes_R1.pdf [Response to Reviewers]

SVDS_main.pdf [Manuscript File]

To view all the submission files, including those not included in the PDF, click on the manuscript title on your EVISE Homepage, then click 'Download zip file'.

SWEVO_2017_337_R1 Revision Request:

Analysis of Selection Hyper-heuristics for Population-based Meta-heuristics in Real-valued Dynamic Optimization

Author comments:

Thank you for the second round of reviews, the comments again helped focus both the background and conclusion sections much more. Please find the requested feedback below, and an outline of the manuscript changes they evoked.

Reviewer 1:

Reviewer:

The authors have almost addressed all my concerns. However, regarding the concept of adaptation in EC, it is not clear. The authors claim that "HMHH never adapts any of these meta-heuristic methods (or any of their sub-components). At all times, all of the heuristics are working together on the problem. HMHH periodically selects which heuristics to allocate the most computational resources (candidate solutions) to."

To my understanding, giving more candidate solutions to a certain operator(s) is a kind of adaptation. This means that more individuals would choose that operator(s) although some may choose other operators (all operators still work together). This would probably lead to a better performance than the version of HMHH without using this strategy.

Author Comment:

This is good observation and remains an intriguing question that many readers might also wonder about: should changing the number of candidate solutions assigned to a heuristic be considered an algorithm configuration change versus being considered as "adding more data" for the algorithm to exploit?

On the one hand, HMHH "embeds" all component and parameter value choices of each heuristic into a "sealed unit of configuration." HMHH never changes/adapts any heuristic's encapsulated algorithm logic, parameter values, operator functionality, or any other design decisions. The aim is to have a pool of multiple different (yet specific) algorithm configurations with certain known behavior. Different heuristics, in essence, become distinct components of unique fixed logic that yield different outcomes when applied to the same candidate solution. I.e. a candidate solution acted upon for one iteration by *rand/1/bin* DE algorithm logic will generally have entirely different output than a candidate solution acted on for one iteration by Quantum PSO or RIGA algorithm logic. The behavior of an individual candidate solution in the search space is noticeably altered depending on which heuristic acts upon the candidate solution. Section 2.4 was modified to make this distinction clearer.

On the other hand, adaptation in HMHH occurs by changing the number of candidate solutions assigned to each heuristic. Generally, more candidate solutions allow EC and SI algorithms to better exploit the search domain in fewer iterations. The HMHH heuristic selection mechanism tries to give the most promising heuristics every opportunity to succeed (i.e. more candidate solutions). We argue that giving a heuristic more candidate solutions does not alter the algorithmic logic embedded in the heuristic – the logic remains unchanged. However, more candidate solutions lets the heuristic be more informed about the search space.

The paper explores 10 different HMHH heuristic selection mechanism schemes ranging from fixed allocations, random allocations, and various types of performance-based allocations (which may be either deterministic or probabilistic, and may make use of learning and memory of good assignments). Good HMHH selection mechanisms prevent a downward spiral where one heuristic "takes over" by having all entities assigned to it, and then holding on to those entities forever (heuristic space convergence). If this happens, the search gets "stuck" using only that one heuristic. In the previous revision, the paper was heavily modified to focus more on the *heuristic space diversity* behavior of the various HMHH selection mechanisms to measure

1) how well different heuristic selection schemes for HMHH manage to avoid premature convergence to a single heuristic, and
2) how rapidly HMHH can "escape" from such a converged state to find more suitable heuristics.

Tables 4, 8,9,10 show how certain heuristic selection mechanisms are better at this task than others.

Nevertheless, your comment about the effect of reallocating candidate solutions between different heuristics in a HMHH context raises the following questions to investigate further:

- What should each heuristic's minimum and maximum allowed population size be, or is it simply unbounded (i.e. all entities in HMHH)? What population size range yields the greatest effectiveness for each heuristic?
- How are the *stability*, *robustness*, *satisficability* and *convergence speed* (as discussed in reference [5]) of different heuristics impacted by the hyper-heuristic reallocating entities between intelligent heuristics?
- Are these measures correlated with published population size scalability studies of the heuristics?
- Can these metrics help craft guidelines to guide practitioners on how to select suitable heuristics for a hyper-heuristic pool?
- How do the answers to any of the aforementioned open questions differ across various types of DOPs?

The conclusion section was modified to discuss these points of further study.

Reviewer 2:

Reviewer:
The authors have addressed all my comments and suggestions. The revised paper has been significantly improved.

Author Comment:
Thank you for all your help!

Reviewer 3:

Reviewer:
The revision version is good, and I suggest accepting it.

Author Comment:
Thank you for all your help!

Reviewer 4:

The whole paper generally is well written and the revision work is satisfactory. I still have following two concerns.

Reviewer:
(1) There are also some other concepts related to hyper-heuristics, such as ensemble algorithm and algorithm portfolios. Please also discuss them in the paper and cite some references as below.
• Population-based algorithm portfolios for numerical optimization. Evolutionary Computation, IEEE Transactions on, 2010. 14
• Ensemble of differential evolution variants. Information Sciences, 2018, 423:172–186.
• Differential Evolution with Multi-Population Based Ensemble of Mutation Strategies. Information Sciences, 329 (2016): 329-345

Author Comment:
These references make sense and are good recent examples. I incorporated them into the introduction section and in section 2.4 as part of the background on related techniques.

Reviewer:
(2) The language and the presentation should be further improved.

Author Comment:
We found that some language errors slipped through in the previous revision, notably in section 2.4 that saw heavy editing. These errors were corrected, and the entire paper was thoroughly reviewed again as well.

Regarding the presentation of the paper, we used the Elsevier template and tried to make text section, tables and figures look as good as we could (knowing that the final paper typesetting would have the trademark Elsevier paper layout applied). We also reformatted the appendix to be after the references, and slightly optimized the flow of section 2.4 and the conclusion based on reviewer 1's comments.

We stand by to make any other requested paper presentation changes and can certainly move any elements into different section & appendices if needed.

Analysis of Selection Hyper-heuristics for Population-based Meta-heuristics in Real-valued Dynamic Optimization

Stefan A.G. van der Stockt[a,*], Andries P. Engelbrecht[a]

[a] *Computational Intelligence Research Group (CIRG), University of Pretoria, cnr Lynnwood Road and Roper Street, Hatfield, South Africa*
Postal address: University of Pretoria, Private bag X20, Hatfield, 0028, South Africa

Abstract

Dynamic optimization problems provide a challenge in that optima have to be tracked as the environment changes. The complexity of a dynamic optimization problem is determined by the severity and frequency of changes, as well as the behavior of the values and trajectory of optima. While many efficient algorithms have been developed to solve these types of problems, the choice of the best algorithm is highly dependent on the type of change present in the environment. This paper analyses the ability of popular selection operators used in a hyper-heuristic framework to continuously select the most appropriate optimization method over time. Empirical studies examine the behavioral differences between various hyper-heuristic selection operators to better understand their mode of operation. The results show that these hyper-heuristic approaches can yield higher performance more consistently across difference types of environments.

Keywords: hyper-heuristics, dynamic optimization, evolutionary computation, swarm intelligence

1. Introduction

A dynamic optimization problems (DOP) is a special class of problem where optima change as time goes by. DOPs differ in their frequency and severity of changes, patterns in optima trajectory, search space composition (i.e. single or multiple base functions), homogeneity of optima movement, change pervasiveness among optima, and the hardness of the search space [1][2]. Recent surveys show that a significant amount of research focuses on the meta-heuristic approaches of *evolutionary computation* (EC) and *swarm intelligence* (SI) [3][4][5][6]. The surveys echo that various algorithms perform better in certain types of DOPs than in others. This presents a challenge to practitioners since it takes time to understand the nature of a given problem, and to identify a suitable algorithm to solve the problem. The wrong algorithm (or meta-heuristic parameter) choice can yield detrimental performance. Ideally, practitioners need an immediate "off the peg" solution to a DOP while effort is spent to develop more tailored approaches.

In parallel, the field of *operations research* has produced complementary methods called *hyper-heuristics* that adapt the optimization process by continually choosing which low-level heuristic to apply to a problem over time. The term *hyper-heuristic* was first used by Cowling and Soubeiga [7] in 2000, but early work in the field dates back to the probabilistic scheduling rules of Fisher and Thompson [8] in 1961. Burke *et al.* [9] review hyper-heuristic literature for both combinatorial and continuous optimization and distinguish between *selection* and *generative* hyper-heuristics. *Selection* hyper-heuristics use predefined selection operators inside a hyper-heuristic framework to choose a suitable heuristic to apply to a problem at time t. *Generative* hyper-heuristics iteratively evolve customized selection operators (mostly via *genetic programming*) that are tailored to a domain. Generative hyper-heuristics are out of scope for this paper.

Burke *et al.* [9] position hyper-heuristics against other adaptive control mechanisms such as *adaptive operator selection* (AOS) [10], *parameter control* (PC) for evolutionary algorithms (EAs) [11][12][13], *adaptive memetic algorithms* (MAs) [14][15], and *algorithm portfolios* [16]. Recent research into *differential evolution* ensembles [17][18] are also related to hyper-heuristics. What distinguishes hyper-heuristics from most other control adaptation approaches is the clear separation between solving a *problem* and searching for a suitable *method* (or *heuristic*) to solve a problem. Hyper-heuristics treat both the problem and heuristics as 'black boxes' by not having detailed knowledge about the problem space, nor knowing exactly how heuristics solve a problem. A number of studies investigate the application of hyper-heuristics to DOPs [19][20][21][22][23][24][25]. Most focus on simple heuristics (eg. Gaussian mutation). Later studies by Van der Stockt and Engelbrecht [24][25] use DOP-specific population-based meta-heuristic methods managed by the *heterogeneous meta-hyper-heuristic* (HMHH) framework

*Corresponding author
Email addresses: stefan.vanderstockt@gmail.com (Stefan A.G. van der Stockt), engel@cs.up.ac.za (Andries P. Engelbrecht)

[26][27][28] on a range of different classes of DOPs.

This paper investigates how well various hyper-heuristic selection operators can continually balance computational resources across different population-based meta-heuristics in order to solve a DOP better than the individual meta-heuristics can. The aim is to better understand what heuristic allocation behavior leads to improved performance. The purpose is not to find the best algorithm to solve DOPs, nor to compare hyper-heuristics with state-of-the-art DOP algorithms, nor to exhaustively determine the best factors that characterize a good heuristic pool.

The following novel contributions are made: Firstly, this is the first study to analyze a broad range of hyper-heuristic selection operators across the full spectrum of real-valued DOP types. Unique environments are systematically created with parameter values that are compliant with scenario 2 of the *moving peaks benchmark* [29][30], and based on the holistic classification of Duhain and Engelbrecht [1] (which unites the well-known classifications of Eberhart and Shi [31] and Angeline [32] with spatial and temporal change severity classes). Secondly, a contradictory situation faced by DOP-focused meta-heuristics is identified: statically tuning meta-heuristic parameters is impossible in DOPs [33], yet dynamically adapting multiple meta-heuristic parameters in an ad hoc fashion using state-of-the-art self-adaptive parameter control methods produces poor results (the so-called *'patchwork problem'*) [13]. This paper shows how selection hyper-heuristics can address this problem, and extends the HMHH framework by establishing the criteria needed to identify appropriate meta-heuristics to enable HMHH to solve DOPs. Thirdly, metrics capturing heuristic allocation behavior, heuristic space diversity, and allocation stability are used to understand *why* and *how* hyper-heuristic selection operators differ in their behavior. Top-performing selection operators have the ability to periodically converge on the most appropriate heuristic for time t, yet rapidly increase heuristic space diversity when the prevailing heuristic is no longer suitable.

The paper is organized as follows. Section 2 gives an overview of related work into DOPs, meta-heuristics, and hyper-heuristics. Section 3 motivates using hyper-heuristics to improve performance while solving DOPs, and also presents the rationale behind the heuristic choices and the 10 selection operators investigated in this study. Section 4 presents the experimental procedure used, and section 5 discusses the results. Section 6 concludes the study.

2. Related Work

A brief overview of real-valued dynamic optimization, meta-heuristics, and hyper-heuristics is outlined below.

2.1. Dynamic Optimization Problems

In the context of this study, a dynamic optimization problem (DOP) refers to the special class of problems that are solved by an optimization algorithm 'as time goes by' [5]. A real-valued DOP with static constraints is defined as

$$\text{maximize} \quad f(\vec{x}, \vec{\omega}(t)), \vec{x} \in \mathbb{R}^{n_x}$$

where $\vec{\omega}(t) = \langle \omega_1(t), ..., \omega_{n_\omega}(t) \rangle$ are time-dependent control parameters [34][1]. Solving $f(\vec{x}, \vec{\omega}(t))$ means finding the global optimum of the search landscape at time t, namely $\vec{x}^*(t) = \max_{\vec{x}} f(\vec{x}, \vec{\omega}(t))$ (for maximization).

The *moving peaks benchmark* (MPB) by Branke [29] is still the most often-used benchmark for real-valued DOPs [3][4][5][6][30]. The function value of the *moving peaks benchmark*, f, is the maximum of the combined peak functions, i.e.

$$f(\vec{x}, t) = \max \{B(\vec{x}), \max_{p=1..n_p} \{P(\vec{x}, h_p(t), w_p(t), \vec{l}_p(t))\}\} \tag{1}$$

where f is the MPB function, B is the basis function landscape (zero in this study), n_p is the number of peaks, P defines the peak for each peak p with height h_p, width w_p, and location $\vec{l}_p$ at time t. In this study P is defined as

$$P(\vec{x}, h_p(t), w_p(t), \vec{l}_p(t)) = h_p(t) - w_p(t) \sqrt{\sum_{j=1}^{n_x} (\vec{l}_p(t) - x_j)^2} \tag{2}$$

where n_x is the dimension of $\vec{x}$. Peak heights and widths are modified as

- height: $h_p(t) = h_p(t-1) + heightSeverity \cdot \sigma(t)$
- width: $w_p(t) = w_p(t-1) + widthSeverity \cdot \sigma(t)$

where $\sigma(t) \sim N(0, 1)$. The MPB shift vector $\vec{s}_v(t)$ is

$$\vec{s}_v(t) = \frac{s}{|\vec{p}_r + \vec{s}_v(t-1)|}((1-\lambda)\vec{p}_r + \lambda \vec{s}_v(t-1)) \tag{3}$$

where $\vec{p}_r$ is a random vector normalized to length s (the spatial severity).

Many classification systems have been devised to categorize DOPs. Eberhart *et al.* [31][35] define three types of dynamic environments based on the direction of change of the optima: Type I environments, where optima positions change but not their values; Type II environments, where optima values change but not their positions; and Type III environments where both the optima values and positions change. Angeline [32] shows that optima trajectory changes can be linear, circular or random. Duhain and Engelbrecht [1] combine the classifications of Eberhart *et al.* and Angeline with spatial and temporal change severity classes (*quasi-static, abrupt, progressive, and chaotic*) into

[1]This definition of a DOP may seem simplistic, yet is general enough to allow this study to focus on DOPs that (as per Nguyen *et al.* [5] and Cruz *et al.* [4]) 1) have a single objective, 2) mixes predictable and unpredictable change patterns, 3) have visible changes, i.e. no detection strategies are needed, 4) are unconstrained, and 5) do not have explicit time-linkage between states that depend on algorithm interactions with the environment.

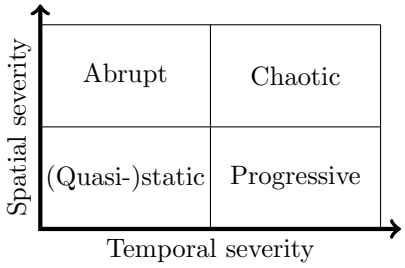

Figure 1: Spatial vs. temporal severity trade-off [1]

27 unique DOP types. Figure 1 illustrates the relationships between the change severity classes. Quasi-static environments are not investigated further in this paper since the focus is on evaluating how well hyper-heuristics adapt to landscape changes.

Duhain and Engelbrecht [1] provide detailed parameter selection guidelines to define 27 uniquely different types of DOPs. In this study, the 3-tuple notation (X, Y, Z) encodes the 27 environment types of Duhain and Engelbrecht, where $X \in \{\text{A, P, C}\}$ indicates if the environment change is *abrupt*, *progressive* or *chaotic*, $Y \in \{1, 2, 3\}$ indicates Eberhart *et al.*'s classification into Type I, II or III, and $Z \in \{\text{L, C, R}\}$ indicates Angeline's classification into *linear*, *circular* or *random* peak trajectories. The majority of the resulting environments are relatively straightforward to construct. For Type I and III circular environments, n-dimensional function rotation is used to rotate the peaks in a cyclic fashion (using variable cycle lengths to control the change severity). For Type II environments (where peaks remain stationary), linear, circular or random behavior is defined by the manner in which peak heights change. More detail on the exact considerations to take for each environment can be found in Appendix A.

Moser and Chiong [30] review a wide range of studies that use the MPB to test the performance of various EC, SI and hybrid approaches. Moser and Chiong note that most of the literature use parameter value ranges that are consistent with the original scenario 2 proposed by Branke due to the appropriateness of the resulting problem difficulty and solvability [29][30]. Accordingly, this study uses MPB parameters influenced by the guidelines of both Moser and Chiong and Duhain and Engelbrecht. Table 1 shows the resulting parameters shared by each environment, namely peak height h_p, and peak width w_p, function domain, dimensions, constraints, and number of peaks. Table 2 shows the MPB parameter values that yield 27 unique environments using the parameter considerations of Duhain and Engelbrecht, where h_s is the height severity, w_s is the width severity, s is the spatial severity, λ controls the randomness of a peak's trajectory, i is the number of iterations before a function landscape change, C is the cycle length of a full function landscape rotation (if applicable), and ϕ indicates the growing/shrinking behavior pattern of each peak (i.e. linear, circular or random).

Table 1: General MPB parameters

Parameter	**Value**	**Parameter**	**Value**
dimensions (d)	$d = 5$	peak height	$h_p \in [30, 70]$
function domain	$R(0, 100)^d$	peak width	$w_p \in [0.8, 7]$
constraints	*unconstrained*	peaks	10

Table 2: MPB parameters yielding each environment type

Env	h_s	w_s	s	λ	i	C	ϕ	Env	h_s	w_s	s	λ	i	C	ϕ
A1L	0	1	5	1	50	–	R	C1L	0	1	5	1	10	–	R
A1C	0	1	0	0	50	62	R	C1C	0	1	0	0	10	62	R
A1R	0	1	5	0	50	–	R	C1R	0	1	5	0	10	–	R
A2L	7	1	0	0	50	–	L	C2L	7	1	0	0	10	–	L
A2C	7	1	0	0	50	–	C	C2C	7	1	0	0	10	–	C
A2R	7	1	0	0	50	–	R	C2R	7	1	0	0	10	–	R
A3L	7	1	5	1	50	–	R	C3L	7	1	5	1	10	–	R
A3C	7	1	0	0	50	62	R	C3C	7	1	0	0	10	62	R
A3R	7	1	5	0	50	–	R	C3R	7	1	5	0	10	–	R
P1L	0	0.05	1	1	1	–	R	P2R	1	0.05	0	0	1	–	R
P1C	0	0.05	0	0	1	314	R	P3L	1	0.05	1	1	1	–	R
P1R	0	0.05	1	0	1	–	R	P3C	1	0.05	0	0	1	314	R
P2L	1	0.05	0	0	1	–	L	P3R	1	0.05	1	0	1	–	R
P2C	1	0.05	0	0	1	–	C	-	-	-	-	-	-	-	

h_s is zero for Type I environments (where peak heights remain the same), C is only defined in circular type I and type III environments, and s is zero for Type II environments (where peaks never move).

2.2. DOP performance evaluation

The reviews of Cruz *et al.* [4] and Nguyen *et al.* [5] present well-known DOP-specific algorithm performance and behavior measures. Popular performance measures include *best of generation*, *collective mean fitness*, *modified off-line error*, *modified off-line performance*, *best-error-before-change*, *optimization accuracy*, and *normalized scores*. Behavioral measures evaluate certain algorithm traits that are considered useful to solve DOPs and include measuring *diversity levels*, *stability*, *robustness*, *reactivity* and *convergence speed*.

Cruz *et al.* [4], Nguyen *et al.* [5], and Moser and Chiong [30] each highlight the inadequacy of current practices of summarizing the entire run of an optimization algorithm as a single aggregation such as the statistical mean. The authors emphasize the need to measure what happens *during* a run, and recommend using rigorous rank-based, non-parametric statistical tests for comparison of performance. Helbig and Engelbrecht [36][37] propose such a measure that considers each environment change period individually, which allows comparisons of algorithms' tracking capability. Their method compares the measures for each algorithm across each individual change period, and uses a Kruskal-Wallis test to confirm if statistical significant differences exist between the different algorithms [38]. If the results are deemed different, a pairwise Mann-Whitney-Wilcoxon rank sum test with Holm correction is

Algorithm 1 Wins / Losses Ranking Method [37]

```
for each heuristic / hyper-heuristic do
    for each period ψ_p in each environment p do
        Perform Kruskal-Wallis test on measure m
        if measures are statistically different then
            for each pair of algorithms do
                Perform Mann-Whitney U-test on m
                if measures are statistically different then
                    Assign wins and losses using median
                end if
            end for
        end if
    end for
    calculate γ_norm(p, m) using equation (4)
end for
```

used to assess differences between each pair of algorithms [39]. The median performance over the respective sample set is used to award a win to the superior algorithm and a loss to the inferior algorithm. For measure m, the wins, w, and losses, l, are normalized by dividing by the number of environment change periods ψ_p as

$$\gamma_{norm}(p,m) = \frac{\sum_{i=1}^{\psi_p}(w_{i,m} - l_{i,m})}{\psi_p} \quad (4)$$

where $\gamma_{norm}(p,m)$ is the average wins-minus-losses of an algorithm for a specific environment using measure m. The resulting ranks can be used to compare algorithm performance while considering measure readings across the entire optimization run, and not just a single averaged metric. The ranking procedure is shown in algorithm listing 1.

2.3. Meta-heuristics for DOP-specific optimization

Sörensen and Glover [40] define a *metaheuristic* as a high-level problem-independent set of guidelines and strategies to design heuristic optimization algorithms. Sörensen [41] reaffirms that a meta-heuristic is less of an explicit algorithm and more of a consistent set of complimentary ideas, concepts and operators. Recent surveys provide comprehensive overviews of meta-heuristics specialized to solve DOPs [3][4][5][6]. Jin and Branke [3], Cruz *et al.* [4], and Nguyen *et al.* [5] give comprehensive overviews on *evolutionary dynamic optimization* (EDO) and *swarm intelligence dynamic optimization* (SIDO). Mavrovouniotis *et al.* [6] provides in-depth focus on existing and emerging SIDO approaches. The surveys in [3][4][5][6] present key 'building blocks' found in the majority of DOP-specific meta-heuristic methods, namely *introducing / maintaining diversity*, *reacting to change*, *use of explicit/implicit memory*, *multiple populations*, *detection of change*, *predicting change*, and using *self-adaptive mechanisms*.

Modern meta-heuristics combine these building blocks to balance the *exploration* of an ever-changing search landscape against the *exploitation* of each successive search landscape. State-of-the-art EC and SI examples include: *DynDE* [42], which is a *differential evolution* (DE) variant that combines Brownian individuals with exclusion criteria to prevent populations converging to the same optima; *competitive differential evolution* (CDE) [2], where a multi-population DE strategy that uses competition for function evaluations is used to locate optima fast in dynamic environments; *self-adaptive differential evolution* (SaDE) by Qin and Suganthan [43] that automatically adapts its learning strategy and parameters during the search; *jDE* [44] that encodes the DE control parameters into the indivual to allow self-adaption; *multi-phase multi-individual extremal optimization* (MMEO) [45] where multiple solutions are repeatedly mutated using a power-law distribution that attempts to exclude bad solutions rather than to find good solutions; *adaptive multi-population framework* (AMP) [46] that adaptively adjusts the number of populations based on feedback about the number of peaks in the search landscape; and *population-based incremental learning* (PBIL) [47] which uses a combination of population-based EA and incremental learning mechanisms together with a random immigrants approach to learn and exploit a probability vector of promising solutions.

Successful meta-heuristics (such as the examples above) differ in their implementation of each DOP building block, which in turn yields different strengths and weaknesses. Sörensen [41] argues that different meta-heuristics such as SI or EC algorithms are congruent to different *styles* of cooking, such as French or Chinese cuisine, and not *recipes* for specific dishes per se. Asking "*are SI approaches better than EC approaches?*" is akin to asking "*Is Chinese cooking better than French cooking?*" The answer invariably is "*it depends on who is dining*" since each meta-heuristic brings a unique set of operators and design decisions to the table. Sörensen [41] concludes that a component-based view of meta-heuristics (where operators from multiple different meta-heuristic frameworks can be combined into more powerful methods) is key to producing deep insights into why meta-heuristics work.

Wolpert and Macready [48] published the 'No Free Lunch' (NFL) theorems for optimization in 1997 which, informally, state that all optimization algorithms have equal performance when evaluated over all possible problems (for any performance measure). The implication is clear: in domains where NFL theorems hold, no method can be considered 'generally better' than any other method. However, Auger and Teytaud [49] show that no NFL theorems hold generally in continuous domains. Alabert *et al.* [50] sharpen the results of Auger and Teytaud by proving that there are indeed no NFL theorems for functions in continuous domains, except for a few extreme edge cases which require additional technical conditions that are simply too restrictive to be found in practice. Additionally, Poli and Graff [51] show that the NFL theorems do not automatically apply to meta-search methods such as hyper-heuristics. Specifically, the authors show that, in practice, the NFL does not apply to hyper-heuristic methods when

the set of problems under consideration is '*not too large*'[2]. The results of both Alabert *et al.* and Poli and Graff imply the possibility that, in any practical problem, an intelligent selection hyper-heuristic could continually improve search performance over using heuristics in isolation.

2.4. Hyper-heuristics

A hyper-heuristic is defined by Cowling and Chakhlevitch as a high-level heuristic control mechanism to manage low-level heuristics that searches for good *methods* and not good solutions, and uses limited problem-level knowledge [52]. The roots of hyper-heuristics lie in job scheduling, time tabling, routing problems, and combinatorial optimization. Early studies employ hand-crafted hyper-heuristic selection operators that manage a pool of highly domain dependent heuristics (usually also manually created by humans). A modern definition is that selection hyper-heuristics are 'heuristics to choose heuristics' [9]. Recent work uses EC and SI meta-heuristics both as low-level heuristics and/or hyper-heuristics selection operators [9]. Selection hyper-heuristics offer an attractive 'off-the-peg' control mechanism that does not rely on a full understanding of the problem space, but instead uses continual performance feedback to search for the best *method* to use at time t.

The *heterogeneous meta-hyper-heuristic* (HMHH) by Grobler *et al.* [26][27][28] is a selection hyper-heuristic framework which manages a pool of population-based meta-heuristics that are executed concurrently. Each heuristic is a concrete meta-heuristic algorithm configuration comprising of specific algorithm logic, parameter values, operator functionality, and other design decisions. HMHH treats each heuristic as a "*sealed unit*" and never adapts any of the aforementioned components. The aim is to have a pool of multiple different (yet specific) algorithm configurations with certain known behavior. Different heuristics, in essence, become distinct components of unique fixed logic that yield different outcomes when applied to the same candidate solution. I.e. the same candidate solution acted upon for one iteration by *rand/1/bin* DE algorithm logic will generally have entirely different output than a candidate solution acted on for one iteration by Quantum PSO or RIGA algorithm logic. The behavior of an individual candidate solution in the search space is noticeably altered depending on which heuristic acts upon the candidate solution.

Adaptation in HMHH occurs by changing the number of candidate solutions assigned to each heuristic. Generally, more candidate solutions allow EC and SI algorithms to better exploit the search domain in fewer iterations. The HMHH heuristic selection mechanism tries to give the most promising heuristics every opportunity to succeed (i.e. more candidate solutions). Giving a heuristic more candidate solutions does not alter the heuristic's embedded algorithmic logic - the heuristic is however more informed about the search space. Good HMHH selection mechanisms prevent a downward spiral where one heuristic "takes over" by holding onto all assigned entities forever (heuristic space convergence). If this happens, the search gets "stuck" using only that one heuristic. Grobler and Engelbrecht show that *heuristic space diversity* (HSD) plays an important role in hyper-heuristic performance in static environments [28]. The HSD metric $\mathcal{H}(t)$ measures the spread of heuristics in the heuristic space with respect to their use by the population of entities as

$$\mathcal{H}(t) = \alpha \left(1 - \frac{\sum_{m=1}^{n_h} |T - n_m(t)|}{1.5 n_s}\right) \tag{5}$$

where n_s is the total number of entities, n_h is the number of managed heuristics, n_m is the number of entities assigned to heuristic m, α is a scaling factor (here $\alpha = 100$), and $T = n_s/n_h$.[3]

HMHH is shown in algorithm 2 (with some notation adapted to match section 3.2). Every k iterations HMHH employs a *selection operator*, ς, to assign all entities[4] in the parent population E to heuristics. The performance feedback of each heuristic h_m, namely Q_{δ_m}, is used by ς to allocate more entities to well-performing heuristics and fewer entities to under-performing heuristics. Different selection operators allows HMHH to exhibit radically different heuristic allocation behavior.

HMHH was originally devised for static environments. Van der Stockt and Engelbrecht [24][25] investigate the effectiveness of HMHH in DOPs by using a *simple random* selection operator to manage a pool of DOP-specific meta-heuristics. The studies show that *random selection-based* HMHH outperforms each stand-alone heuristic in certain classes of DOPs, but is not superior in every class of DOP. They conclude that further research should determine how well various intelligent selection operators in HMHH could raise performance even further.

Distributed evolutionary algorithms (dEAs) bear some resemblance to HMHH, particularly population-based dEAs based on the heterogeneous island model [53]. HMHH is different in that entities do not migrate between heuristic populations ('*islands*') in the dEA sense. Each heuristic in HMHH operates exclusively on its own disjoint sub-population of dedicated entities for k iterations.

[2]The authors show that for problems with n distinct fitness values $[f_1, f_2, ..., f_n]$ at points $[\vec{x}_1, \vec{x}_2, ..., \vec{x}_n]$ there are $n!$ possible permutations of all fitness assignments across all points. If the problem set under consideration contains fewer than $n!$ problems, then the set is not closed under permutation (required by NFL) and the NFL theorem does not apply.

[3]Note that if there are more than four heuristics and all n_s entities are assigned to just a single heuristic (i.e. $n_m = 0$ for all but one heuristic), then $\mathcal{H}(t) < 0$ (since equation (5) then simplifies to $\alpha(1 - 2(n_h - 1)/1.5n_h)$). To avoid this situation (and since some heuristics have a minimum entity requirement as explained in section 3.2), this study maintains $n_m > 4$ at all times to ensure that $\mathcal{H}(t) > 0$.

[4]Entities represent candidate solutions and any associated state such as particle velocities, global best positions, etc.

Algorithm 2 Heterogeneous Meta-Hyper-Heuristic [28]

$E \leftarrow$ Initialize parent population of n_s solution entities.
$h_j(t) \leftarrow$ the heuristic algorithm applied to entity e_j at iteration t.
for all entities $e_j \in E$ **do**
  $h_j(1) \leftarrow$ choose random initial heuristic algorithm for e_j.
end for
$t = 1, k = 5$.
while a stopping condition is not met **do**
  for all entities $e_j \in E$ **do**
    Apply $h_j(t)$ to entity j for k iterations.[1]
    $Q_{\delta m}(t) \leftarrow$ total improvement of all entities assigned to heuristic algorithm h_m for the last k iterations.[2]
  end for
  for all entities $e_j \in E$ **do**
    $h_j(t+k) \leftarrow$ Select next heuristic algorithm for entity e_j using selection operator $\varsigma(e_j, Q\delta_m(t))$.[3]
  end for
  $t = t + k$.
end while

Notes:

1. Entities assigned to h_m collectively form a distinct subpopulation $s_m \subset E$ that h_m operates on exclusively for k iterations.
2. In [28], $\varsigma(e_j, Q\delta_m(t))$ is rank-based tabu search [54].
3. Entities may require additional heuristic-specific state information (see *semantic decoration* discussed in section 3.2).

The collective performance of all entities assigned to each heuristic m over the previous k iterations is used to calculate the performance feedback Q_{δ_m} for that heuristic h_m. Every k iterations, all heuristic populations are 'evacuated' and new populations are chosen from a common parent entity population E.

Karafotias *et al.* [13] present a survey of adaptive parameter control methods, and distinguish between the *parameter tuning problem* and *parameter control problem* as (respectively) stationary and non-stationary ways to adapt meta-heuristic performance. The *parameter tuning problem* is deemed essential by Karafotias *et al.* to the successful deployment of any meta-heuristic. Eiben and Smit [55] present a taxonomic breakdown of over 30 different parameter tuning methods, and propose a conceptual framework of tuning methods and a tuning-aware experimental methodology. However, meta-heuristic parameter tuning alone will not suffice in DOPs. Leonard and Engelbrecht [33] show that it is impossible to statically optimize PSO parameters for any given DOP. Each environment change results in a different optimal PSO parameter configuration. Intuitively, this finding extends to EC approaches, and implies that a differently tuned method (or even a different method entirely) may be needed in each subsequent environment landscape. Leonard and Engelbrecht emphasize the importance of varying algorithm behavior over time while the problem is being solved.

On the other side of the coin, Karafotias *et al.* [13] note that the *parameter control problem* has not been adequately solved. The survey discusses the '*patchwork problem*, which describes the difficulty in creating good parameter combinations when haphazardly combining multiple individual parameter control methods into a single process. The resulting system is likely suboptimal. Karafotias *et al.* recommend future research should focus on reducing the patchwork problem by developing techniques that operate on multiple parameters simultaneously. The HMHH hyper-heuristic framework strives to be as generic as possible in this regard. Each meta-heuristic managed by HMHH is a fully encapsulated method where 1) each meta-heuristic contains several parameters, 2) each different meta-heuristic may have different *types* of parameters, and 3) each meta-heuristic's parameter choices yield a specific type of behavior.

Selection hyper-heuristics can be compared to a number of parameter control adaptation methods [9], as discussed below:

- *Parameter control* (PC) for EAs [11][12] optimize various EA parameters at run-time. Karafotias *et al.* [13] classify PC mechanisms into: 1) *parameter specific* methods that adapt specific EA parameters such as population size, variation, selection, fitness function modification, or parallel EA parameters; 2) *multiple parameter ensembles* that combine multiple heterogeneous control mechanisms into either a) variation and population, or b) variation and selection combinations; or 3) *parameter independent* methods applicable to any (numeric) EA parameter.

- *Adaptive Operator Selection* (AOS) [10], a specific type of PC, continuously selects the most appropriate EA variation operator to use at time t. A *credit assignment* mechanism rewards different EA variation operators based on observed quality. Sources of feedback can include individuals' fitness or fitness improvement, diversity measures, or EA-specific measures such as offspring survival and population tenure of individuals [13].

- *Adaptive memetic algorithms* (MA) [14] is a hybrid EA approach that self-adaptively combines population-based global search methods and individual local learning approaches. Ong *et al.* [15] provide a classification of adaptation mechanisms for MAs.

- *Self-learning PSO* [56][57] and *heterogeneous PSO* [58][59] are both examples of techniques that adapt the search behaviors of individual particles based on feedback received during the search.

- *Algorithm portfolios*, first proposed by Huberman [16], run different algorithms concurrently. A time-sharing mechanism determines how much computation time each algorithm should receive. Pen *et al.* [60] developed a population based algorithm portfolio (PAP) that offsets the *risk* of spending too much computational *budget* on inferior methods by *diversifying* the optimization process across multiple SI

and EC approaches. Peng *et al.* investigated multiple variations of PAP across various different types of static environments.

- *Ensemble* methods diversify the optimization process by utilizing multiple search strategies and/or parameters. Wu *et al.* [18] present a high-level *ensemble of DE variants* (EDEV) that divide a population of individuals into smaller *indicator* sub-populations and a *larger* reward sub-population. The DE variants compete against each other using performance on the indicator sub-populations. The reward population is adaptively allocated to the best-performing DE algorithm every set number of generations. EDEV is inspired by *multi-population ensemble DE* (MPEDE) [17] that uses a similar approach on low-level DE mutation strategies.

Similar to the approaches above, a selection hyper-heuristic operator continuously adapts the optimization approach at run-time to use the most appropriate optimization strategy at time t. Similar to AOS, a hyper-heuristic relies purely on high-level performance feedback from heuristics operating on the problem domain to decide which heuristics to select next. Unlike the approaches above, a hyper-heuristic is completely shielded from operational details of heuristics or the exact nature of the underlying problem. A hyper-heuristic manages a pool of heterogeneous, independent, and self-contained heuristics that are treated as "black-box" optimization methods. A heuristic may be as simple as a Gaussian mutation operator, or be a complex meta-heuristic implementation that employs any blend of optimization techniques, algorithm parameters, entity topologies, design decisions, and self-adaptation mechanisms. This complexity is fully encapsulated inside the heuristic and the hyper-heuristic is unaware of any of these details.

3. Motivation for Hyper-heuristic Approaches for Dynamic Optimization Problems

Section 2.4 highlighted a contradictory situation faced by DOP-focused meta-heuristics: static parameter tuning of DOP-specific meta-heuristic parameters is unattainable, while mixing together multiple self-adaptive parameter control methods in an ad hoc fashion is typically sub-optimal (i.e. the *patchwork problem*). The HMHH hyper-heuristic framework addresses this conundrum by separating the implementation of the 'building blocks' for DOP approaches discussed in section 2.3 into a *heuristic layer* and *hyper-heuristic layer*[5].

The *heuristic layer* is responsible for *maintaining / introducing diversity* in the problem space, *reacting to change* of the environment, and *managing memory/state* in the problem space. HMHH encapsulates each heuristic as a distinct collection of low-level operators, design decisions, and parameter values. The result is a mix of highly varied modes of operation. For example, an SI approach comprising of specific *neighborhood structures*, *electrostatic charges*, *position update* operators, and *velocity update* operators will exhibit completely different behavior than an EC approach with *cross-over*, *mutation*, *replacement*, and *elitism* operators. Deliberate parameter values generally produce very specific *exploration* and *exploitation* behavior in each meta-heuristic [11][55].

The *hyper-heuristic* layer is responsible for: managing *multiple populations*[6] of candidate solutions that focus on different aspects of the search; *maintaining a memory* of correlations between heuristic allocation, feedback values, and resulting performance; and *self-adaptation* of search behavior by learning how to best allocate computational resources to the most promising heuristics at time t. Every selection operator for HMHH implements these building blocks in different ways. *Detection of change* (as the last building block) is out of scope in this study to avoid bias caused by any variance in change detection strategies. Generally the pool of heuristics would manage change detection in practical applications.

3.1. Motivation for heuristic choices

In this paper, HMHH manages a mix of proven DOP-specific meta-heuristics that implement *exploration* and *exploitation* behavior in different ways. The heuristic pool also contains heuristics that are not traditionally used as stand-alone methods to solve DOPs, but (through parameter choices) are known to exhibit strong convergence behavior. The expectation is that HMHH will appropriately manage resource allocation across these diverse heuristics. Two variants of almost every heuristic are included in the pool, where each variant uses parameter configurations that are known to slightly favor either *exploration* or *exploitation* behavior respectively.

To avoid any bias, the pool of heuristics in this study does not contain any *self-adaptive* heuristics, heuristics that employ multiple *sub-populations*, or heuristics that utilize *explicit memory* of predictable environment behavior. The rationale for these decisions are as follows:

- Self-adaptive heuristics could result in heuristics converging (in heuristic space) on similar types of behavior. Many heuristics would exhibit no noticeable difference in behavior, which would impede the ability of HMHH to alter how the search is conducted. Additionally, many HMHH selection operators employ learning and memory mechanisms to learn which heuristic's behavior is appropriate given specific environmental feedback. Self-adaptive

[5] This is an example of using a component-based view of meta-heuristics, as proposed by Sörensen [41]

[6] As section 2.4 shows, HMHH does not technically maintain different populations but each collection of entities assigned to the same heuristic at time t can be interpreted as a distinct population.

heuristics would continually change their behavior over time, causing the hyper-heuristic's knowledge of the workings of each heuristic to become outdated. Future studies should investigate the trade-off and synergies between these two levels of self-adaptation, and be mindful of not exasperating the *patchwork problem* by carefully managing which parameters are adapted, and in what manner.

- While HMHH can technically utilize heuristics that manage sub-populations of entities, this study is limited to heuristics with single populations. Multi-population heuristics would result in a fragmented 'sub-populations of sub-populations' situation, which would increase the number of entities required for HMHH to be effective. This would greatly diminish any gains in computational efficiency HMHH would bring. Additionally, heuristic state management is simpler in a single-population heuristic approach when entity reallocation occurs.

- Nguyen *et al.* [5] highlight a number of studies that show how the use of environment change prediction could negatively impact the optimization method. The wrong training data, lack of training data, or the very nature of the DOP could all lead to extremely poor performance by wrongly biasing the search to certain areas. The goal of this study is to investigate a broadly applicable hyper-heuristic approach and not to specialize the method to a subset of DOP types that exhibit predictable behavior.

The reasons above exclude many state-of-the-art self-adaptive and/or multi-population methods discussed in section 2.3. However, the overall aim of studying the performance and behavior of different heuristic selection mechanisms is not compromised.

The following heuristics below are used in the study. Table 3 shows the parameters used for each heuristic. Appendix A elaborates on the implementation details for each heuristic, as well as the considerations about how entities are initialized and reassigned between heuristics.

Table 3: Heuristic parameter values

Parameter	Value	Parameter	Value
Atomic PSO (APSO) and Charged PSO (CPSO)			
APSO %swarm charged	$\approx 50\%$	R_p	10
CPSO %swarm charged	100%	R_c	1
Quantum PSO (QPSO)			
% swarm charged	$\approx 50\%$	r_{cloud} values	{1, 10}
PSO shared parameters (APSO, CPSO, QPSO)			
update policy	synchronous	*topology*	gbest
charge	1		
Differential Evolution (DE)			
recombination prob. P_r	{0.3, 0.9}	*scaling factor* β	0.5
on-change mutation rate	0.1	*DE scheme*	*rand/1/bin*
Random Immigrants GA			
replacement rate	0.1	*selection type*	*Elitism*
cross-over type	*Arithmetic*	*mutation*	*Gaussian*
cross-over rate	0.6	*mutation rate*	*0.001*

3.1.1. *Atomic and Charged Particle Swarm Optimization*

Atomic particle swarm optimization (APSO) and *charged particle swarm optimization* (CPSO) were proposed by Blackwell and Bentley [61] as adaptations of the classic PSO algorithm. APSO and CPSO introduce *charged* particles that repel each other based on the magnitude of the distance between them. The resulting 'suspension of charged particles' ensures high diversity throughout the search process. The difference between APSO and CPSO is that all particles in CPSO are charged, whereas only 50% of particles are charged in APSO. The classic PSO velocity equation is modified to include an *acceleration* term $\mathbf{a}_i(t) = \sum_{j=1,j\neq i}^{n_s} \mathbf{a}_{ij}(t)$ for all n_s particles with the repulsion between particles i and j defined as:

$$\mathbf{a}_{ij} = \begin{cases} \frac{Q_i Q_j (\mathbf{x}_i(t) - \mathbf{x}_j(t))}{\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|^3} & \text{if } R_c \leq \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| \leq R_p \\ \frac{Q_i Q_j (\mathbf{x}_i(t) - \mathbf{x}_j(t))}{R_c^2 \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|} & \text{if } \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| < R_c \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where Q_i and Q_j are the charges of particles i and j respectively, $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|$ is the Euclidean distance between particles, and R_c and R_p are respectively the core limit and perception limit of the particle. *Charged* particles have a charge $Q_i > 0$ while *neutral* particles have $Q_i = 0$. Coulomb-like repulsion occurs between pairs of charged particles in the swarm when the Euclidean distance between a pair is in the range $[R_c, R_p]$. No acceleration occurs if the distance between particles is greater than R_p. To avoid extremely large accelerations between any two particles closer than R_c to each other, repulsion is capped at the same value as at R_c.

CPSO has a stronger focus on exploration and maintains a higher solution diversity than APSO, resulting in CPSO working well in spatially severe environments [61]. APSO balances exploration and exploitation more than CPSO, and works well in environments with high temporal change severity [62]. Both variants are included in the heuristic pool with the expectation that HMHH will, as needed, utilize CPSO to foster greater exploration and APSO to yield greater exploitation.

3.1.2. *Quantum Particle Swarm Optimization*

Quantum particle swarm optimization (QPSO) was proposed by Blackwell and Branke as a computationally simplified model inspired by APSO and CPSO [63][64]. The swarm consists of a mix of charged and neutral particles. Instead of using Coulomb-like repulsion between charged particles, the charged particles are simply re-initialized randomly inside an n-dimensional sphere B_n of radius r_{cloud} centered around the particle. The PSO

position update becomes

$$\mathbf{x}(t+1) = \begin{cases} \mathbf{x}(t) + \mathbf{v}(t) & \text{if } Q_i = 0 \\ B_n(r_{cloud}) & \text{if } Q_i \neq 0 \end{cases} \quad (7)$$

where $\mathbf{v}(t)$ is the classic PSO velocity update equation [34]. Blackwell *et al.* [63] comment that CPSO and APSO can be difficult to control, since the spatial extent of the charged swarm depends on the Euclidean distance between charged particles. The $B_n(r_{cloud})$ operator of QPSO, on the contrary, yields sustained and controlled diversification behavior. Larger r_{cloud} values facilitate exploration while smaller values tend to foster exploitation. QPSO is empirically shown to perform as well or better than APSO and CPSO in some classes of DOPs [63][64]. Consequently, this study uses two variants of QPSO as alternative methods to maintain diversity, abbreviated as **QPSO1** and **QPSO10** with $r_{cloud} = 1$ and $r_{cloud} = 10$ respectively.

3.1.3. ***Differential Evolution***

Differential evolution (DE) by Storn and Price [65] is a population-based optimization algorithm for real-valued search. Classic DE has been shown to perform poorly in DOPs – once DE converges it becomes impossible to detect new or moved optima [66]. The classic DE algorithm has been extended with diversity management strategies such as entity reinitialization, quantum individuals, Brownian individuals, or added random noise [34]. State-of-the-art DE methods for DOPs are discussed in section 2.3. Notwithstanding, the *rand/1/bin* DE scheme has been empirically shown to have excellent convergence characteristics in static environments [67][66]. A low P_r value generally decreases convergence speed yet increases robustness, while a larger P_r value often results in faster convergence [34][68].

In this study, DE is included in the heuristic pool to act as an exploiter heuristic, leaving diversity management to other heuristics. The expectation is that the inclusion of DE would foster much faster convergence after a change, as well as rapid re-acquisition of new optima that may appear over time. The decision to choose when to apply DE is left to HMHH selection operators. **DE_H** and **DE_L** represent two *rand/1/bin* DE variants with $P_r = 0.9$ and $P_r = 0.3$ respectively. Gaussian mutation is applied to DE individuals after environment changes using a mutation rate of 0.1. DE requires that $n_s > 2n_v + 1$, where n_s is the number of individuals and n_v is the number of difference vectors ($n_v = 1$ for *rand/1/bin*) [34].

3.1.4. ***Random Immigrants Genetic Algorithm***

The *random immigrants genetic algorithm* (RIGA) by Grefenstette [69] is a modification to the standard GA to allow the algorithm to cope better in DOPs. Both Cruz *et al.* [4] and Nguyen *et al.* [5] list RIGA as a good example of a DOP-specific method that maintains diversity throughout the optimization run. Every generation a subset of the population is replaced by randomly generated individuals according to a *replacement rate*. To allow RIGA to be applied to real-valued problems, *arithmetic cross-over* (as described by Michalewicz [70]) is used instead of uniform cross-over. Arithmetic cross-over requires that two parent entities $\mathbf{x}_1$ and $\mathbf{x}_2$ create an offspring entity $\mathbf{o}$ as follows: $\mathbf{o} = (1-\lambda)\mathbf{x}_1 + \lambda\mathbf{x}_2$, where $\lambda \in [0, 1]$. Grefenstette [69] proposes a replacement rate of 0.1 and a very low mutation rate of 0.001 to prevent too much perturbation of solutions.

In this study, RIGA represents a completely different type of diversification operator that uses hereditary instead of relying on trajectory information as DE or the various PSO variants do. As a heuristic, RIGA continually incorporates a steady stream of fresh, diversified solutions. HMHH can subsequently reassign diversified entities to other heuristics if needed.

3.2. HMHH selection operator choices

This study expands HMHH [28] by using different selection operators and problem space performance measures (i.e. ς and Q_{δ_m} in algorithm 2). Each selection operator for HMHH may employ a mix of deterministic or non-deterministic selection mechanisms, and may utilize internal memory and learning mechanisms to keep track of good entity-to-heuristic assignments.

The following formal definitions are used:

- The set E contains the parent population of all entities, where each entity $e_j \in E$ is a candidate solution.
- $n_s = |E|$ is the total number of entities in HMHH.
- The set H contains one or more heuristics $h_m \in H$.
- $n_h = |H|$ is the total number of heuristics in HMHH.
- The set S contains disjoint subsets of entities from E, where each subset $s_m(t) \in S$ forms a sub-population of the entities assigned to heuristic h_m for k iterations. For any t we have $\cup_{i=1}^{n_h} s_i(t) = E$.
- $n_m = |s_m(t)|$ is the number of entities assigned to sub-population s_m (i.e. to heuristic h_m) at time t.
- The fitness of any entity e_j at time t is denoted as $f_j(t)$.
- $Q_{\delta_m}(t)$ is a problem space performance measure that determines the quality of heuristic h_m at time t based on the previous k iterations. $Q_{\delta_m}(t)$ is calculated in different ways by various selection operators using different types of feedback.
- The selection operator $\varsigma(e_j, Q_{\delta_m}(t))$ uses $Q_{\delta_m}(t)$ to assign entity $e_j \in E$ to the most suitable heuristic h_m at time t.
- $\Theta(t) = (P_1, P_2, ..., P_{n_h})$ is a list of probabilities of selecting each heuristic h_m at time t. Various selection operators use $Q_{\delta_m}(t)$ to calculate $\Theta(t)$ in unique ways.

HMHH initially assigns the n_s entities to heuristics randomly, and candidate solutions are randomly generated within the function domain. Each heuristic $h_m \in H$ is executed independently for k algorithm iterations and operates only on the sub-population of entities s_m associated with h_m. After k iterations, all entities in all sub-populations $s_m(t) \in S$ are returned to the parent population E to be reassigned by the HMHH selection operator. Each of the 10 selection operators below determines the new entity assignments in different ways, i.e. $\Theta(t)$ is calculated differently by each selection operator.

Heuristic space convergence occurs when all entities are allocated to a single heuristic, $h_c \in H$. Other heuristics $h_m \in H$ for $m \neq c$ will then have zero assigned entities. If this happens, heuristics with zero entities will never improve any entities and $Q_{\delta_m}(t) = 0$ for those heuristics. Some selection operators may subsequently be unable to improve the probability of selection of any heuristic $h_m \in H$ for $m \neq c$, resulting in that heuristic never being chosen again. To avoid this situation, each heuristic maintains a minimum entity count. Since *rand/1/bin* DE requires a minimum of four entities (see subsection 3.1.3), the minimum entity count is set to four for each heuristic (i.e. $n_m \geq 4$).

When an entity e_j previously assigned to heuristic h_m is assigned to a different heuristic h_{m^*} for the next k iterations, the entity must operate as part of the sub-population $s_{m^*}(t)$ of the new heuristic h_{m^*}. The entity may require additional configuration, and must be initialized with the context and state information required by h_{m^*}. The simplest strategy is to initialize a new member of the sub-population $s_{m^*}(t)$ and update the new entity with the candidate solution and fitness information from the old entity e_j. That is, if entity e_j acted as a DE individual for the previous k iterations and is assigned to operate as a CPSO particle for the next k iterations, the entity must be supplied with 1) a charge value, 2) a personal best value, 3) neighborhood information, and 4) particle velocity information. If entity e_j is reassigned to the same heuristic h_m, the entity simply keeps any heuristic-specific state it acquired (if applicable). Appendix A describes how each heuristic semantically decorates reassigned entities.

The following 10 selection operators for HMHH are included in this study:

- *Fixed* selection (HH_Fix) never changes the original entity allocations, i.e. s_m remains constant for each heuristic h_m. Probabilities of selection $\Theta(t)$ are ignored.

- *Simple random* selection (HH_Rand) assigns each entity $e_j \in E$ to heuristic h_m with equal probability, i.e. the same constant $\Theta(t)$ is always used where each $P_m = \frac{1}{n_h}$.

- *Roulette wheel* selection (HH_Roul) assigns each entity $e_j \in E$ to heuristic h_m with a probability based on the performance of entities in s_m relative to the performance of entities in all heuristics. The mean fitness of entities in s_m is

$$\Upsilon_m(t) = \frac{\sum_{j=1}^{n_m} f_j(t)}{n_m} \tag{8}$$

and each P_m in $\Theta(t)$ is set to $P_m = \Upsilon_m(t) \,/\, \sum_{l=1}^{n_h} \Upsilon_l$. The same $\Theta(t)$ is used to assign all entities.

- *Tournament* selection (HH_Tour) holds a tournament to select the best heuristic. A new tournament set $T_j \subset H$ of heuristics is randomly selected for each entity $e_j \in E$. The tournament size is two. The mean fitness, Υ_m, of entities assigned to heuristic $h_m \in T_j$ is computed using equation (8). The winner is that heuristic $h_w \in T_j$ that has the highest Υ_m value (for maximization). $\Theta(t)$ is set to $P_m = 1$ for $m = w$ and $P_m = 0$ for $m \neq w$. $\Theta(t)$ may vary per entity.

- *Ant-inspired rank-based* selection [58] (HH_ARank) is inspired by the ant colony optimization metaheuristic [71]. Each heuristic h_m is assigned a *pheromone concentration* $\rho_m(t)$ as relevancy score that is used to calculate $\Theta(t)$ using

$$P_m(t) = \frac{\rho_m(t)}{\sum_{l=1}^{n_h} \rho_l(t)} \tag{9}$$

Roulette wheel selection uses $\Theta(t)$ to assign each entity $e_j \in E$ to a heuristic. Initially, each heuristic h_m has a pheromone concentration $\rho_m(1) = \frac{1}{n_h}$. Pheromone levels are updated based on whether the fitness of each entity $e_j \in s_m$ increased, decreased, or stagnated over the previous k iterations, i.e.

$$\rho_m(t) = \rho_m(t-k) + \sum_{j=1}^{n_m} \begin{cases} 1.0 & \text{if } f_j \text{ improved} \\ 0.5 & \text{if } f_j \text{ remained the same} \\ 0.0 & \text{if } f_j \text{ worsened} \end{cases} \tag{10}$$

Pheromone concentrations are evaporated every k iterations to avoid the build-up of extremely large scores, i.e.

$$\rho_m(t) \leftarrow \frac{\sum_{l=1, l\neq i}^{n_h} \rho_l(t)}{\sum_{l=1}^{n_h} \rho_l(t)} \times \rho_m(t) \tag{11}$$

- *Ant-inspired proportional* selection [58] (HH_AProp) is similar to ant-inspired rank-based selection, but $\rho_m(t)$ is updated using the fitness improvement of entities $e_j \in s_m$ over the previous k iterations (for maximization), i.e.

$$\rho_m(t) = \rho_m(t-k) + \sum_{j=1}^{n_m} (f_j(t) - f_j(t-k)) \tag{12}$$

- *Frequency improvement* selection (HH_Freq) was used by Nepomuceno and Engelbrecht's *frequency-based heterogeneous PSO* behavior selection scheme

(FB-HPSO) [59] to probabilistically select particle behaviors based on the frequency with which each behavior improves the fitness of assigned particles over a period. FB-HPSO is adapted here to select the best heuristic for each entity $e_j \in E$. A frequency score, $\chi_m(t)$, is calculated for each heuristic h_m based on how many times each entity $e_j \in s_m$ improved its fitness over the previous k iterations, i.e.

$$\chi_m(t) = \sum_{m=1}^{k} \sum_{j=1}^{n_m} \begin{cases} +1 & \text{if } f_j \text{ improved} \\ -1 & \text{if } f_j \text{ remained the same} \\ -1 & \text{if } f_j \text{ worsened} \end{cases} \tag{13}$$

Entities $e_j \in E$ are assigned a heuristic using tournament selection using $\chi_m(t)$ instead of $\Upsilon_m(t)$ from equation (8).

- *Frequency improvement reinforcement learning* selection (HH_ReinfFr) applies a reinforcement learning approach similar to Narayek [72] and Burke *et al.* [54]. A rank score r_m is maintained for each heuristic. Heuristics are rewarded or punished based on the frequency with which the heuristics improve the fitness of assigned entities. Initially, each heuristic h_m has a rank $r_m = r_{min} = 0$. The change in rank for h_m is $\Delta r_m(t) = \chi_m(t)$, where $\chi_m(t)$ is the frequency score in equation (13). Since the maximum range of $\chi_m(t)$ is $[-(n_s \times k), (n_s \times k)]$ the maximum rank $r_{max} = n_s \times k$. Ranks are updated as $r_m(t) = r_m(t-k) + \Delta r_m(t)$. Every entity $e_j \in E$ is assigned to the highest ranked heuristic (essentially all other heuristics are on the tabu list as per Burke's approach [54]). Values in $\Theta(t)$ are set to $P_m = 1$ for $m = r$ for the highest ranked heuristic $h_r \in H$ while $P_m = 0$ for $m \neq r$. Rank ties are broken randomly. The same $\Theta(t)$ is used to assign all entities.

- *Fitness proportional reinforcement learning* selection (HH_ReinfPr) is similar to *frequency improvement reinforcement learning* but uses the mean *change* in fitness of entities $e_j \in s_m$ assigned to heuristic h_m, namely $\phi_m(t)$, to reinforce ranks as (for maximization)

$$\phi_m(t) = \frac{\sum_{j=1}^{n_m} (f_j(t) - f_j(t-k))}{n_m} \tag{14}$$

and the change in rank $\Delta r_m(t)$ for each heuristic h_m is

$$\Delta r_m(t) = \begin{cases} +1 & \text{if } \phi_m(t) > 0 \\ -1 & \text{if } \phi_m(t) = 0 \\ -1 & \text{if } \phi_m(t) < 0 \end{cases} \tag{15}$$

Each heuristic's rank is updated as $r_m(t) = r_m(t-k) + \Delta r_m(t)$. The maximum rank is $r_{max} = n_h$ since the maximum rank change $r_\Delta(t) = \pm 1$. Every heuristic h_m has an initial rank of $r_m = r_{min} = 0$. Every entity $e_j \in E$ is assigned to the highest ranked heuristic (essentially all other heuristics are on the tabu list as per Burke's approach [54]). Values in $\Theta(t)$ are set to $P_m = 1$ for $m = r$ for the highest ranked heuristic $h_r \in H$ while $P_m = 0$ for $m \neq r$. Rank ties are broken randomly. The same $\Theta(t)$ is used to assign all entities. *Fitness proportional reinforcement learning* ranks heuristics using mean entity fitness changes over the previous k iterations (regardless of the number of improving moves), while *frequency improvement reinforcement learning* rewards the frequency with which a heuristic improved fitness over the previous k iterations (regardless of fitness change magnitudes).

- *Difference proportional* selection [73] (HH_DiffPr) by Spanevello and Montes de Oca probabilistically assigns each entity $e_j \in E$ to the sub-population s_b of the heuristic h_b that contains the fittest entity e_b. *Difference proportional* selection increases the probability of assigning poor performing entities to h_b, and lowers the probability of reassigning well-performing entities to different heuristics. The probability $P_b(t)$ of reassigning entity e_j to h_b at time t is

$$P_b(t) = \frac{1}{1 + \exp\left(-\beta \frac{f_{e_b}(t) - f_j(t)}{\text{abs}(f_{e_b}(t))}\right)} \tag{16}$$

where β is Euler's number e, which is lower than Spanevello and Montes de Oca's values of $\beta = 5$ or $\beta = 10$ due to the incredibly high selection pressure of their original values. Values in $\Theta(t)$ are set to $P_m = P_b(t)$ for $m = b$ and $P_m = 0$ for $m \neq b$. $\Theta(t)$ is recomputed for every entity.

4. Experimental Procedure

It is vital to determine if any increased performance of any hyper-heuristic is simply due to *speciation* or *random chance*, or a result of using an intelligent selection operator. As recommended by Karafotias *et al.* [13], multiple control groups are used:

1. *Single-population stand-alone* configurations of each heuristic to test if hyper-heuristic selection yields any benefit over the individual heuristics.
2. *Homogeneous speciation* configurations where the *same* heuristic algorithm manages seven independent, fixed sub-populations of entities. This approach tests if hyper-heuristic selection's increased performance is simply due to speciation alone, and if intelligent heuristic selection can increase performance even further.
3. *Fixed heuristic selection*, represented by **HH_Fix**, excludes any heuristic selection to test if intelligent hyper-heuristic selection yields any benefit over not relocating entities between *different* heuristics.

4. *Random heuristic selection*, represented by **HH_Random**, acts as a randomized control strategy to test if intelligent heuristic selection raises performance further than randomly relocating entities between heuristics.

The stand-alone heuristics are denoted by the abbreviations defined in section 3.1 (i.e. **CPSO**), and the homogeneous speciation versions use the same abbreviations prepended with "**S_**", i.e. **S_CPSO**. All heuristics across all control groups used the same literature-recommended parameter configurations outlined in table 3. No parameter tuning was performed on any heuristic for any of the 27 environments. The MPB generator was used to create instances of the 27 classes of DOPs discussed in section 2.1. Each algorithm was run with 100 entities for 1000 iterations on 50 random instances of each environment. The following measures were used:

- The *fitness error*, $\varepsilon(e,t) = f(\vec{x}^*,t) - f(e,t)$ (for maximization), is the difference in fitness of entity e at time t compared to the global optimum $f(\vec{x}^*,t)$ at time t.
- *Heuristic space diversity,* $\mathcal{H}(t)$, measures the spread of entities across different heuristics using equation (5).
- The number of entities allocated to each heuristic over time.
- The number of entities, ΔE, that were assigned to a *different* heuristic compared to the previous k iterations.

The normalized wins-minus-losses approach of Helbig and Engelbrecht outlined in section 2.2 was applied to the $\varepsilon(e,t)$ values recorded at the end of every change period.

5. Results

To reiterate, the purpose of the experiments is to 1) determine how well HMHH balances computational resources across heuristics to improve performance in DOPs beyond that of using any of the individual heuristics in isolation, and 2) better understand the heuristic allocation behavior that leads to any such improved performance. The focus is not on comparing hyper-heuristics to state-of-the-art DOP algorithms, nor to determine the best factors that characterize a good heuristic pool. This is left for future work. Subsection 5.1 discusses the performance of all hyper-heuristics relative to the control groups outlined above. Subsection 5.2 explores the relationships between increased performance and entity allocation behavior using the $\mathcal{H}(t)$, ΔE, and entity-to-heuristic allocation measures outlined above.

5.1. Overall performance

Table 4 shows the normalized wins-minus-losses for each algorithm in each environment, as well as the mean (μ) and standard deviation (σ) of the normalized wins-minus-losses for each algorithm. The bottom nine rows of table 4 respectively show the total normalized wins-minus-losses for each algorithm across each *type* of dynamism using the naming convention presented in section 2.1, i.e. '$A**$' shows the total for *Abrupt* environments. Tables 5, 6 and 7 respectively show the *collective mean error* (CME) [74] of each environment for the hyper-heuristics, stand-alone heuristics, and homogeneous speciated heuristics.

The hyper-heuristics generally performed well relative to each of the control groups:

Hyper-heuristics versus stand-alone heuristics. Table 4 shows that each hyper-heuristic ranked higher overall than any of the stand-alone heuristics. Generally, the hyper-heuristics had lower standard deviations in wins-minus-losses than the stand-alone heuristics, which shows that the hyper-heuristic approaches were more stable across all environments than any of the stand-alone heuristics. Considering each type of dynamism in isolation, the bottom nine rows of table 4 shows that every hyper-heuristic had substantially higher wins-minus-losses than any of the individual heuristics. This is not a hard rule however, and exceptions do occur in certain individual environments. In general, using a hyper-heuristic significantly improved wins-minus-losses performance and stability relative to using any of the hyper-heuristics in isolation.

Hyper-heuristics versus homogeneous speciation. **S_CPSO** and **S_APSO** performed substantially better than the other homogeneous speciation approaches. Apart from **S_CPSO** and **S_APSO**, each hyper-heuristic outperformed every homogeneous speciation approach overall. Half of the hyper-heuristics (namely **HH_Fix**, **HH_Tour**, **HH_Freq**, **HH_ReinfFr**, and **HH_DiffPr**) did outperform every homogeneous speciation approaches overall. Every hyper-heuristic had a significantly lower standard deviation in wins-minus-losses than any of the speciation approaches. The bottom nine rows of table 4 substantiate this, as every hyper-heuristic showed high positive wins-minus-losses across dynamism types, while the speciation approaches showed turbulent behavior with large positive and negative values across dynamism types. Generally, using a hyper-heuristic resulted in more stable performance compared to using a speciated approach.

Hyper-heuristics versus fixed heterogeneous speciation. **HH_Fix** was ranked fourth overall and showed the second-lowest standard deviation in rank value in table 4. Simply statically assigning entities to multiple different heuristic outperformed any of those same individual heuristics ran in either stand-alone or multi-population speciated configurations. However, **HH_DiffPr**, **HH_Freq**, and **HH_Tour** did manage to outperform **HH_Fix** overall.

Table 4: Average wins-minus-losses for $\varepsilon(e,t)$ measured at the end of each change period, per environment. Bold values indicate the best performer per environment.

		Hyper-heuristic selection operators										Single population stand-alone heuristics							Homogeneous speciation approaches						
	Env	**HH_Fix**	**HH_Rand**	**HH_Roul**	**HH_Tour**	**HH_ARank**	**HH_AProp**	**HH_Freq**	**HH_ReinfFr**	**HH_ReinfPr**	**HH_DiffPr**	**CPSO**	**APSO**	**QPSO1**	**QPSO10**	**DE_L**	**DE_H**	**RIGA**	**S_CPSO**	**S_APSO**	**S_QPSO1**	**S_QPSO10**	**S_DE_L**	**S_DE_H**	**S_RIGA**
	A1C	7.05	4.65	4.15	6.45	4.55	5.65	5.85	6.85	5.40	7.65	-8.25	-5.50	-1.30	-12.15	6.80	**7.80**	-14.30	6.90	6.40	5.70	-8.65	-9.25	-16.80	-15.65
	A1L	3.75	3.90	4.95	4.65	6.25	6.85	**7.50**	4.50	5.20	6.85	0.25	-5.25	0.10	-10.85	5.30	4.90	-15.25	5.15	5.05	6.00	-7.60	-8.55	-14.95	-18.70
	A1R	3.55	3.80	6.75	5.40	2.75	4.35	5.10	3.45	4.50	**8.95**	-5.00	-1.80	-0.15	-10.85	1.35	4.90	-13.60	5.85	5.65	6.40	-7.15	-1.40	-12.90	-19.90
	A2C	4.45	0.20	0.50	1.05	0.10	0.05	0.55	2.40	2.20	**4.95**	-4.90	-3.80	-3.65	-11.00	0.00	-2.70	-1.35	1.90	2.90	3.95	-0.60	1.85	3.55	-2.60
	A2L	3.15	2.10	-0.30	1.50	0.65	2.60	2.25	6.40	4.70	**9.40**	-7.25	-7.60	-0.65	-16.85	2.50	1.35	-1.95	0.40	-0.80	3.85	-9.55	8.15	4.95	-9.00
	A2R	3.25	0.25	1.10	1.85	0.95	1.25	0.70	1.40	2.05	2.00	-5.20	-3.10	-2.65	-14.10	1.15	0.95	-0.70	1.35	2.40	**4.55**	-2.25	4.00	2.85	-4.05
	A3C	2.80	3.60	3.10	4.30	3.15	2.90	4.40	2.75	2.90	5.25	-3.25	-0.85	0.85	-14.10	1.95	1.70	-3.60	3.55	3.30	**6.35**	1.55	-6.35	-15.10	-11.15
	A3L	3.20	3.80	2.75	3.00	3.80	4.05	4.65	3.60	4.20	5.20	-0.75	0.25	-0.65	-9.85	2.95	2.60	-11.60	4.95	4.90	**6.70**	0.70	-10.40	-13.95	-14.10
	A3R	3.10	2.05	1.70	2.25	4.45	2.30	2.95	2.85	2.75	3.60	-1.30	-1.00	0.30	-13.70	0.90	0.90	-6.75	4.05	2.50	**6.35**	-1.30	-0.15	-5.30	-13.50
	C1C	5.85	**8.51**	5.97	6.31	6.23	6.24	6.85	6.37	5.11	7.97	-1.67	1.93	1.96	-10.14	4.76	-5.46	-18.64	8.47	7.83	3.93	-7.41	-11.69	-19.62	-19.66
	C1L	6.14	7.19	9.02	8.37	7.42	10.61	**11.26**	5.10	-1.56	9.03	3.04	6.56	2.37	-7.22	-5.46	-3.16	-17.13	9.57	9.92	-15.95	0.71	-18.54	-19.93	-17.36
	C1R	6.61	5.92	**9.31**	7.40	7.22	3.59	8.94	3.19	-0.45	7.39	-1.51	-0.94	4.89	-12.22	0.55	0.31	-20.26	5.61	8.24	9.21	-7.16	-8.55	-16.47	-20.82
	C2C	6.71	-1.10	-1.45	-0.42	-1.27	-1.11	0.35	4.51	3.02	3.03	-4.21	-3.99	-4.09	-10.36	-0.38	-0.48	-7.55	4.65	3.61	6.42	-2.50	**7.83**	3.64	-4.86
	C2L	8.91	7.56	8.00	7.71	8.29	7.85	**9.66**	7.79	8.77	8.79	-13.75	-13.88	-11.88	-20.93	6.93	8.29	-5.76	-7.85	-7.50	-6.81	-14.79	7.76	9.50	-12.66
	C2R	**7.39**	-0.74	-0.57	0.46	-0.35	-0.23	0.06	4.20	2.38	1.73	-5.42	-6.25	-3.16	-11.15	-0.67	-0.27	-4.25	3.61	3.70	3.49	-2.59	6.57	6.50	-4.44
	C3C	5.55	4.36	3.26	4.64	4.51	4.06	3.74	5.02	2.79	4.63	0.02	1.03	-1.22	-2.80	1.61	-9.00	-14.55	8.69	**8.88**	4.07	3.52	-9.33	-16.53	-16.95
	C3L	5.52	5.29	5.14	6.15	5.68	6.27	6.17	4.99	-0.03	6.12	3.28	3.84	1.34	-1.47	-1.69	-1.14	-13.82	10.00	**10.39**	-16.42	3.90	-15.27	-18.82	-15.42
	C3R	5.04	2.97	2.47	3.74	2.90	3.94	3.87	2.67	0.09	4.66	0.45	-0.60	-0.02	-5.09	-1.69	-2.08	-14.79	**8.92**	6.92	5.30	2.87	-4.65	-10.89	-17.00
	P1C	5.32	5.90	4.86	6.00	4.99	5.86	7.00	5.01	5.93	**8.50**	5.65	4.12	-18.15	3.37	4.17	-0.90	-18.39	6.20	6.88	-17.38	4.39	-5.33	-15.65	-18.35
	P1L	5.82	7.41	5.86	6.50	7.29	5.34	5.89	5.87	6.70	7.09	6.09	7.56	-21.92	5.15	-2.18	1.13	-16.26	**9.10**	7.12	-20.31	7.32	-15.04	-15.28	-16.23
	P1R	2.74	1.42	2.94	3.11	2.62	1.66	1.30	3.00	2.72	2.29	1.14	1.61	0.58	-9.24	2.08	3.54	-14.10	2.25	-0.32	2.83	-0.46	2.17	**3.97**	-19.85
	P2C	1.67	0.21	0.07	0.12	-0.20	-0.12	0.14	1.75	0.89	0.63	-1.12	-0.93	-1.15	-3.60	-0.63	-1.21	-1.12	0.97	0.68	**2.24**	0.57	0.09	2.09	-2.05
	P2L	-2.93	6.04	10.11	**15.50**	11.14	11.97	13.61	10.68	12.61	11.91	-13.00	-13.47	-6.97	-18.07	-2.29	-1.82	-2.17	-6.03	-9.21	-0.99	-14.34	-0.85	-0.73	-10.71
	P2R	**1.92**	1.08	-0.09	1.41	0.92	0.89	0.87	**1.92**	1.29	1.34	-0.23	-0.84	-0.80	-9.65	-0.15	-0.04	-2.51	1.82	0.93	0.48	-0.55	0.89	0.84	-1.73
	P3C	5.47	5.69	5.81	5.94	5.21	5.14	5.66	5.00	4.58	6.63	3.56	3.96	-16.04	1.05	-4.77	-2.64	-15.96	7.10	**7.61**	-14.40	6.26	-4.09	-10.48	-16.27
	P3L	6.11	6.47	6.20	6.50	5.87	6.42	6.86	5.45	6.79	7.46	6.77	6.76	-22.13	4.81	-5.59	1.64	-14.98	**7.62**	7.39	-20.22	7.36	-12.03	-15.64	-15.88
	P3R	1.35	-0.08	0.15	0.93	0.12	0.04	0.50	1.52	0.06	1.05	0.50	-0.74	0.16	-1.56	-0.50	0.54	-6.23	2.49	**2.87**	2.04	0.89	0.16	1.65	-7.91
	Total	**119.5**	**98.4**	**101.8**	**120.8**	**105.2**	**108.4**	**126.7**	**118.3**	**95.6**	**154.1**	**-46.1**	**-32.9**	**-104**	**-222.6**	**17**	**9.6**	**-277.6**	**117.3**	**108.3**	**-22.6**	**-46.9**	**-102**	**-199.5**	**-346.8**
	μ	**4.43**	**3.65**	**3.77**	**4.47**	**3.90**	**4.02**	**4.69**	**4.38**	**3.54**	**5.71**	**-1.71**	**-1.22**	**-3.85**	**-8.24**	**0.63**	**0.36**	**-10.28**	**4.34**	**4.01**	**-0.84**	**-1.74**	**-3.78**	**-7.39**	**-12.84**
	σ	*2.42*	*2.81*	*3.28*	*3.38*	*3.11*	*3.26*	*3.62*	*2.13*	*3.08*	*2.99*	*5.16*	*5.33*	*7.43*	*6.87*	*3.30*	*3.67*	*6.44*	*4.32*	*4.68*	*9.58*	*6.04*	*7.57*	*9.80*	*6.13*
	Rank	**4**	**11**	**10**	**3**	**9**	**7**	**2**	**5**	**12**	**1**	**17**	**16**	**20**	**22**	**13**	**14**	**23**	**6**	**8**	**15**	**18**	**19**	**21**	**24**
	A**	34.30	24.35	24.70	30.45	26.65	30.00	33.95	34.20	33.90	**53.85**	-35.65	-28.65	-7.80	-113.45	22.90	22.40	-69.10	34.10	32.30	49.85	-34.85	-22.10	-67.65	-108.65
	C**	**57.72**	39.96	41.15	44.36	40.63	41.22	50.90	43.84	20.12	53.35	-19.77	-12.30	-9.81	-81.38	3.96	-12.99	-116.75	51.67	51.99	-6.76	-23.45	-45.87	-82.62	-129.17
	P**	27.46	34.13	35.92	46.00	37.95	37.20	41.83	40.21	41.56	**46.89**	9.37	8.03	-86.42	-27.75	-9.86	0.23	-91.72	31.52	23.96	-65.71	11.44	-34.03	-49.24	-108.97
	1	46.83	48.69	53.81	54.19	49.32	50.15	59.69	43.34	33.54	**65.72**	-0.25	8.29	-31.62	-64.15	17.36	13.06	-147.93	59.10	56.77	-19.57	-26.01	-76.18	-127.63	-166.51
	2	34.51	15.60	17.38	29.18	20.22	23.15	28.19	41.05	37.91	**43.77**	-55.08	-53.86	-35.00	-115.72	6.47	4.07	-27.36	0.82	-3.28	17.19	-46.59	36.29	33.19	-52.10
	3	38.14	34.15	30.58	37.44	35.69	35.12	38.80	33.85	24.13	44.60	9.28	12.65	-37.40	-42.71	-6.84	-7.48	-102.28	**57.37**	54.76	-20.23	25.74	-62.11	-105.06	-128.18
	**C	44.86	32.02	26.27	34.39	27.27	28.67	34.54	39.66	32.81	**49.24**	-14.17	-4.03	-42.79	-59.73	13.51	-12.89	-95.46	48.43	48.09	0.87	-2.87	-36.27	-84.90	-107.54
	**L	39.67	49.75	51.74	59.87	56.39	61.96	67.85	54.38	47.38	**71.84**	-15.33	-15.23	-60.39	-75.29	0.47	13.79	-98.92	32.92	27.27	-64.15	-26.29	-64.77	-84.85	-130.06
	**R	34.95	16.67	23.76	26.55	21.57	17.78	24.29	24.21	15.39	33.01	-16.56	-13.66	-0.85	-87.57	3.02	8.74	-83.19	35.94	32.89	**40.65**	-17.70	-0.96	-29.76	-109.20

Table 5: Mean and standard deviations of the CME for the hyper-heuristics calculated over 50 runs for each environment.

Env	HH_Fix		HH_Rand		HH_Roul		HH_Tour		HH_ARank		HH_AProp		HH_Freq		HH_ReinfFr		HH_ReinfPr		HH_DiffPr	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
A1C	7.6	6.2	9.2	7.1	10.5	8.2	8.3	7.9	9.9	7.7	9.1	8.4	8.3	7.2	8.1	6.5	10.5	7.0	**7.2**	**6.7**
A1L	10.8	7.3	10.1	7.4	10.1	7.9	10.4	8.6	8.6	7.2	8.0	6.1	**7.1**	**5.5**	10.8	6.7	11.9	6.8	8.0	7.0
A1R	9.3	6.1	9.6	7.8	8.4	7.3	8.9	7.2	10.2	7.4	8.8	5.8	8.2	6.7	10.5	5.9	10.7	6.0	**6.9**	**4.6**
A2C	12.0	4.0	16.9	3.7	16.9	4.5	16.1	4.0	16.9	3.3	17.1	4.2	16.2	4.9	13.5	4.9	14.1	4.5	**11.5**	**4.5**
A2L	8.5	7.1	10.0	9.0	12.7	8.5	11.1	9.2	12.4	9.5	9.6	8.1	9.9	8.0	6.4	6.8	7.9	7.5	**6.0**	**5.8**
A2R	**9.9**	**5.8**	13.9	7.6	12.1	5.7	11.1	5.6	11.8	6.7	11.9	6.2	13.0	6.9	11.5	6.3	10.4	5.4	10.5	5.7
A3C	13.0	6.1	12.0	5.8	13.1	7.0	11.6	6.0	12.6	6.0	13.2	6.4	11.3	5.4	13.4	5.8	14.4	6.4	**10.9**	**5.5**
A3L	13.3	5.5	12.4	5.9	13.9	5.9	13.7	5.9	12.2	5.4	12.5	5.4	11.6	4.7	12.8	4.6	13.5	4.4	**11.0**	**5.5**
A3R	12.0	5.1	13.6	5.9	13.9	5.2	13.4	5.5	**11.2**	**5.7**	12.7	6.5	12.5	5.4	13.1	5.1	14.4	5.7	12.3	6.5
C1C	9.4	4.5	8.4	6.5	10.6	7.3	9.7	5.9	10.1	7.7	10.3	7.2	9.1	5.4	9.7	5.8	11.4	5.1	**7.4**	**4.6**
C1L	11.2	4.4	12.0	5.4	10.5	4.8	11.0	4.5	11.4	5.0	9.6	4.0	**9.3**	**3.9**	13.9	6.7	17.7	5.3	9.8	4.6
C1R	10.0	4.4	10.9	6.3	**9.2**	**4.2**	10.9	5.8	10.1	5.3	13.3	6.2	9.8	5.3	13.5	6.6	14.8	4.6	9.9	4.8
C2C	**10.7**	**3.7**	17.5	1.1	17.8	1.3	16.6	1.7	17.3	1.0	17.5	1.4	15.6	2.6	11.5	3.3	13.1	3.1	12.5	3.5
C2L	**3.1**	**2.7**	7.5	9.2	5.9	7.1	6.2	7.1	5.2	6.8	5.5	6.6	3.6	5.3	4.4	6.0	5.3	9.3	**3.1**	**2.7**
C2R	**8.8**	**3.6**	15.7	4.5	15.7	4.2	14.5	4.0	15.6	3.8	15.2	3.9	15.3	4.0	10.6	3.8	12.2	3.7	12.7	3.6
C3C	**14.6**	**4.5**	16.5	3.6	18.3	3.6	16.5	2.9	16.5	3.9	17.4	4.2	17.6	3.5	15.6	3.4	19.8	3.9	16.0	4.4
C3L	16.6	3.3	18.1	2.5	18.1	3.2	17.3	2.9	17.9	2.7	17.0	3.4	16.9	2.8	17.9	4.2	24.8	4.7	**16.4**	**3.5**
C3R	**15.6**	**3.6**	18.6	4.3	19.4	4.0	17.7	2.9	18.7	3.8	17.5	3.2	17.5	3.3	18.5	3.6	21.3	3.9	16.1	3.7
P1C	11.4	8.3	10.1	8.3	12.7	10.7	10.5	10.8	11.7	9.1	10.2	7.9	8.9	7.6	12.0	8.3	10.8	10.5	**8.6**	**9.2**
P1L	12.4	7.7	**10.1**	**7.0**	12.7	9.8	10.9	7.8	10.2	8.1	12.7	7.4	12.4	8.6	13.4	9.1	10.4	6.3	10.4	8.3
P1R	9.0	6.5	11.5	10.2	8.7	7.1	8.7	7.6	10.1	10.1	11.0	8.9	12.0	11.1	**8.7**	**6.7**	9.6	8.6	10.4	8.4
P2C	13.0	3.3	16.6	1.0	17.0	1.0	16.1	1.3	17.0	1.3	17.0	1.0	15.9	2.2	**12.4**	**4.0**	13.9	3.2	13.9	3.6
P2L	9.7	13.6	11.2	13.3	6.8	10.9	**3.9**	**7.2**	8.5	12.0	7.5	11.2	6.6	10.5	8.0	13.2	8.2	12.7	7.6	11.5
P2R	10.6	6.4	12.9	6.5	15.4	6.2	11.2	6.2	12.6	7.0	12.7	6.2	12.8	7.1	**9.8**	**5.9**	12.0	5.7	10.7	6.4
P3C	14.8	8.0	13.8	6.8	13.6	5.7	13.9	6.6	14.5	6.5	14.4	6.8	13.9	5.8	15.5	6.8	15.8	7.2	**12.8**	**6.0**
P3L	15.0	7.4	14.0	5.2	15.4	9.5	14.0	6.0	15.4	6.3	14.3	5.9	13.2	4.7	17.4	7.6	13.5	5.9	**12.5**	**5.8**
P3R	**12.4**	**5.8**	16.5	6.8	15.2	5.9	13.7	6.3	16.0	7.1	16.5	7.2	14.3	5.7	12.5	5.5	15.8	6.5	13.9	8.8

Clearly, an intelligent selection operator can raise performance above that of a fixed heterogeneous speciation approach.

Hyper-heuristics versus random heterogeneous speciation. Every hyper-heuristic except **HH_ReinfPr** outperformed **HH_Rand** overall. Using intelligent hyper-heuristics significantly improved performance over blindly allocating resources in an ad hoc fashion. However, overall **HH_Rand** still outperformed every stand-alone heuristic, and every homogeneous speciated approach apart from **S_CPSO** and **S_APSO**. A very low standard deviation in wins-minus-losses compared to any of these approaches shows that **HH_Rand** could consistently produce competitive results in most environments versus the hit-or-miss nature of the stand-alone or speciated approaches.

5.2. Heuristic diversity and entity allocation behavior

Tables 8, 9 and 10 respectively show the mean and standard deviation of entity allocations to individual heuristics, $\mathcal{H}(t)$, and ΔE for each of the 27 environments. Figure 2 shows illustrative plots from the A3R environment of typical entity allocations over time for each selection operator.

Tables 8, 9 and 10 show general patterns per hyper-heuristic:

- **HH_Fix** and **HH_Rand** each (respectively) had nearly identical entity allocation, $\mathcal{H}(t)$, and ΔE values across all 27 environments. In essence, **HH_Fix** and **HH_Rand** only differ in ΔE behavior, where **HH_Fix** never reallocated entities and **HH_Rand** reallocated entities every k iterations. Regardless of the timing of entity allocation, both approaches are invariant with respect to either the environment or heuristic performance feedback.
- **HH_Roul**, **HH_Tour**, **HH_ARank**, **HH_AProp**, and **HH_Freq** each showed high $\mathcal{H}(t)$ and ΔE values, with reasonably similar values per selection operator across all environments. This shows that these methods continually reallocated most entities across all heuristics, without any heuristic space convergence to a single method.
- **HH_ReinfFr**, **HH_ReinfPr** and **HH_DiffPr** each had substantially different allocation counts and ΔE values in each of the 27 environments. Mean and standard deviation values for $\mathcal{H}(t)$ for each selection operator are nearly identical (roughly 18.2). Such low $\mathcal{H}(t)$ values indicate that **HH_ReinfFr**, **HH_ReinfPr** and **HH_DiffPr** each tended to allocate all entities to a single heuristic (i.e. *heuristic space convergence* occurred).

Figure 2 illustrates the above behaviors well.

The resemblance between the entity allocation behavior of **HH_Rand**, **HH_ARank**, and **HH_AProp** is striking, and can be seen visually in figure 2. **HH_Rand**, **HH_ARank**, and **HH_AProp** each used roulette wheel selection to allocate entities to heuristics. **HH_Rand** used fixed uniform random probabilities, while the ant-based

Table 6: Mean and standard deviations of the CME for the stand-alone heuristics calculated over 50 runs for each environment.

Env	**CPSO**		**APSO**		**QPSO1**		**QPSO10**		**DE_L**		**DE_H**		**RIGA**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
A1C	15.0	8.3	12.6	7.3	14.9	11.8	16.5	10.3	9.2	6.0	**8.8**	**5.9**	19.5	7.4
A1L	**11.6**	**7.5**	13.1	8.3	13.6	10.5	14.6	8.3	12.4	7.1	12.0	7.2	21.8	5.7
A1R	14.3	8.8	**11.1**	**7.8**	13.0	9.8	15.2	9.3	13.5	7.0	11.5	7.9	18.5	6.5
A2C	22.0	5.2	22.3	6.2	19.7	4.4	24.1	3.4	**17.0**	**4.0**	19.4	3.5	18.3	4.1
A2L	15.0	9.0	14.8	9.2	13.2	11.1	21.1	11.7	**10.6**	**8.2**	11.4	10.4	13.3	7.6
A2R	16.7	5.8	16.2	7.0	17.3	8.3	21.2	7.7	**11.6**	**6.1**	12.8	6.8	13.7	5.0
A3C	18.9	7.0	17.2	7.8	17.4	8.2	22.7	7.5	**15.8**	**6.8**	17.0	9.4	20.2	5.7
A3L	16.1	6.9	**14.8**	**6.6**	18.5	9.7	19.2	7.1	15.9	6.8	16.6	6.0	23.4	5.1
A3R	**16.1**	**6.7**	16.3	6.8	16.6	8.8	22.1	7.8	17.5	6.2	17.9	7.2	21.0	5.5
C1C	12.2	6.7	**10.3**	**4.4**	19.1	13.3	16.8	8.9	12.2	7.5	21.4	13.8	28.8	8.4
C1L	10.5	4.3	**10.0**	**5.1**	26.8	19.4	15.7	7.2	19.4	6.6	18.3	6.4	33.2	3.7
C1R	12.6	7.3	**12.5**	**6.7**	16.2	10.7	17.8	9.6	14.5	5.5	14.8	7.4	28.7	4.3
C2C	20.1	2.1	20.1	2.6	22.0	7.7	24.2	1.3	**16.8**	**2.5**	17.7	1.3	23.6	4.6
C2L	12.1	12.4	11.8	11.5	21.3	18.8	26.9	16.7	7.8	11.0	10.9	15.4	**6.7**	**5.3**
C2R	19.7	4.1	20.0	4.3	18.4	4.7	22.9	4.3	**15.6**	**4.4**	15.8	4.3	19.0	5.3
C3C	**20.5**	**3.9**	19.6	4.3	27.6	13.3	22.5	5.5	20.6	4.8	31.9	11.2	31.8	5.0
C3L	18.9	3.3	**18.2**	**3.2**	29.9	13.9	21.8	4.1	28.1	7.6	26.2	5.2	33.7	2.5
C3R	**19.6**	**4.3**	20.2	3.4	26.3	8.2	22.6	4.4	22.9	3.7	23.4	4.5	30.8	3.8
P1C	**10.6**	**8.8**	14.0	9.2	35.8	20.8	13.5	9.3	13.6	8.7	16.3	9.7	28.6	9.3
P1L	12.0	8.9	**9.6**	**6.2**	57.2	5.3	13.8	8.8	18.0	9.7	16.6	8.9	33.8	10.1
P1R	12.4	9.0	11.0	7.5	13.3	11.0	16.1	9.8	10.7	9.0	**8.6**	**8.0**	17.3	7.2
P2C	19.0	7.3	18.0	2.8	19.1	7.6	20.9	2.3	**16.7**	**2.2**	17.6	1.2	19.5	2.3
P2L	13.2	15.5	15.5	18.3	19.0	18.4	20.5	18.4	10.3	13.7	12.8	16.7	**9.5**	**10.4**
P2R	16.5	8.8	17.2	11.7	17.7	8.9	21.2	9.1	14.7	7.9	**14.1**	**5.4**	16.2	7.1
P3C	16.7	8.2	**16.7**	**5.8**	35.0	17.4	18.9	6.8	22.1	7.6	22.1	9.5	29.6	8.0
P3L	13.6	5.2	**13.6**	**4.9**	58.1	5.4	17.0	6.4	23.8	9.3	19.6	8.0	31.4	7.4
P3R	15.3	6.9	17.5	6.8	15.2	6.3	18.3	6.5	17.2	7.7	**14.7**	**7.1**	20.7	5.5

Table 7: Mean and standard deviations of the CME for the homogeneous speciation approaches calculated over 50 runs for each environment.

Env	**S_CPSO**		**S_APSO**		**S_QPSO1**		**S_QPSO10**		**S_DE_L**		**S_DE_H**		**S_RIGA**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
A1C	**7.9**	**6.0**	9.0	8.7	10.0	6.7	13.2	6.1	22.8	12.4	30.7	13.5	20.5	6.8
A1L	9.6	7.4	9.6	6.2	**9.3**	**5.3**	13.0	6.1	19.4	7.2	25.4	9.9	23.2	5.2
A1R	8.1	6.7	**7.8**	**5.6**	8.5	6.2	12.5	6.4	14.0	5.8	18.6	7.7	22.5	6.8
A2C	14.5	6.5	13.6	7.2	13.5	6.5	16.7	4.8	13.9	4.9	**12.9**	**5.2**	18.5	3.6
A2L	9.7	7.4	11.0	9.5	9.0	7.6	13.7	6.8	**6.6**	**6.5**	8.6	9.5	14.8	6.6
A2R	11.9	7.5	10.9	6.1	9.6	6.4	13.8	5.3	**9.0**	**4.2**	9.6	6.5	15.2	5.6
A3C	12.5	7.5	12.4	6.5	**10.5**	**5.3**	13.3	4.3	25.4	11.6	32.3	14.4	21.8	4.7
A3L	10.8	5.6	11.0	5.7	**10.8**	**5.1**	13.5	4.1	25.5	8.1	29.0	10.9	22.9	4.6
A3R	**10.9**	**5.8**	12.5	5.1	11.4	5.8	15.1	5.5	17.8	5.8	20.2	7.8	22.8	4.8
C1C	**7.4**	**4.6**	7.5	3.7	15.4	10.2	13.2	4.8	25.7	12.3	40.3	13.8	30.9	7.4
C1L	9.1	3.7	**8.6**	**2.9**	35.7	13.0	10.2	2.5	45.7	5.7	50.7	4.8	33.7	4.5
C1R	10.0	5.1	**8.8**	**4.0**	13.1	4.0	13.3	4.3	17.6	4.5	22.4	6.5	29.7	5.8
C2C	11.5	4.0	12.2	4.1	11.9	3.6	17.2	3.9	**10.7**	**3.9**	12.4	4.2	19.9	5.1
C2L	6.9	9.3	5.5	7.1	10.1	12.5	9.7	5.3	**5.2**	**7.8**	7.3	12.8	9.2	5.0
C2R	11.0	4.3	10.6	3.3	11.3	4.8	15.3	3.7	9.9	4.3	**9.4**	**4.8**	17.5	5.5
C3C	11.9	3.6	**11.9**	**2.8**	19.8	10.7	16.1	3.5	32.6	11.8	40.8	12.4	34.4	5.0
C3L	**12.7**	**2.1**	12.4	2.4	43.7	9.3	16.0	2.9	44.7	5.5	51.2	3.7	35.7	3.0
C3R	**12.7**	**3.7**	13.8	3.9	19.0	4.2	15.7	3.7	24.4	5.0	28.7	6.7	32.0	4.4
P1C	9.8	7.7	**9.1**	**8.1**	31.0	15.0	12.6	8.3	19.6	11.6	26.4	13.7	29.5	10.8
P1L	**8.3**	**6.5**	10.5	7.2	48.8	6.5	10.1	6.2	29.2	9.5	34.4	9.7	32.9	10.3
P1R	12.1	13.9	16.4	13.6	9.4	8.5	12.4	6.9	10.3	7.9	**8.2**	**7.3**	21.3	7.2
P2C	14.8	8.9	15.3	9.6	12.9	6.1	14.5	4.0	15.0	4.0	**12.3**	**4.4**	17.8	4.6
P2L	6.9	12.3	9.7	14.1	9.8	15.7	9.3	9.6	**5.4**	**8.0**	7.7	12.2	8.2	5.0
P2R	**11.0**	**7.5**	12.1	10.3	11.9	7.4	15.8	10.7	12.4	5.7	12.2	7.3	14.4	6.8
P3C	**12.4**	**6.2**	12.6	6.3	31.4	13.1	13.0	5.4	22.6	8.7	26.9	9.3	31.1	9.9
P3L	**11.8**	**4.5**	12.3	4.6	48.3	7.3	12.5	5.5	27.3	6.2	36.4	7.1	34.1	9.0
P3R	12.7	8.2	12.8	6.3	12.7	7.3	14.0	6.8	14.7	6.3	**12.5**	**5.6**	21.5	7.6

Table 8: Mean and standard deviations of $\mathcal{H}(t)$ and entity to heuristic allocations (all abrupt environments).

Alg	HH_Fix		HH_Rand		HH_Roul		HH_Tour		HH_ARank		HH_AProp		HH_Freq		HH_ReinfFr		HH_ReinfPr		HH_DiffPr		Env
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	
APSO	14.5	3.0	14.3	3.5	12.3	6.0	10.3	6.1	12.7	3.5	11.4	3.7	15.3	6.5	6.9	14.1	7.2	14.7	24.4	32.4	
CPSO	13.8	3.2	14.2	3.5	12.5	5.7	9.3	5.6	11.0	3.3	10.6	3.5	13.9	6.6	4.3	3.9	6.7	13.6	20.5	30.1	
DE_H	14.0	2.4	14.2	3.5	16.1	6.6	20.9	6.2	16.5	3.8	17.9	4.7	6.2	3.2	4.2	2.8	26.2	33.2	6.5	13.1	
DE_L	14.2	2.4	14.3	3.5	17.4	9.3	15.6	6.6	16.4	3.9	15.4	4.5	5.7	3.1	4.2	3.3	23.1	31.7	6.2	12.2	
QPSO1	14.6	2.8	14.3	3.5	13.2	6.7	14.6	6.9	14.3	3.6	15.2	4.3	18.6	6.1	25.0	32.7	6.9	14.2	11.3	21.5	A1C
QPSO10	14.7	3.1	14.3	3.5	11.1	5.6	5.8	4.4	11.6	3.4	9.6	4.2	13.4	6.4	9.3	18.7	8.3	16.9	24.7	32.5	
RIGA	14.3	3.0	14.4	3.5	17.5	8.7	23.5	8.2	17.6	4.0	19.9	4.9	26.8	4.6	46.1	35.4	21.6	30.9	6.4	12.6	
$\mathcal{H}(t)$	**89.4**	**3.6**	**86.9**	**3.9**	**76.2**	**16.6**	**65.0**	**5.0**	**83.5**	**4.8**	**78.9**	**6.9**	**65.2**	**4.9**	**18.1**	**5.0**	**18.1**	**5.0**	**18.3**	**5.3**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*77.0*	*17.1*	*76.6*	*6.9*	*84.7*	*7.0*	*83.5*	*7.2*	*74.9*	*6.8*	*2.2*	*12.2*	*42.0*	*35.4*	*0.9*	*6.3*	
APSO	14.1	3.1	14.4	3.5	12.5	6.3	10.9	6.3	12.7	3.4	11.9	3.8	15.6	6.4	7.5	15.3	7.3	14.9	19.7	29.6	
CPSO	14.4	2.8	14.3	3.5	11.3	6.3	10.1	6.2	11.2	3.3	11.2	3.5	14.2	6.2	5.0	8.1	6.0	11.6	18.3	28.6	
DE_H	14.1	2.9	14.3	3.5	16.0	6.9	18.4	7.8	16.4	3.9	17.2	4.4	5.9	3.0	4.2	2.8	25.8	33.0	7.2	14.7	
DE_L	14.1	2.7	14.3	3.5	17.7	9.0	15.0	6.8	16.4	3.9	14.7	4.3	5.6	2.9	4.2	2.8	22.9	31.6	7.2	14.5	
QPSO1	14.1	2.7	14.2	3.5	12.7	6.9	15.2	7.7	14.4	3.7	16.0	4.4	18.9	6.0	24.1	32.3	7.2	14.7	12.6	23.1	A1L
QPSO10	14.4	2.6	14.2	3.5	12.3	6.5	6.3	4.7	11.5	3.4	9.6	4.0	12.9	6.3	13.3	24.1	5.6	10.5	29.8	34.4	
RIGA	14.7	3.2	14.3	3.5	17.4	7.2	24.2	7.2	17.3	4.0	19.3	4.7	26.8	4.6	41.8	35.9	25.4	32.8	5.3	9.3	
$\mathcal{H}(t)$	**89.0**	**3.5**	**87.0**	**3.9**	**75.1**	**16.4**	**64.9**	**5.0**	**83.7**	**4.8**	**80.1**	**6.4**	**65.2**	**4.9**	**18.1**	**5.0**	**18.1**	**5.0**	**18.3**	**5.3**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*76.4*	*15.9*	*76.7*	*7.0*	*84.8*	*7.0*	*83.9*	*7.2*	*74.9*	*6.8*	*2.6*	*13.4*	*41.6*	*35.5*	*1.1*	*6.9*	
APSO	14.0	3.1	14.3	3.5	13.2	6.4	10.7	6.4	12.7	3.5	11.8	3.7	15.5	6.5	4.3	4.2	7.3	15.0	21.0	30.4	
CPSO	13.8	2.9	14.4	3.5	12.9	5.7	10.2	6.1	11.1	3.3	11.1	3.8	14.2	6.3	4.4	5.0	6.3	12.4	24.9	32.6	
DE_H	15.2	3.1	14.2	3.4	16.7	7.4	19.7	7.0	16.4	3.9	17.2	4.4	6.0	2.9	4.1	2.6	26.6	33.4	5.0	8.3	
DE_L	13.2	2.8	14.2	3.5	16.4	5.7	15.2	6.6	16.4	3.9	14.8	4.3	5.6	2.9	4.2	3.2	21.2	30.6	9.4	18.7	
QPSO1	15.2	3.6	14.3	3.5	11.6	6.1	14.1	7.9	14.5	3.7	15.7	4.5	18.4	6.6	23.3	31.8	7.8	15.9	11.7	22.1	A1R
QPSO10	14.4	2.9	14.4	3.5	12.5	5.8	6.2	4.6	11.5	3.5	9.8	4.2	13.4	6.3	10.6	20.7	7.0	14.2	23.3	31.7	
RIGA	14.2	2.7	14.2	3.5	16.6	6.3	23.8	7.8	17.4	3.9	19.4	4.7	26.9	4.7	49.1	34.8	23.9	32.1	4.7	6.5	
$\mathcal{H}(t)$	**88.5**	**3.4**	**87.0**	**3.9**	**78.4**	**14.0**	**64.9**	**5.0**	**83.6**	**4.8**	**80.0**	**6.6**	**65.1**	**4.9**	**18.1**	**5.1**	**18.1**	**5.0**	**18.3**	**5.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*79.1*	*13.8*	*76.7*	*7.0*	*84.8*	*7.0*	*83.8*	*7.2*	*74.9*	*6.8*	*3.6*	*15.5*	*42.7*	*35.3*	*1.0*	*6.8*	
APSO	14.8	2.9	14.3	3.5	12.2	6.0	9.9	5.8	12.8	3.5	10.6	3.7	15.2	6.8	10.5	20.5	8.3	16.9	15.4	26.1	
CPSO	14.2	3.1	14.3	3.5	12.0	5.9	9.4	5.5	10.8	3.3	10.3	3.7	15.0	6.5	8.3	16.9	9.9	19.7	17.2	27.7	
DE_H	14.5	2.7	14.3	3.5	16.5	5.6	20.9	7.3	16.8	3.9	19.2	4.9	6.1	3.1	4.8	7.3	24.5	32.5	12.7	23.3	
DE_L	13.4	2.6	14.3	3.5	16.2	6.0	16.0	6.6	16.3	3.9	16.6	5.0	5.3	2.6	4.6	6.3	21.5	30.9	9.7	19.2	
QPSO1	13.5	2.7	14.3	3.5	14.2	6.6	12.6	5.7	13.8	3.6	14.3	4.5	17.6	6.6	18.8	29.1	9.9	19.6	12.2	22.6	A2C
QPSO10	15.1	2.9	14.3	3.5	11.5	6.3	6.3	4.8	12.0	3.5	9.1	4.2	14.8	6.4	18.2	28.6	8.8	17.8	23.6	31.9	
RIGA	14.6	2.7	14.3	3.5	17.4	6.3	25.0	6.3	17.6	4.0	19.8	5.0	26.0	5.8	34.9	35.6	17.1	27.7	9.2	18.4	
$\mathcal{H}(t)$	**89.2**	**3.2**	**87.0**	**3.9**	**77.6**	**13.5**	**64.9**	**4.9**	**83.4**	**4.8**	**76.9**	**7.0**	**65.3**	**5.1**	**18.1**	**5.0**	**18.1**	**5.0**	**18.4**	**5.4**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*79.0*	*12.6*	*76.3*	*6.9*	*84.7*	*7.0*	*83.0*	*7.3*	*74.9*	*6.8*	*8.5*	*23.1*	*34.6*	*35.9*	*1.6*	*8.6*	
APSO	14.9	3.0	14.3	3.5	12.2	8.6	9.8	5.9	12.6	3.5	10.8	3.7	15.2	7.1	10.2	20.1	9.1	18.5	13.7	24.4	
CPSO	13.5	2.8	14.3	3.5	10.9	6.8	9.2	5.6	10.6	3.3	10.4	3.7	15.4	6.6	8.7	17.7	8.7	17.7	21.5	30.7	
DE_H	14.0	3.4	14.4	3.5	17.0	7.7	21.6	6.5	17.0	4.0	19.0	5.1	6.1	3.1	5.1	8.7	24.7	32.5	7.3	15.0	
DE_L	14.8	2.8	14.2	3.5	18.2	9.4	16.8	5.8	16.6	4.0	16.5	4.8	5.6	2.9	5.1	8.6	20.5	30.2	13.8	24.5	
QPSO1	14.0	2.4	14.3	3.5	11.8	7.5	11.4	5.6	13.4	3.6	14.3	4.4	18.1	6.4	27.3	33.6	12.3	22.9	9.6	19.1	A2L
QPSO10	14.7	3.1	14.3	3.5	12.0	7.3	6.0	4.5	12.0	3.4	9.1	4.1	14.6	6.7	17.3	27.9	9.1	18.3	23.9	32.1	
RIGA	14.1	3.2	14.3	3.5	18.0	7.8	25.1	5.5	17.8	4.0	19.9	5.0	25.1	6.9	26.3	33.2	15.6	26.4	10.3	20.1	
$\mathcal{H}(t)$	**89.0**	**3.7**	**87.0**	**3.9**	**71.5**	**17.7**	**65.2**	**5.3**	**82.9**	**5.0**	**77.2**	**7.1**	**65.2**	**4.8**	**18.1**	**5.0**	**18.1**	**5.0**	**18.3**	**5.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*73.5*	*17.6*	*76.4*	*6.9*	*84.6*	*7.0*	*83.1*	*7.2*	*74.9*	*6.8*	*12.6*	*27.3*	*35.4*	*36.0*	*0.8*	*6.2*	
APSO	14.4	3.1	14.3	3.5	11.9	5.8	10.2	5.8	12.6	3.5	10.7	3.7	15.7	6.8	10.2	20.1	11.1	21.4	17.1	27.5	
CPSO	13.9	2.7	14.3	3.5	12.1	5.5	9.4	5.5	10.7	3.2	10.6	3.8	15.3	6.5	7.1	14.5	9.1	18.4	14.7	25.4	
DE_H	14.4	3.4	14.3	3.5	16.4	6.0	20.6	7.5	16.8	4.0	19.0	5.0	6.2	3.1	5.1	8.7	23.5	32.0	13.8	24.5	
DE_L	14.7	3.0	14.3	3.5	16.7	5.4	15.7	6.8	16.4	3.9	16.4	5.0	5.2	2.6	5.0	8.3	19.7	29.7	13.3	23.9	
QPSO1	15.2	3.1	14.3	3.5	14.3	6.1	12.0	5.8	13.8	3.6	14.3	4.4	17.5	6.7	24.5	32.4	10.8	21.0	11.1	21.2	A2R
QPSO10	13.5	2.8	14.3	3.5	11.9	5.5	6.5	4.9	11.9	3.5	9.2	4.1	14.6	6.6	14.3	25.1	10.2	20.1	22.6	31.3	
RIGA	14.0	2.9	14.3	3.5	16.8	5.4	25.7	5.1	17.8	4.0	19.8	5.0	25.6	6.5	33.8	35.4	15.5	26.3	7.4	15.1	
$\mathcal{H}(t)$	**88.5**	**2.7**	**86.9**	**3.9**	**78.9**	**12.0**	**65.0**	**4.9**	**83.2**	**4.9**	**77.3**	**6.9**	**65.2**	**5.0**	**18.1**	**5.1**	**18.1**	**5.1**	**18.5**	**5.5**	
ΔE	*0.0*	*0.0*	*85.2*	*7.0*	*80.0*	*11.5*	*76.3*	*6.9*	*84.7*	*7.0*	*83.1*	*7.2*	*74.9*	*6.8*	*12.5*	*27.2*	*35.4*	*36.0*	*1.7*	*8.9*	
APSO	14.2	2.4	14.2	3.5	12.1	6.0	10.4	6.2	12.6	3.5	11.7	3.8	15.6	6.4	4.3	3.9	7.5	15.4	20.3	29.9	
CPSO	14.4	2.7	14.3	3.5	11.8	5.7	9.5	5.8	10.8	3.3	11.0	3.6	14.5	6.3	5.6	10.6	7.3	14.9	17.2	27.6	
DE_H	13.9	3.0	14.3	3.5	16.1	7.3	21.0	6.4	16.5	3.9	17.8	4.6	6.1	3.1	4.3	3.9	25.4	32.9	5.5	9.9	
DE_L	14.6	2.6	14.2	3.5	17.6	8.9	16.2	6.2	16.5	3.8	15.1	4.5	5.7	2.9	4.2	3.6	22.4	31.4	7.9	16.1	
QPSO1	14.1	2.5	14.3	3.5	12.6	6.5	13.7	7.6	14.4	3.7	15.4	4.5	18.0	6.8	24.1	32.3	7.4	15.1	16.3	26.8	A3C
QPSO10	14.1	2.9	14.3	3.5	12.0	6.1	6.6	5.0	11.6	3.5	9.7	4.1	13.4	6.5	9.4	18.8	6.6	13.3	23.9	32.0	
RIGA	14.7	2.6	14.3	3.5	17.8	8.6	22.5	9.1	17.5	4.0	19.3	4.7	26.7	4.9	48.1	35.0	23.4	31.9	8.9	17.8	
$\mathcal{H}(t)$	**89.9**	**2.6**	**86.9**	**3.9**	**75.9**	**16.6**	**64.9**	**4.8**	**83.3**	**4.9**	**79.5**	**6.7**	**65.2**	**5.1**	**18.1**	**5.1**	**18.1**	**5.0**	**18.5**	**5.4**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*76.7*	*17.2*	*76.6*	*6.9*	*84.7*	*7.0*	*83.7*	*7.2*	*75.0*	*6.8*	*2.9*	*14.0*	*43.6*	*35.2*	*1.8*	*8.8*	
APSO	14.2	3.5	14.3	3.5	12.3	6.1	10.7	6.4	12.8	3.5	12.0	3.8	15.2	6.6	6.5	12.9	6.3	12.5	22.1	31.1	
CPSO	14.0	2.4	14.3	3.5	11.2	6.0	9.6	6.1	11.2	3.3	11.4	3.7	14.6	6.3	4.2	3.4	6.7	13.6	22.3	31.2	
DE_H	14.9	2.9	14.3	3.5	16.4	6.4	19.6	6.8	16.4	3.9	17.0	4.6	6.1	3.1	4.2	3.8	25.3	32.8	5.5	10.2	
DE_L	14.1	2.9	14.3	3.5	18.0	9.0	14.1	7.0	16.3	3.8	14.4	4.4	5.9	3.1	4.2	3.3	25.0	32.7	6.9	14.0	
QPSO1	14.9	2.6	14.3	3.5	12.2	6.5	15.4	7.4	14.5	3.7	16.0	4.5	18.3	6.6	26.4	33.3	6.5	13.0	16.8	27.3	A3L
QPSO10	13.7	3.1	14.3	3.5	12.1	6.3	6.4	4.8	11.5	3.4	9.9	4.3	13.1	6.6	6.6	13.2	5.7	10.8	18.2	28.4	
RIGA	14.2	2.5	14.3	3.5	17.7	6.5	24.2	7.4	17.3	4.0	19.4	4.8	26.8	4.7	48.0	35.1	24.6	32.5	8.2	16.5	
$\mathcal{H}(t)$	**89.0**	**2.8**	**87.0**	**4.0**	**75.4**	**15.4**	**65.0**	**5.0**	**83.7**	**4.9**	**80.1**	**6.8**	**65.2**	**4.9**	**18.1**	**5.1**	**18.1**	**5.0**	**18.4**	**5.4**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*77.0*	*14.9*	*76.7*	*6.9*	*84.7*	*7.0*	*83.8*	*7.3*	*74.8*	*6.8*	*3.1*	*14.4*	*44.9*	*34.8*	*1.5*	*8.3*	
APSO	14.1	3.1	14.3	3.5	12.4	7.1	10.1	6.4	12.8	3.5	11.8	3.7	14.8	6.6	6.5	13.1	6.9	14.0	28.4	33.8	
CPSO	14.2	2.9	14.3	3.5	11.7	5.7	9.5	6.0	11.3	3.3	11.2	3.7	14.4	6.6	5.7	10.7	8.0	16.4	16.1	26.6	
DE_H	14.0	2.8	14.4	3.5	16.3	5.9	20.1	6.5	16.4	3.9	17.4	4.5	6.3	3.3	4.2	3.4	24.3	32.3	5.7	10.7	
DE_L	14.3	3.3	14.3	3.5	16.9	7.8	15.6	6.2	16.3	3.9	14.7	4.2	5.8	3.1	4.1	2.7	24.5	32.4	6.9	14.0	
QPSO1	15.1	2.7	14.2	3.6	13.4	6.5	15.2	7.4	14.4	3.6	16.0	4.4	18.3	6.5	21.5	30.8	7.2	14.8	17.0	27.5	A3R
QPSO10	14.1	3.2	14.2	3.5	12.2	6.2	6.1	4.7	11.5	3.5	9.6	4.0	13.6	6.3	11.2	21.5	6.1	12.0	17.8	28.1	
RIGA	14.2	3.0	14.3	3.5	17.2	6.7	23.3	8.3	17.4	4.0	19.3	4.7	26.8	4.6	46.8	35.3	23.0	31.7	8.1	16.3	
$\mathcal{H}(t)$	**88.4**	**3.2**	**86.9**	**3.9**	**76.9**	**15.3**	**65.0**	**4.9**	**83.6**	**4.8**	**80.0**	**6.5**	**65.2**	**4.9**	**18.1**	**5.0**	**18.1**	**5.0**	**18.5**	**5.5**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*78.2*	*14.8*	*76.7*	*6.9*	*84.8*	*7.0*	*83.8*	*7.2*	*74.9*	*6.8*	*3.2*	*14.7*	*45.0*	*34.8*	*1.8*	*9.0*	

Table 9: Mean and standard deviations of $\mathcal{H}(t)$ and entity to heuristic allocations (all chaotic environments).

Alg	**HH_Fix**		**HH_Rand**		**HH_Roul**		**HH_Tour**		**HH_ARank**		**HH_AProp**		**HH_Freq**		**HH_ReinfFr**		**HH_ReinfPr**		**HH_DiffPr**		**Env**
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	
APSO	14.4	3.2	14.2	3.5	13.3	4.7	12.0	7.0	14.0	3.7	12.4	3.9	14.9	5.9	7.9	16.3	10.2	20.1	26.5	33.2	
CPSO	14.2	3.1	14.3	3.4	13.5	4.5	10.8	7.0	12.5	3.6	11.3	3.7	13.9	6.0	6.6	13.3	11.4	21.8	25.1	32.6	
DE_H	13.7	3.3	14.3	3.5	15.3	5.3	19.2	7.2	15.1	3.8	16.2	4.4	6.0	3.2	4.2	3.4	21.2	30.7	5.9	11.2	
DE_L	13.4	3.1	14.2	3.5	14.7	5.4	13.1	7.1	14.6	3.7	13.3	4.1	5.9	3.4	4.2	3.1	24.8	32.6	6.7	13.3	
QPSO1	15.5	3.9	14.3	3.5	15.0	5.8	19.3	8.2	16.2	3.9	16.9	4.7	20.8	5.5	34.4	35.5	6.8	13.8	9.2	18.4	C1C
QPSO10	14.2	2.9	14.3	3.5	13.0	5.9	6.3	5.0	12.4	3.7	11.0	4.3	11.8	5.8	12.7	23.4	11.0	21.3	22.0	31.0	
RIGA	14.5	3.3	14.3	3.5	15.2	5.0	19.3	8.9	15.2	3.9	18.9	4.8	26.7	4.7	30.0	34.5	14.6	25.4	4.7	6.5	
$\mathcal{H}(t)$	**87.4**	**3.0**	**87.0**	**3.9**	**82.8**	**11.2**	**65.0**	**5.0**	**85.1**	**4.5**	**80.9**	**6.8**	**65.3**	**4.9**	**18.1**	**5.1**	**18.1**	**5.1**	**18.3**	**5.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*82.7*	*11.3*	*77.6*	*7.0*	*84.9*	*7.0*	*83.9*	*7.2*	*74.8*	*6.8*	*3.2*	*14.6*	*19.8*	*32.1*	*1.2*	*7.7*	
APSO	14.4	3.0	14.3	3.5	14.3	4.7	13.9	7.7	14.1	3.7	12.8	3.9	14.5	6.0	12.4	23.0	9.3	18.7	22.8	31.5	
CPSO	14.4	2.8	14.3	3.5	13.5	5.2	12.9	7.7	13.6	3.6	12.5	3.8	14.6	5.9	8.5	17.3	10.6	20.7	26.6	33.3	
DE_H	13.8	3.0	14.3	3.5	14.9	5.0	16.3	8.0	14.7	3.7	15.3	4.2	5.6	2.9	4.3	4.6	20.6	30.3	5.6	10.2	
DE_L	14.5	3.1	14.3	3.5	14.6	5.5	10.8	6.6	14.3	3.6	12.1	3.9	6.2	3.5	4.3	3.9	20.4	30.2	5.8	11.0	
QPSO1	13.8	3.2	14.3	3.5	14.6	5.2	20.4	8.5	15.7	3.9	17.0	4.7	21.0	5.5	23.4	31.9	6.5	13.1	11.1	21.3	C1L
QPSO10	14.3	2.7	14.3	3.5	13.0	4.8	7.3	5.7	12.7	3.7	11.6	4.4	11.5	5.7	18.6	28.9	12.1	22.7	23.7	32.0	
RIGA	14.8	3.2	14.3	3.5	15.2	5.4	18.4	8.7	14.9	3.8	18.7	4.7	26.6	4.9	28.5	34.1	20.5	30.2	4.4	5.1	
$\mathcal{H}(t)$	**88.7**	**3.4**	**87.0**	**3.9**	**83.2**	**10.6**	**65.0**	**4.9**	**85.8**	**4.3**	**82.0**	**6.8**	**65.2**	**4.9**	**18.1**	**5.0**	**18.1**	**5.1**	**18.3**	**5.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*82.9*	*10.7*	*77.4*	*7.0*	*85.1*	*7.0*	*84.2*	*7.2*	*74.9*	*6.8*	*3.7*	*15.9*	*14.1*	*28.6*	*1.1*	*7.2*	
APSO	14.1	2.9	14.2	3.4	14.0	5.0	13.4	7.8	14.3	3.7	13.0	3.8	14.5	6.0	16.5	27.3	8.6	17.4	24.8	32.5	
CPSO	13.8	3.2	14.3	3.4	14.1	5.1	13.0	7.6	13.5	3.6	12.3	3.7	14.3	5.9	10.7	20.9	7.5	15.4	25.9	32.9	
DE_H	14.2	3.2	14.3	3.5	15.0	4.7	16.3	8.0	14.7	3.7	15.2	4.1	5.9	3.0	4.2	3.1	22.1	31.2	5.9	11.2	
DE_L	14.6	3.5	14.3	3.5	14.8	4.8	12.0	6.8	14.5	3.7	12.6	3.9	6.3	3.6	4.2	3.0	29.1	34.3	6.0	11.6	
QPSO1	15.0	3.2	14.3	3.5	14.1	5.3	20.5	8.6	16.1	3.9	17.3	4.8	21.1	5.4	25.0	32.7	7.2	14.8	13.0	23.6	C1R
QPSO10	14.5	3.1	14.3	3.5	12.7	5.1	6.6	5.3	12.4	3.7	11.4	4.4	11.4	5.9	7.9	16.2	7.7	15.8	19.7	29.6	
RIGA	13.8	2.9	14.3	3.6	15.4	5.3	18.1	8.7	14.6	3.7	18.2	4.7	26.7	4.7	31.5	34.9	17.8	28.3	4.8	7.2	
$\mathcal{H}(t)$	**88.3**	**3.8**	**87.0**	**3.9**	**82.8**	**10.1**	**64.9**	**4.9**	**85.7**	**4.3**	**82.0**	**6.7**	**65.2**	**4.9**	**18.1**	**5.1**	**18.1**	**5.0**	**18.3**	**5.3**	
ΔE	*0.0*	*0.0*	*85.2*	*7.0*	*82.6*	*10.1*	*77.5*	*7.0*	*85.1*	*7.0*	*84.2*	*7.2*	*74.8*	*6.8*	*5.1*	*18.4*	*32.9*	*35.8*	*1.3*	*7.9*	
APSO	14.7	2.7	14.2	3.5	13.0	4.7	9.5	5.8	13.4	3.8	10.9	3.7	15.5	6.7	9.0	18.2	9.2	18.6	19.2	28.9	
CPSO	14.2	3.4	14.3	3.5	12.7	4.6	8.9	5.4	10.7	3.3	10.4	3.6	14.9	6.5	6.9	14.1	8.8	17.8	14.5	24.9	
DE_H	14.6	3.2	14.3	3.5	16.0	5.8	21.1	6.2	15.8	3.9	18.5	4.7	6.2	3.0	4.6	6.4	24.5	32.5	12.9	23.1	
DE_L	14.1	2.7	14.3	3.5	15.5	5.2	16.6	5.5	15.4	3.8	16.1	4.7	5.1	2.4	4.6	6.2	26.8	33.4	11.8	21.8	
QPSO1	14.0	3.1	14.2	3.5	15.0	4.7	12.1	5.6	15.1	3.9	14.7	4.3	17.8	6.4	19.2	29.3	8.2	16.8	9.3	18.4	C2C
QPSO10	14.1	3.3	14.3	3.5	11.6	5.2	5.7	4.3	12.8	3.8	9.8	4.2	14.5	6.3	16.1	26.9	8.7	17.8	23.1	31.4	
RIGA	14.3	3.2	14.3	3.5	16.1	5.5	26.1	5.2	16.7	4.0	19.6	4.8	26.1	5.9	39.6	36.0	13.8	24.6	9.4	18.7	
$\mathcal{H}(t)$	**88.3**	**3.6**	**86.9**	**4.0**	**82.0**	**10.4**	**64.9**	**4.9**	**84.1**	**4.7**	**78.5**	**6.9**	**65.2**	**4.9**	**18.1**	**5.0**	**18.1**	**5.0**	**19.0**	**5.9**	
ΔE	*0.0*	*0.0*	*85.4*	*7.0*	*82.5*	*11.0*	*76.3*	*6.9*	*84.9*	*7.0*	*83.4*	*7.2*	*74.8*	*6.8*	*7.7*	*22.1*	*37.3*	*35.9*	*4.5*	*14.2*	
APSO	14.5	2.5	14.3	3.5	13.5	4.5	9.8	5.8	12.9	3.6	10.8	3.6	15.7	6.9	9.5	19.0	10.1	19.9	12.4	22.9	
CPSO	15.1	2.8	14.2	3.5	13.2	4.9	9.0	5.3	10.9	3.3	10.3	3.7	15.5	6.6	8.6	17.5	9.3	18.7	29.0	34.2	
DE_H	14.2	3.0	14.3	3.5	15.4	4.7	20.6	6.8	16.3	3.9	19.1	5.0	6.3	3.3	4.9	7.6	20.0	29.9	6.1	12.0	
DE_L	14.2	2.5	14.3	3.5	15.6	5.7	16.5	6.0	15.6	3.8	16.3	4.8	5.4	2.7	4.9	7.6	23.8	32.1	10.2	20.0	
QPSO1	13.9	3.6	14.4	3.5	14.3	5.0	12.7	5.5	14.0	3.7	14.3	4.5	17.7	6.6	21.2	30.7	6.7	13.6	9.5	18.9	C2L
QPSO10	14.3	2.8	14.3	3.4	12.2	5.6	5.7	4.3	12.4	3.5	9.3	4.4	14.3	6.5	16.0	26.8	8.0	16.5	26.0	33.1	
RIGA	13.9	2.7	14.3	3.5	15.8	5.5	25.7	6.0	18.0	4.0	19.9	5.0	25.2	7.1	35.0	35.6	22.1	31.2	6.8	13.7	
$\mathcal{H}(t)$	**89.1**	**2.7**	**86.9**	**3.9**	**82.6**	**10.8**	**65.0**	**4.8**	**83.7**	**4.8**	**77.2**	**7.0**	**65.2**	**4.9**	**18.1**	**5.1**	**18.1**	**5.0**	**18.3**	**5.3**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*82.7*	*11.3*	*76.2*	*6.9*	*84.7*	*7.0*	*83.2*	*7.2*	*74.8*	*6.8*	*8.6*	*23.3*	*40.2*	*35.7*	*1.0*	*6.9*	
APSO	14.9	3.1	14.3	3.6	12.8	5.1	9.8	5.8	13.5	3.7	11.5	3.9	15.8	6.8	8.6	17.5	8.4	17.1	16.5	26.9	
CPSO	14.3	2.9	14.3	3.5	12.6	4.7	9.3	5.4	10.7	3.4	10.5	3.6	15.1	6.5	7.8	16.0	8.8	17.8	17.0	27.4	
DE_H	14.1	2.6	14.3	3.5	16.1	5.4	20.3	7.2	15.9	3.9	18.2	4.6	6.1	3.1	4.8	7.5	26.1	33.2	14.1	24.7	
DE_L	13.7	2.8	14.3	3.5	15.9	5.2	16.7	5.8	15.4	3.8	15.9	4.6	5.3	2.5	4.7	7.0	26.1	33.2	14.8	25.2	
QPSO1	13.7	2.7	14.2	3.5	14.4	5.5	12.0	5.7	15.4	3.8	14.6	4.4	17.6	6.6	18.7	29.0	7.1	14.5	12.3	22.7	C2R
QPSO10	15.0	3.0	14.3	3.5	11.9	6.2	5.9	4.5	12.4	3.6	10.3	4.2	14.1	6.5	17.2	27.8	8.8	17.8	18.1	28.2	
RIGA	14.2	2.7	14.3	3.5	16.3	5.9	26.0	5.4	16.6	4.0	19.1	4.7	25.9	6.0	38.1	35.9	14.8	25.7	7.2	14.3	
$\mathcal{H}(t)$	**89.2**	**3.2**	**86.9**	**4.0**	**80.9**	**11.9**	**65.0**	**4.9**	**84.1**	**4.7**	**79.4**	**6.8**	**65.2**	**5.0**	**18.1**	**5.0**	**18.1**	**5.0**	**18.9**	**5.9**	
ΔE	*0.0*	*0.0*	*85.2*	*7.0*	*82.0*	*11.6*	*76.4*	*6.9*	*84.8*	*7.0*	*83.7*	*7.2*	*74.9*	*6.8*	*9.4*	*24.2*	*38.1*	*35.9*	*3.8*	*13.1*	
APSO	14.6	3.3	14.3	3.5	13.8	4.5	11.6	7.0	13.6	3.7	12.2	3.9	14.5	6.3	12.5	23.2	13.9	24.7	24.0	31.9	
CPSO	13.2	2.6	14.3	3.5	13.5	4.8	10.9	6.8	12.2	3.6	11.8	3.9	13.7	6.0	5.0	8.4	9.3	18.7	23.7	31.7	
DE_H	13.8	2.8	14.3	3.5	15.2	4.7	17.6	8.1	15.3	3.8	16.0	4.4	6.2	3.6	4.2	3.6	16.1	26.9	6.3	12.3	
DE_L	14.5	3.0	14.3	3.5	15.1	5.8	13.1	7.0	15.0	3.7	13.7	4.3	6.2	3.6	4.2	3.5	30.7	34.7	7.2	14.4	
QPSO1	14.8	2.8	14.3	3.5	14.9	4.9	19.4	8.2	16.1	3.9	16.7	4.8	20.7	5.7	32.7	35.2	7.0	14.3	11.9	22.2	C3C
QPSO10	14.6	2.4	14.3	3.5	12.2	5.2	6.7	5.4	12.3	3.7	10.9	4.5	12.1	6.1	9.6	19.1	7.6	15.6	21.1	30.3	
RIGA	14.4	2.6	14.3	3.5	15.4	4.9	20.7	8.2	15.5	3.9	18.7	4.9	26.6	4.8	31.8	35.0	15.4	26.2	5.9	11.1	
$\mathcal{H}(t)$	**89.5**	**3.7**	**86.9**	**3.9**	**83.0**	**9.8**	**64.9**	**4.9**	**85.0**	**4.6**	**81.0**	**7.3**	**65.2**	**5.0**	**18.1**	**5.0**	**18.1**	**5.0**	**18.7**	**5.7**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*83.3*	*10.2*	*77.5*	*7.0*	*85.0*	*7.0*	*83.9*	*7.4*	*74.8*	*6.8*	*3.1*	*14.5*	*16.7*	*30.4*	*3.0*	*11.7*	
APSO	14.4	2.8	14.3	3.5	14.1	5.0	13.4	7.7	14.2	3.7	13.0	3.9	14.5	6.0	11.9	22.4	11.6	22.1	26.3	33.1	
CPSO	13.7	2.8	14.3	3.5	13.7	5.1	12.6	7.7	13.6	3.6	12.6	3.7	14.2	5.9	8.4	17.2	12.0	22.6	24.9	32.5	
DE_H	14.8	3.4	14.3	3.5	15.1	5.8	17.1	7.8	14.7	3.7	14.9	4.2	5.8	3.0	4.3	3.9	13.5	24.3	5.5	10.0	
DE_L	14.1	2.5	14.3	3.5	14.4	5.9	11.3	6.6	14.3	3.7	12.1	3.8	6.1	3.4	4.2	3.7	18.9	29.1	5.3	9.3	
QPSO1	14.3	3.7	14.3	3.5	14.7	5.8	20.2	8.6	15.5	3.8	17.3	4.8	21.3	5.3	39.7	36.0	8.8	17.9	9.2	18.5	C3L
QPSO10	14.0	3.2	14.3	3.5	12.7	5.0	7.4	5.9	12.8	3.7	11.6	4.5	11.5	5.8	12.1	22.7	9.3	18.8	21.7	30.7	
RIGA	14.7	2.9	14.3	3.5	15.3	5.7	18.0	8.8	14.9	3.7	18.5	4.8	26.5	4.9	19.4	29.5	25.8	33.0	7.0	14.1	
$\mathcal{H}(t)$	**88.4**	**3.2**	**86.9**	**4.0**	**82.4**	**11.7**	**64.9**	**4.9**	**85.8**	**4.3**	**81.9**	**7.0**	**65.2**	**4.9**	**18.1**	**5.1**	**18.1**	**5.1**	**18.5**	**5.5**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*82.4*	*12.3*	*77.5*	*6.9*	*85.2*	*7.0*	*84.1*	*7.3*	*74.8*	*6.8*	*3.2*	*14.8*	*12.1*	*26.9*	*1.9*	*9.3*	
APSO	13.6	3.1	14.3	3.5	13.7	5.0	13.3	7.7	14.0	3.6	13.2	4.0	14.6	5.8	10.6	20.7	9.9	19.6	20.5	30.0	
CPSO	15.3	2.8	14.3	3.5	13.6	4.8	12.3	7.7	13.6	3.6	12.5	3.9	14.4	5.9	10.7	20.8	9.1	18.4	17.7	28.0	
DE_H	14.3	2.8	14.3	3.4	15.3	5.7	17.2	7.4	14.8	3.7	15.0	4.1	5.8	3.2	4.2	3.3	24.8	32.6	6.6	13.3	
DE_L	14.1	2.6	14.3	3.5	14.6	5.2	11.9	6.7	14.3	3.6	12.3	3.9	6.0	3.3	4.2	3.6	26.1	33.2	7.3	14.7	
QPSO1	13.4	3.0	14.3	3.5	14.7	5.5	20.2	9.0	15.5	3.9	17.2	4.6	21.1	5.5	26.0	33.1	7.5	15.4	18.8	28.8	C3R
QPSO10	14.6	2.3	14.3	3.5	12.3	5.2	6.7	5.5	12.8	3.7	11.5	4.4	11.4	5.8	11.2	21.5	6.8	13.8	21.8	30.7	
RIGA	14.7	3.2	14.3	3.5	15.7	6.2	18.4	8.6	14.9	3.8	18.4	4.8	26.7	4.7	33.1	35.3	15.8	26.6	7.3	14.6	
$\mathcal{H}(t)$	**89.3**	**2.7**	**87.0**	**4.0**	**81.9**	**11.2**	**64.9**	**4.9**	**85.9**	**4.3**	**82.0**	**6.7**	**65.1**	**4.9**	**18.1**	**5.0**	**18.1**	**5.0**	**18.7**	**5.7**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*82.3*	*11.3*	*77.5*	*7.0*	*85.1*	*7.0*	*84.2*	*7.3*	*74.8*	*6.8*	*4.7*	*17.7*	*28.4*	*35.2*	*2.9*	*11.7*	

Table 10: Mean and standard deviations of $\mathcal{H}(t)$ and entity to heuristic allocations (all progressive environments).

Alg	HH_Fix		HH_Rand		HH_Roul		HH_Tour		HH_ARank		HH_AProp		HH_Freq		HH_ReinfFr		HH_ReinfPr		HH_DiffPr		Env
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	
APSO	14.5	2.8	14.3	3.5	13.3	8.4	16.8	7.8	16.0	4.0	15.5	4.5	14.6	5.9	10.8	21.0	20.0	29.9	25.2	32.7	
CPSO	14.4	3.1	14.3	3.5	12.6	6.2	14.8	7.8	15.1	4.0	14.5	4.5	14.0	5.8	8.2	16.8	17.7	28.2	17.0	27.5	
DE_H	15.4	3.5	14.3	3.5	15.7	8.4	14.9	7.5	13.6	3.7	14.1	4.3	5.7	2.9	4.2	2.9	9.7	19.4	6.1	11.8	
DE_L	14.4	3.0	14.3	3.5	16.6	10.6	9.7	6.0	10.8	3.3	10.8	3.8	5.7	3.0	4.1	2.7	11.8	22.3	10.3	20.2	
QPSO1	13.7	3.1	14.3	3.5	13.5	6.4	21.9	8.7	18.7	4.3	19.8	5.0	21.5	5.5	26.1	33.2	15.3	26.1	9.7	19.3	P1C
QPSO10	14.0	3.2	14.3	3.5	13.4	5.8	8.9	6.9	12.8	3.7	12.0	4.6	12.2	5.7	7.5	15.4	18.3	28.7	27.6	33.7	
RIGA	13.6	2.9	14.3	3.5	14.9	6.7	12.9	8.2	12.9	4.2	13.3	5.1	26.3	4.9	39.1	35.9	7.2	14.7	4.3	3.8	
$\mathcal{H}(t)$	**88.1**	**3.2**	**86.9**	**4.0**	**76.6**	**17.7**	**65.3**	**5.7**	**82.9**	**5.0**	**80.5**	**6.9**	**65.4**	**5.3**	**18.1**	**5.0**	**18.1**	**5.0**	**18.2**	**5.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*76.5*	*18.9*	*77.4*	*7.0*	*84.6*	*7.0*	*83.9*	*7.3*	*75.0*	*6.9*	*5.3*	*18.7*	*11.2*	*26.0*	*0.8*	*6.1*	
APSO	14.4	3.0	14.3	3.5	13.8	6.3	17.4	8.6	16.9	4.1	16.9	4.7	14.7	5.9	12.2	22.8	24.8	32.6	14.3	25.1	
CPSO	14.4	3.1	14.3	3.5	14.2	6.0	17.3	8.1	16.5	4.1	16.6	4.6	14.3	6.0	13.5	24.3	18.7	28.9	20.3	30.1	
DE_H	14.4	3.3	14.3	3.5	15.5	6.2	13.2	7.4	13.3	3.7	12.9	4.0	5.3	2.5	4.2	3.7	7.9	16.3	6.8	13.7	
DE_L	14.5	3.0	14.3	3.5	15.0	5.6	8.2	5.0	9.2	3.0	8.4	3.2	6.3	3.5	4.2	3.6	7.9	16.3	6.3	12.5	
QPSO1	13.8	2.9	14.2	3.5	12.9	6.1	21.9	8.1	18.5	4.2	19.9	5.2	21.2	5.9	21.7	31.0	12.4	23.0	12.5	23.1	P1L
QPSO10	14.7	2.8	14.3	3.4	14.1	5.8	11.1	7.2	13.5	3.8	13.3	4.5	11.9	5.8	18.3	28.7	20.0	29.9	35.1	35.6	
RIGA	13.8	3.1	14.3	3.5	14.5	5.5	10.9	7.4	12.0	3.6	11.9	4.9	26.3	5.0	25.8	33.0	8.3	17.0	4.6	6.4	
$\mathcal{H}(t)$	**88.8**	**3.3**	**87.0**	**3.9**	**80.4**	**13.3**	**65.1**	**4.9**	**81.7**	**5.0**	**78.5**	**6.8**	**65.1**	**5.0**	**18.1**	**5.0**	**18.1**	**5.0**	**18.2**	**5.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*80.5*	*12.3*	*76.6*	*6.9*	*84.4*	*7.0*	*83.3*	*7.3*	*74.8*	*6.8*	*3.4*	*15.2*	*7.7*	*22.2*	*0.7*	*5.9*	
APSO	14.2	3.1	14.3	3.5	12.7	6.7	11.9	7.5	13.8	3.7	13.0	4.0	14.8	5.9	7.0	14.2	13.5	24.3	15.8	26.2	
CPSO	14.5	2.8	14.3	3.5	12.1	6.1	11.2	7.2	12.9	3.6	12.3	3.7	13.3	6.1	5.6	10.4	11.5	21.9	12.0	22.2	
DE_H	14.5	3.0	14.3	3.5	17.7	9.4	17.1	7.3	15.0	3.8	15.6	4.3	6.0	3.3	4.2	3.1	20.5	30.2	15.4	25.8	
DE_L	14.3	3.0	14.3	3.5	16.1	7.3	13.1	6.6	13.4	3.6	13.1	4.0	6.6	4.0	4.1	2.3	18.2	28.6	13.1	23.4	
QPSO1	14.0	2.4	14.3	3.5	12.4	6.4	21.2	8.1	17.8	4.1	18.3	4.6	21.1	6.0	27.7	33.8	11.4	21.8	28.9	33.9	P1R
QPSO10	14.3	3.2	14.2	3.5	11.4	6.1	6.2	5.0	10.7	3.4	10.0	4.1	11.8	5.9	9.9	19.7	14.2	25.0	9.1	18.1	
RIGA	14.2	2.7	14.3	3.5	17.6	8.4	19.3	8.3	16.4	4.0	17.6	4.8	26.4	4.9	41.5	35.9	10.7	20.9	5.8	11.0	
$\mathcal{H}(t)$	**89.1**	**3.0**	**86.9**	**4.0**	**74.9**	**18.2**	**65.0**	**5.0**	**83.7**	**4.7**	**81.2**	**6.4**	**65.3**	**5.2**	**18.1**	**5.0**	**18.1**	**5.0**	**18.8**	**5.9**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*75.9*	*17.1*	*77.6*	*6.9*	*84.7*	*7.0*	*84.1*	*7.2*	*75.0*	*6.8*	*4.4*	*17.1*	*17.2*	*30.7*	*2.1*	*8.4*	
APSO	15.2	3.3	14.3	3.5	13.8	6.9	10.9	7.2	13.1	3.6	12.2	3.8	15.6	6.5	10.3	20.3	15.0	25.8	13.4	23.7	
CPSO	14.3	2.9	14.3	3.5	11.6	5.7	9.4	6.4	12.3	3.5	11.3	3.7	14.6	6.3	7.2	14.8	14.0	24.8	16.9	27.2	
DE_H	13.7	2.8	14.3	3.5	15.8	6.9	17.1	6.1	15.8	3.9	16.1	4.3	5.9	3.0	4.4	5.3	13.2	23.9	10.6	20.0	
DE_L	13.6	3.5	14.3	3.5	16.5	6.9	12.7	5.7	14.5	3.8	13.9	4.2	5.4	2.7	4.4	5.2	15.3	26.2	7.6	14.9	
QPSO1	14.5	2.8	14.3	3.5	12.7	6.7	16.7	7.1	15.6	3.9	15.9	4.3	17.3	6.8	21.0	30.5	11.8	22.3	20.5	29.8	P2C
QPSO10	14.1	3.1	14.3	3.5	12.9	7.8	6.6	5.5	11.1	3.5	10.9	4.3	14.6	6.6	13.0	23.8	16.7	27.4	19.4	29.1	
RIGA	14.6	3.0	14.2	3.5	16.7	7.0	26.4	5.5	17.5	4.1	19.7	5.0	26.5	5.2	39.6	36.0	14.1	25.0	11.6	21.8	
$\mathcal{H}(t)$	**88.0**	**3.4**	**86.9**	**4.0**	**76.9**	**16.2**	**64.9**	**4.9**	**83.9**	**4.8**	**81.0**	**6.8**	**65.2**	**5.0**	**18.1**	**5.0**	**18.1**	**5.0**	**19.2**	**6.1**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*77.6*	*15.5*	*77.1*	*6.9*	*84.7*	*7.0*	*84.0*	*7.2*	*74.8*	*6.8*	*4.6*	*17.6*	*40.4*	*35.7*	*4.3*	*13.3*	
APSO	14.5	3.2	14.2	3.5	13.7	7.7	11.0	7.2	14.1	3.7	11.7	3.7	15.4	6.7	9.9	19.7	13.5	24.3	23.7	32.0	
CPSO	14.2	3.5	14.3	3.5	12.3	5.9	9.9	6.5	12.0	3.4	10.9	3.6	14.9	6.5	6.9	14.1	10.4	20.4	21.2	30.6	
DE_H	14.1	3.0	14.3	3.5	15.3	6.2	16.9	6.1	13.2	3.7	16.6	4.4	5.9	2.9	4.5	5.9	13.6	24.4	11.2	21.5	
DE_L	14.1	2.3	14.3	3.5	16.2	8.0	11.9	5.9	12.0	3.4	13.9	4.2	5.7	2.9	4.5	5.6	12.8	23.5	6.9	13.8	
QPSO1	14.2	2.3	14.4	3.5	12.6	7.0	17.2	6.6	16.1	3.8	15.7	4.4	17.3	6.7	22.0	31.1	9.4	19.0	7.1	14.5	P2L
QPSO10	14.7	3.1	14.3	3.5	12.4	5.7	6.3	5.1	13.1	3.6	9.8	4.1	14.1	6.5	13.6	24.4	12.7	23.4	21.1	30.5	
RIGA	14.3	3.4	14.3	3.5	17.5	10.1	26.9	4.8	19.6	4.2	21.5	4.9	26.6	5.1	38.7	35.9	27.6	33.8	8.8	17.7	
$\mathcal{H}(t)$	**89.1**	**3.8**	**87.0**	**3.9**	**77.1**	**17.8**	**64.9**	**5.0**	**83.3**	**4.8**	**79.0**	**6.5**	**65.2**	**4.9**	**18.1**	**5.1**	**18.1**	**5.0**	**18.2**	**5.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*77.0*	*18.7*	*77.1*	*6.9*	*84.6*	*7.0*	*83.6*	*7.1*	*74.9*	*6.8*	*6.4*	*20.5*	*29.8*	*35.4*	*0.7*	*5.8*	
APSO	14.8	2.8	14.4	3.5	12.0	6.7	11.1	7.3	13.2	3.6	12.2	3.8	15.6	6.8	8.5	17.3	16.9	27.6	18.0	28.2	
CPSO	13.8	2.6	14.3	3.5	10.9	7.1	9.8	6.5	12.0	3.5	11.2	3.7	14.2	6.2	11.0	21.3	10.7	20.9	11.9	22.2	
DE_H	13.9	2.7	14.3	3.5	16.5	10.8	16.9	6.3	15.1	3.8	16.3	4.4	6.0	3.1	4.3	4.7	14.3	25.2	9.6	18.6	
DE_L	14.0	2.8	14.3	3.4	18.7	11.2	12.7	5.7	14.1	3.7	14.2	4.2	5.5	2.8	4.6	6.3	14.2	25.1	10.5	20.0	
QPSO1	14.6	2.7	14.2	3.5	12.6	9.4	17.3	6.8	15.9	3.9	15.7	4.3	17.9	6.4	26.4	33.3	11.4	21.8	21.6	30.5	P2R
QPSO10	14.5	2.6	14.3	3.5	11.9	6.7	6.2	5.1	11.5	3.6	10.3	4.1	14.3	6.6	12.0	22.5	16.6	27.3	21.8	30.8	
RIGA	14.5	2.7	14.3	3.5	17.3	8.7	26.0	6.4	18.1	4.1	20.0	4.9	26.5	5.1	33.2	35.3	15.9	26.7	6.7	13.4	
$\mathcal{H}(t)$	**89.6**	**2.6**	**87.0**	**4.0**	**70.8**	**20.9**	**64.8**	**4.9**	**83.8**	**4.8**	**80.6**	**6.7**	**65.2**	**4.9**	**18.1**	**5.0**	**18.1**	**5.1**	**18.8**	**5.9**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*71.9*	*21.6*	*77.0*	*6.9*	*84.7*	*7.0*	*83.9*	*7.2*	*74.8*	*6.8*	*6.5*	*20.6*	*24.7*	*34.1*	*2.8*	*10.9*	
APSO	14.8	2.9	14.3	3.5	13.9	4.8	16.2	8.1	15.6	3.9	15.4	4.4	14.9	5.8	10.7	20.9	22.6	31.5	24.6	32.3	
CPSO	14.5	3.4	14.3	3.5	13.6	5.5	15.0	7.8	14.8	3.9	14.4	4.3	13.9	5.6	8.0	16.4	17.4	28.0	20.4	30.0	
DE_H	14.2	3.1	14.3	3.5	15.0	5.1	14.9	7.4	14.3	3.7	14.2	4.1	5.8	2.9	4.2	2.8	11.1	21.4	7.0	14.0	
DE_L	14.9	2.9	14.2	3.5	14.7	5.7	9.5	5.8	11.6	3.5	11.0	3.8	5.7	3.1	4.2	3.3	11.7	22.2	6.7	13.5	
QPSO1	14.0	3.3	14.3	3.6	13.9	5.5	23.5	7.4	18.0	4.2	18.9	4.8	21.1	6.0	21.5	30.8	14.1	24.9	12.1	22.6	P3C
QPSO10	13.3	2.7	14.3	3.5	13.9	6.2	8.9	6.8	12.1	3.7	12.1	4.4	12.1	5.9	12.3	23.0	15.5	26.3	24.3	32.2	
RIGA	14.3	2.9	14.2	3.5	15.0	5.4	12.0	8.2	13.6	3.9	14.0	5.1	26.5	4.9	39.1	35.9	7.6	15.6	4.9	7.8	
$\mathcal{H}(t)$	**88.6**	**3.2**	**86.9**	**3.9**	**82.4**	**12.0**	**65.0**	**4.8**	**83.8**	**4.8**	**81.6**	**6.6**	**65.1**	**4.9**	**18.1**	**5.0**	**18.1**	**5.0**	**18.4**	**5.4**	
ΔE	*0.0*	*0.0*	*85.2*	*7.0*	*81.9*	*12.2*	*77.1*	*6.9*	*84.8*	*7.0*	*84.2*	*7.2*	*74.8*	*6.8*	*5.7*	*19.3*	*13.6*	*28.1*	*1.4*	*7.9*	
APSO	14.4	3.1	14.2	3.5	13.8	7.5	18.2	8.3	16.4	4.1	16.6	4.6	15.0	6.0	10.0	19.9	21.8	31.0	21.5	30.7	
CPSO	14.0	3.3	14.4	3.5	14.0	5.7	17.7	7.9	16.1	3.9	16.0	4.6	14.3	5.8	9.8	19.5	16.8	27.5	22.5	31.3	
DE_H	14.2	3.1	14.3	3.5	15.1	6.3	13.7	7.2	13.4	3.6	13.4	4.2	5.3	2.7	4.3	4.4	9.3	18.8	6.8	13.8	
DE_L	14.0	3.3	14.3	3.5	14.4	6.0	7.8	4.9	10.1	3.3	8.8	3.4	6.1	3.2	4.3	4.4	10.2	20.2	5.1	8.5	
QPSO1	14.7	2.9	14.3	3.5	14.3	6.0	20.3	9.3	17.7	4.2	19.3	4.9	21.2	5.7	19.6	29.6	12.9	23.6	9.3	18.7	P3L
QPSO10	14.2	2.7	14.3	3.5	13.5	5.7	10.7	7.1	13.4	3.7	13.6	4.5	11.7	5.7	17.8	28.2	20.2	30.0	29.5	34.2	
RIGA	14.6	2.9	14.3	3.5	15.0	6.2	11.5	7.6	12.9	3.8	12.3	4.6	26.2	5.0	34.3	35.5	8.6	17.6	5.3	9.0	
$\mathcal{H}(t)$	**88.3**	**3.2**	**86.9**	**4.0**	**80.7**	**14.0**	**65.0**	**5.0**	**83.1**	**5.0**	**79.8**	**6.9**	**65.2**	**4.9**	**18.1**	**5.1**	**18.1**	**5.0**	**18.4**	**5.4**	
ΔE	*0.0*	*0.0*	*85.2*	*7.0*	*80.5*	*14.3*	*76.6*	*6.9*	*84.6*	*7.0*	*83.7*	*7.3*	*74.8*	*6.8*	*4.7*	*17.7*	*9.3*	*24.1*	*1.4*	*7.8*	
APSO	13.7	3.2	14.3	3.5	12.8	6.3	11.5	7.5	14.0	3.7	13.3	4.0	14.8	6.0	7.4	15.1	13.3	24.1	18.1	28.0	
CPSO	14.5	2.8	14.2	3.5	12.0	5.8	11.0	7.1	13.3	3.6	12.7	3.8	13.7	5.9	4.4	4.8	11.5	21.9	12.4	22.3	
DE_H	14.3	2.7	14.3	3.5	16.6	7.5	17.4	7.2	15.1	3.8	15.6	4.3	5.9	3.1	4.1	2.4	21.0	30.5	14.4	24.8	
DE_L	13.7	2.7	14.3	3.4	17.8	11.4	13.0	6.5	13.9	3.7	13.3	4.1	6.3	3.7	4.1	2.2	16.2	27.0	13.9	24.1	
QPSO1	15.0	2.6	14.3	3.5	12.9	6.3	21.0	8.2	16.9	4.0	17.5	4.6	21.1	5.6	29.0	34.2	11.9	22.4	24.0	31.8	P3R
QPSO10	14.5	2.9	14.3	3.5	11.9	6.3	6.4	5.3	10.9	3.5	10.5	4.2	11.8	6.1	7.7	15.8	15.2	26.0	12.0	22.1	
RIGA	14.3	2.6	14.3	3.5	16.0	7.9	19.7	8.1	16.0	3.9	17.2	4.7	26.4	5.1	43.3	35.8	10.9	21.1	5.2	8.9	
$\mathcal{H}(t)$	**89.4**	**3.1**	**87.0**	**4.0**	**76.0**	**18.4**	**65.0**	**5.3**	**84.5**	**4.7**	**82.0**	**6.7**	**65.1**	**4.9**	**18.1**	**5.0**	**18.1**	**5.0**	**19.3**	**6.2**	
ΔE	*0.0*	*0.0*	*85.3*	*7.0*	*76.1*	*19.0*	*77.5*	*7.0*	*84.9*	*7.0*	*84.2*	*7.2*	*74.9*	*6.8*	*5.7*	*19.3*	*20.0*	*32.2*	*3.6*	*11.3*	

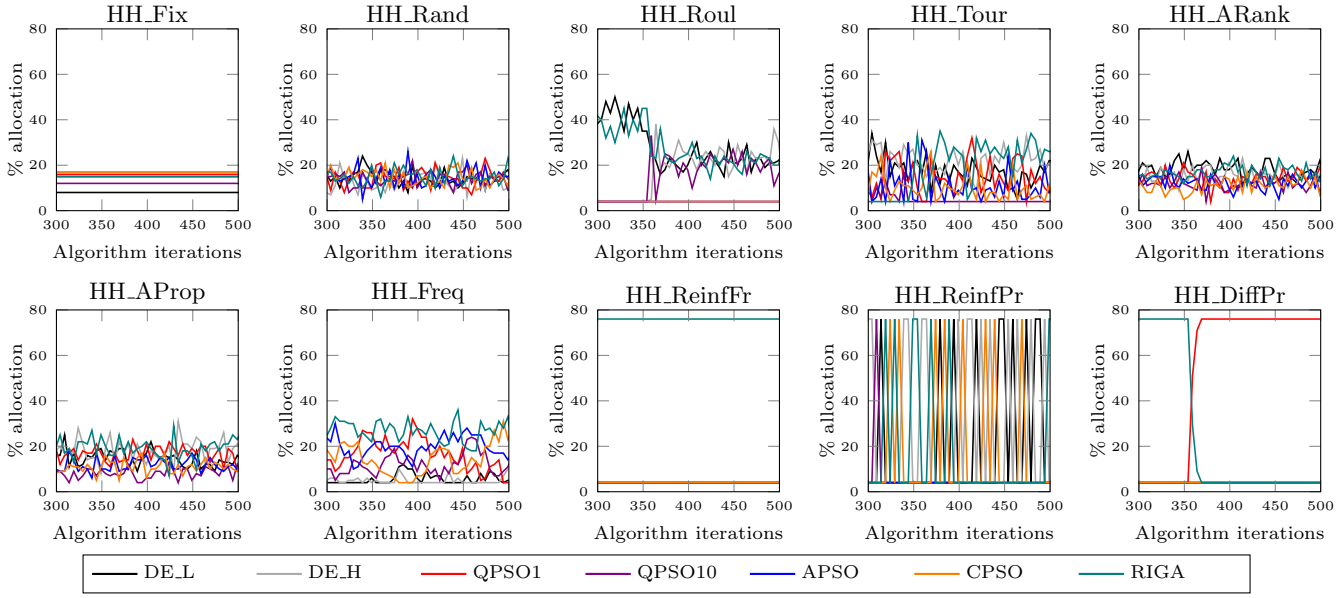

Figure 2: Example entity allocations per selection operator (median sample of the (A3R) environment for iteration $t \in [300, 500]$).

approaches used *pheromone* as a memory of good assignment probabilities. Neither of the ant-based approaches managed to perform better than **HH_Fix**. Pheromone update magnitudes (see equations (10) and (12)) depended on the number of assigned entities per heuristic, while the pheromone evaporation rate intensity (see equation (11)) did not vary. The original evaporation rate proposed by Nepomuceno and Engelbrecht [58] may have been too strong, causing both ant-based approaches to degenerate to uniform random probabilities too quickly. The ant-based approaches could potentially yield better performance if pheromone update and/or evaporation strategies were made adaptive.

In contrast, **HH_Roul** consistently had lower mean $\mathcal{H}(t)$ and ΔE values with much higher standard deviations compared to the other roulette wheel-based methods. The large deviations show that **HH_Roul** oscillated between rapidly allocating the bulk of entities to a few heuristics versus spreading entities out evenly across heuristics. Figure 2 illustrates this behavior well.

HH_Tour and **HH_Freq** both used tournament selection with different types of feedback, and together with **HH_DiffPr** were the only selection operators that improved performance beyond fixed allocation. **HH_Tour** and **HH_Freq** both showed similar rank performance in each environment, similar $\mathcal{H}(t)$ and ΔE values, yet starkly different entity-to-heuristic allocation behavior. Both **HH_Tour** and **HH_Freq** consistently maintained lower heuristic space diversity than the roulette wheel-based approaches, allocating most entities to fewer heuristics for longer periods. The trend can be seen in figure 2 where **HH_Tour** and **HH_Freq** regularly reached higher percentages of entity allocations to a single heuristic, and often hit the minimum heuristic allocation of $n_m \geq 4$, while the roulette-based approaches rarely did. The plot for **HH_Freq** is noticeably less jittery than **HH_Tour**, illustrating that **HH_Freq** generally had lower deviation values for entity-to-heuristic allocations as shown in tables 8, 9 and 10.

HH_Tour rewarded higher mean entity fitness, while **HH_Freq** favored a higher number of improving moves. **HH_Tour** and **HH_Freq** favored RIGA heavily. Per environment, **HH_Tour** showed more fluctuating entity allocations to RIGA than **HH_Freq**, indicated by **HH_Tour**'s higher and more varied σ values for entity allocations to RIGA. In contrast, **HH_Freq** had nearly identical, sustained high entity allocations to RIGA in all environments. This makes sense since newly replaced individuals in RIGA would initially show many improving moves[7], and cause **HH_Freq** to continually assign many entities to RIGA (regardless of the environment type). Together with high ΔE values, the results suggest that **HH_Freq** seemed to favor using RIGA as a 'diversity pool' to continually redistribute diversified entities to the other heuristics.

Tables 8, 9 and 10 show how **HH_Tour** utilized DE_L and DE_H more than **HH_Freq** did, and confirms that the DE heuristics generally showed high convergence behavior: high μ and low σ values for entity allocation to DE_L and DE_H by **HH_Tour** (which rewards high mean fitness) shows that both DE variants often achieved high mean fitness, while low μ and low σ values for entity allocation to DE_L and DE_H by **HH_Freq** (which favors a high number of improvements) shows that both DE variants made few improving moves.

[7]Newly replaced individuals are initialized randomly and generally start to converge to the best RIGA individual.

Note the difference between **HH_Freq** and **HH_ReinfFr** in tables 8, 9 and 10 and in figure 2[8]. **HH_Freq** constantly varied the entity balance between heuristics. In contrast, **HH_ReinfFr** showed infrequent but large entity reassignments, and tended to assign entities to a single heuristic. Table 4 confirms that **HH_ReinfFr** had the lowest deviation in rank values ($\sigma = 2.13$) of all the selection operators or any of the control groups, making **HH_ReinfFr** the most consistent and stable method overall regardless of environment.

HH_DiffPr rapidly assigned all entities to a single heuristic (indicated by low $\mathcal{H}(t)$ values), did not reassign entities often (indicated by low ΔE values), and was able to rapidly reallocate entities to another best suited heuristic at time t (indicated by highly varied entity-to-heuristic allocation values). **HH_ReinfPr** often oscillated between extremes by frequently allocating most entities to one and then the other heuristic, as indicated by the large standard deviation for ΔE values.

6. Conclusions

This study highlighted a challenge faced by meta-heuristic practitioners when solving DOPs, where neither static parameter tuning nor ad hoc combinations of self-adaptive parameter update strategies yield increased results. The HMHH hyper-heuristic framework addresses this challenge by balancing computational resources across heuristics with known, yet diverse, behaviors. The performance of the HMHH hyper-heuristic framework depends greatly on the managed heuristics being complimentary to each other. This study presented the considerations and criteria to select a diverse mix of DOP- specific and non-DOP- specific heuristics to enable HMHH to effectively solve DOPs. These criteria describe how heuristics should 1) utilize low-level operators that apply different types of problem space search strategies, and 2) balance *exploration* and *exploitation* in different ways. This study used a blend of SI and EC approaches that rely on either trajectory or heredity information. Through specific parameter values, two variations of most heuristics that favor either *exploration* or *exploitation* were used.

Empirical investigation across the full spectrum of different DOP types show that many intelligent HMHH selection operators can consistently achieve better performance with lower variance compared to running any heuristics in isolation. If the type of dynamism of the problem is unknown, the results show that using any of the investigated hyper-heuristics yields more stable performance compared to the hit-or-miss nature of either stand-alone or speciated heuristics. There seems to be no clear correlation between heuristic space convergence and HMHH performance in DOPs, as there is in static environments [28]. Some well-performing hyper-heuristics such as **HH_DiffPr** and **HH_ReinFr** showed high convergence to a single heuristic coupled with the ability to rapidly increase heuristic space diversity when needed. Other well-performing hyper-heuristics such as **HH_Freq** and **HH_Tour** showed no heuristic convergence behavior, and always maintained high heuristic space diversity. Many intelligent HMHH selection operators could readily exploit heuristics that, on their own, perform very poorly, but whose behavior is well-suited at time t. HMHH offers an attractive immediate "off the peg" solution to a DOP while practitioners spend time on developing more tailored approaches.

A complex time-dependent relationship exists between overall performance, heuristic space diversity, entity allocation stability, entity trajectories through the search space, problem landscape characteristics, environment change dynamics, performance feedback measures, the hyper-heuristic learning scheme, and the nature of the low-level heuristics. Future studies should determine the exact factors that make up a complimentary set of "black box" heuristics. Focus should also be given to understanding the effect of entity reallocations on a heuristic's efficacy and individual entity behavior. Open questions include: What should each heuristic's minimum and maximum allowed population size be? How are the *stability*, *robustness*, *satisficability*, and *convergence speed* [5] of different heuristics impacted by the hyper-heuristic reallocating entities between intelligent heuristics? Are these measures correlated with published population size scalability studies of the heuristics? Can these measures help craft guidelines to guide practitioners on how to select suitable heuristics for a hyper-heuristic pool? How do the answers to any of the aforementioned open questions differ across various types of DOPs? Future research should also investigate the inclusion of deterministic methods and hybrid exact methods (so-called *matheuristics* [41]) together with multi-population heuristics in the heuristic pool.

[8] Both selection operators used the same type of feedback, yet **HH_Freq** did not utilize memory like **HH_ReinfFr** did. **HH_Freq** also used *probabilistic* tournament selection while **HH_ReinfFr** used *deterministic* winner-takes-all rank-based selection.

References

[1] J. Duhain, A. Engelbrecht, Towards a more complete classification system for dynamically changing environments, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2012, pp. 1–8.

[2] M. C. Du Plessis, Adaptive multi-population differential evolution for dynamic environments, Ph.D. thesis (2012).

[3] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments – a survey, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 303–317.

[4] C. Cruz, J. R. González, D. A. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, Soft Computing 15 (7) (2011) 1427–1448.

[5] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (2012) 1–24.

[6] M. Mavrovouniotis, C. Li, S. Yang, A survey of swarm intelligence for dynamic optimization: algorithms and applications, Swarm and Evolutionary Computation 33 (2017) 1–17.

[7] P. Cowling, E. Soubeiga, Neighborhood structures for personnel scheduling: a summit meeting scheduling problem, in: Proceed-

ings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, by EK Burke, W. Erben, 2000, p. 277.

[8] H. Fisher, G. L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules, Factory Scheduling Conference (Carnegie Institute of Technology).

[9] 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, Journal of the Operational Research Society 64 (12) (2013) 1695–1724.

[10] J. Maturana, F. Lardeux, F. Saubion, Autonomous operator management for evolutionary algorithms, Journal of Heuristics 16 (6) (2010) 881–909.

[11] A. E. Eiben, Z. Michalewicz, M. Schoenauer, J. E. Smith, Parameter control in evolutionary algorithms, in: Parameter setting in evolutionary algorithms, Springer, 2007, pp. 19–46.

[12] F. Lobo, C. F. Lima, Z. Michalewicz, Parameter setting in evolutionary algorithms, Vol. 54, Springer Science & Business Media, 2007.

[13] G. Karafotias, M. Hoogendoorn, A. E. Eiben, Parameter control in evolutionary algorithms: Trends and challenges., IEEE Trans. Evolutionary Computation 19 (2) (2015) 167–187.

[14] N. Krasnogor, J. Smith, A memetic algorithm with self-adaptive local search: Tsp as a case study, in: Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2000, pp. 987–994.

[15] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 36 (1) (2006) 141–152.

[16] B. A. Huberman, R. M. Lukose, T. Hogg, An economics approach to hard computational problems, Science 275 (5296) (1997) 51–54.

[17] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Information Sciences 329 (2016) 329–345.

[18] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P. Suganthan, Ensemble of differential evolution variants, Information Sciences 423 (2018) 172–186.

[19] G. Uludağ, B. Kiraz, A. Ş. Etaner-Uyar, E. Özcan, A hybrid multi-population framework for dynamic environments combining online and offline learning, Soft Computing 17 (12) (2013) 2327–2348.

[20] B. Kiraz, A. Ş. Uyar, E. Özcan, An investigation of selection hyper-heuristics in dynamic environments, in: Applications of Evolutionary Computation, Springer, 2011, pp. 314–323.

[21] E. Ozcan, S. E. Uyar, E. Burke, A greedy hyper-heuristic in dynamic environments, in: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, ACM, 2009, pp. 2201–2204.

[22] B. Kiraz, A. Ş. Etaner-Uyar, E. Özcan, An ant-based selection hyper-heuristic for dynamic environments, in: Applications of Evolutionary Computation, Springer, 2013, pp. 626–635.

[23] H. R. Topcuoglu, A. Ucar, L. Altin, A hyper-heuristic based framework for dynamic optimization problems, Applied Soft Computing 19 (0) (2014) 236 – 251. doi:http://dx.doi.org/10.1016/j.asoc.2014.01.037.
URL http://www.sciencedirect.com/science/article/pii/S156849461400057X

[24] S. van der Stockt, A. P. Engelbrecht, Analysis of hyper-heuristic performance in different dynamic environments, in: Proceedings of the IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), IEEE, 2014, pp. 1–8.

[25] S. Van der Stockt, A. Engelbrecht, Analysis of global information sharing in hyper-heuristics for different dynamic environments, in: Proceedings of the IEEE Congress on Evolutionary Computation, Sendai, Japan, 2015.

[26] J. Grobler, A. P. Engelbrecht, G. Kendall, V. Yadavalli, Alternative hyper-heuristic strategies for multi-method global optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.

[27] J. Grobler, A. Engelbrecht, G. Kendall, V. Yadavalli, Multi-method algorithms: Investigating the entity-to-algorithm allocation problem, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 570–577.

[28] J. Grobler, A. P. Engelbrecht, G. Kendall, V. Yadavalli, Heuristic space diversity control for improved meta-hyper-heuristic performance, Information Sciences 300 (2015) 49–62.

[29] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Washington, DC, USA, 1999, pp. 1875–1882.

[30] I. Moser, R. Chiong, Dynamic function optimization: The moving peaks benchmark, in: Metaheuristics for Dynamic Optimization, Springer, 2013, pp. 35–59.

[31] R. C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of the IEEE Congress on Evolutionary Computation, Vol. 1, IEEE, 2001, pp. 94–100.

[32] P. J. Angeline, Tracking extrema in dynamic environments, in: Evolutionary Programming VI, Springer, 1997, pp. 335–345.

[33] B. Leonard, A. Engelbrecht, On the optimality of particle swarm parameters in dynamic environments, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2013, pp. 1564–1569.

[34] A. P. Engelbrect, Computational Intelligence: An Introduction, 2nd Edition, Wiley, 2007.

[35] X. Hu, R. Eberhart, Tracking dynamic systems with PSO: where's the cheese, in: Proceedings of the Workshop on Particle Swarm Optimization, 2001, pp. 80–83.

[36] M. Helbig, A. P. Engelbrecht, Issues with performance measures for dynamic multi-objective optimisation, in: Proceedings of the IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), IEEE, 2013, pp. 17–24.

[37] M. Helbig, A. P. Engelbrecht, Analysing the performance of dynamic multi-objective optimisation algorithms, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1531–1539.

[38] W. H. Kruskal, W. A. Wallis, Use of ranks in one-criterion variance analysis, Journal of the American Statistical Association 47 (260) (1952) 583–621.

[39] H. B. Mann, D. R. Whitney, et al., On a test of whether one of two random variables is stochastically larger than the other, The Annals of Mathematical Statistics 18 (1) (1947) 50–60.

[40] K. Sörensen, F. W. Glover, Metaheuristics, in: Encyclopedia of operations research and management science, Springer, 2013, pp. 960–970.

[41] K. Sörensen, Metaheuristicsthe metaphor exposed, International Transactions in Operational Research 22 (1) (2015) 3–18.

[42] R. Mendes, A. S. Mohais, DynDE: a differential evolution for dynamic optimization problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, Vol. 3, Ieee, 2005, pp. 2808–2815.

[43] A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, Vol. 2, IEEE, 2005, pp. 1785–1791.

[44] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, V. Zumer, Dynamic optimization using self-adaptive differential evolution., in: Proceedings of the IEEE Congress on Evolutionary Computation, 2009, pp. 415–422.

[45] I. Moser, T. Hendtlass, A simple and efficient multi-component algorithm for solving dynamic function optimisation problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 252–259.

[46] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, S. Yang, An adaptive multipopulation framework for locating and tracking multiple optima, IEEE Transactions on Evolutionary Computation 20 (4) (2016) 590–605.

[47] M. Mavrovouniotis, S. Yang, Population-based incremental learning with immigrants schemes in changing environments,

in: Computational Intelligence, 2015 IEEE Symposium Series on, IEEE, 2015, pp. 1444–1451.
[48] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE transactions on evolutionary computation 1 (1) (1997) 67–82.
[49] A. Auger, O. Teytaud, Continuous lunches are free!, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, 2007, pp. 916–922.
[50] A. Alabert, A. Berti, R. Caballero, M. Ferrante, No-free-lunch theorems in the continuum, Theoretical Computer Science 600 (2015) 98–106.
[51] R. Poli, M. Graff, There is a free lunch for hyper-heuristics, genetic programming and computer scientists, in: European Conference on Genetic Programming, Springer, 2009, pp. 195–207.
[52] K. Chakhlevitch, P. Cowling, Hyperheuristics: recent developments, in: Adaptive and Multilevel Metaheuristics, Springer, 2008, pp. 3–29.
[53] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, J.-J. Li, Distributed evolutionary algorithms and their models: A survey of the state-of-the-art, Applied Soft Computing 34 (2015) 286–300.
[54] E. K. Burke, G. Kendall, E. Soubeiga, A tabu-search hyperheuristic for timetabling and rostering, Journal of Heuristics 9 (6) (2003) 451–470.
[55] A. E. Eiben, S. K. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 19–31.
[56] C. Li, S. Yang, T. T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (3) (2012) 627–646.
[57] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Information Sciences 181 (20) (2011) 4515–4538.
[58] F. Nepomuceno, A. Engelbrecht, A self-adaptive heterogeneous PSO inspired by ants, in: M. Dorigo, M. Birattari, C. Blum, A. Christensen, A. Engelbrecht, R. Gro, T. Sttzle (Eds.), Swarm Intelligence, Vol. 7461 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 188–195.
[59] F. Nepomuceno, A. Engelbrecht, A self-adaptive heterogeneous PSO for real-parameter optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2013, pp. 361–368.
[60] F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, IEEE Transactions on evolutionary computation 14 (5) (2010) 782–800.
[61] T. M. Blackwell, P. J. Bentley, Dynamic search with charged swarms, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '02, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002, pp. 19–26.
[62] T. M. Blackwell, Swarms in dynamic environments, in: Proceedings of the Genetic and Evolutionary Computation Conference, Springer, 2003, pp. 1–12.
[63] T. Blackwell, J. Branke, et al., Multi-swarm optimization in dynamic environments, in: EvoWorkshops, Vol. 3005, Springer, 2004, pp. 489–500.
[64] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Transactions on Evolutionary Computation 10 (4) (2006) 459–472.
[65] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[66] S. Das, P. N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE transactions on evolutionary computation 15 (1) (2011) 4–31.
[67] E. Mezura-Montes, J. Velázquez-Reyes, C. A. Coello Coello, A comparative study of differential evolution variants for global optimization, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM, 2006, pp. 485–492.
[68] I. L. Cruz, L. Van Willigenburg, G. Van Straten, Efficient differential evolution algorithms for multimodal optimal control problems, Applied Soft Computing 3 (2) (2003) 97–122.
[69] J. J. Grefenstette, et al., Genetic algorithms for changing environments, in: 2nd Int. Conf. on Parallel Problem Solving from Nature, Vol. 2, 1992, pp. 137–144.
[70] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.), Springer-Verlag, London, UK, UK, 1996.
[71] M. Dorigo, Optimization, learning and natural algorithms, Ph. D. Thesis, Politecnico di Milano, Italy.
[72] A. Nareyek, Choosing search heuristics by non-stationary reinforcement learning, in: METAHEURISTICS: COMPUTER DECISION-MAKING, Kluwer Academic Publishers, 2001, pp. 523–544.
[73] P. Spanevello, M. M. de Oca, Experiments on adaptive heterogeneous PSO algorithms, in: Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search Algorithms, 2009, pp. 36–40.
[74] R. W. Morrison, Performance measurement in dynamic environments, in: GECCO workshop on evolutionary algorithms for dynamic optimization problems, Citeseer, 2003, pp. 5–8.
[75] R. Bond, A. Engelbrecht, B. Ombuki-Berman, et al., Evaluating landscape characteristics of dynamic benchmark functions, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2015, pp. 1343–1350.
[76] K. L. Duffin, W. A. Barrett, Spiders: a new user interface for rotation and visualization of n-dimensional point sets, in: Proceedings of the IEEE Conference on Visualization, 1994, pp. 205–211, CP22.
[77] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, C. Gagné, DEAP: Evolutionary algorithms made easy, Journal of Machine Learning Research 13 (2012) 2171–2175.
[78] T. M. Blackwell, Particle swarms and population diversity, Soft Computing 9 (11) (2005) 793–802.
[79] M. M. Ali, A. Törn, Population set-based global optimization algorithms: some modifications and numerical studies, Computers & Operations Research 31 (10) (2004) 1703–1725.
[80] J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics 16 (1) (1986) 122–128.

Appendix A. Implementation details

More information is provided about the implementation specifics of the study, including parameter selection of the moving peaks benchmark to allow creation of 27 different types of DOPs, as well as heuristic implementation details.

Moving peaks benchmark parameters

Recent findings by Bond *et al.* [75] show that the MPB is a problematic benchmark. Firstly, the MPB landscape is unrepresentative of real-life problems due to the symmetry of optima. Secondly, the MPB shows a bouncing effect near dimensional boundaries which causes peak shift to be less than the actual shift parameter. Lastly, the MPB control parameters lack the capacity to significantly alter landscape characteristics such as *ruggedness*, *dispersion*, *gradient*, and *searchability*. This study uses the MPB because:

- The symmetrical peaks and the bouncing effect of optima are similar across all environment and algorithm combinations, and do not affect the goals of the investigation.

- The absence of statistically significant differences between the ruggedness, dispersion, gradients and searchability of subsequent search landscapes over time ensures that all algorithms operate on the same search landscape complexity. Differences in algorithm performance will subsequently not be due to the search landscape's complexity changing over time.
- Duhain and Engelbrecht [1] present MPB parameter guidance to rigorously yield each of the 27 types of DOP.

The classification of Duhain and Engelbrecht [1] defines 27 distinctly different types of DOPs. Duhain and Engelbrecht provide considerations to carefully craft each environment:

- **Type I**: $h_s = 0$ and $s \neq 0$
- **Type II**: $h_s \neq 0$ and $s = 0$
- **Type III**: $h_s \neq 0$ and $s \neq 0$
- **Linear**: $\lambda = 1$ and $s \neq 0$
- **Circular**: $s = 0$ and the function is rotated on its center
- **Random**: $\lambda = 0$ and $s \neq 0$
- **Progressive**: low h_s and w_s relative to the domain, very high rate of function landscape change
- **Abrupt**: high h_s and w_s relative to the domain, low rate of function landscape change
- **Chaotic**: high h_s and w_s relative to the domain, high rate of function landscape change

For environments where function rotation is used, the rotation matrix $R_{ab}(\theta)$ rotates axis a towards axis b as follows:

$$R_{ab}(\theta) = \begin{cases} r_{ii} = 1 & \text{where } i \neq a, i \neq b \\ r_{aa} = & \cos(\theta) \\ r_{bb} = & \cos(\theta) \\ r_{ab} = & -\sin(\theta) \\ r_{ba} = & \sin(\theta) \\ r_{ij} = 0 & \text{otherwise} \end{cases} \tag{A.1}$$

where r_{ij} are the entries in the rotation matrix [76]. A single simple rotation is used to ensure that repeated rotations through 2π degrees will again yield the original landscape. The same hyper-plane is used for all rotations during a run and each run has a new randomly chosen hyper-plane to avoid bias. Spatial severity s is not used in a function rotation landscape, since the function rotation transformation changes the landscape instead. Given that the function domain is $R(0, 100)^d$, the cycle length C was set to 314 for progressive environments and 62 for abrupt and chaotic environments. These values for C mimic the spatial severity change step sizes for each environment to be similar to what s would have given. $C = 314$ ensures that points a distance of $r = 50$ from the function center $(50, 50)$ only change position by $^{2\pi r}/_{314} \approx 1$, which is close to $s = 1$ for progressive environments. Similarly, $C = 62$ results in $^{2\pi r}/_{62} \approx 5$, which is close to $s = 5$ for abrupt and chaotic environments.

For Type II environments, as defined by Eberhart *et al.* [31][35], optima maintain their positions while optima values change. Angeline's [32] *linear*, *circular* and *random* dynamics subsequently need to be expressed in terms of peak height changes as recommended by Duhain and Engelbrecht [1]. The ϕ parameter in table 2 shows how peak values are increased or decreased in linear, random, or circular patterns. The initial direction of peak height changes (up or down) is decided randomly every run. Linear peak changes start at one extreme (top or bottom) and progress towards the other extreme of the peak height range $h_p \in [30, 70]$ over the duration of the run. Circular peak changes oscillate over the peak height range $h_p \in [30, 70]$, and cycle lengths depend on each particular environment's height severity parameter h_s as outlined in table 2.

Using the considerations above together with permissible parameter value ranges of the original scenario 2 settings [29][30][77] yields 27 unique environments that are comparable to the majority of the DOP literature.

A noteworthy observation is that each iteration of HMHH always yielded the same number of fitness function evaluations for each environment. HMHH operated on a fixed population of 100 entities, and each chosen heuristic sampled the fitness function the same number of times every iteration (regardless of the exact entity-to-heuristic allocations). Hence the performance and behavior of the different algorithms were directly comparable in each of the 27 environments. If future studies include any heuristics that violates this property (i.e. if a heuristic does not to re-sample the fitness of every entity, or performs more than one re-sampling of fitness per entity), the study should instead focus on number of function evaluation directly, and not rely on the number of algorithm iterations.

Heuristic implementation details

More information is provided below about the implementation of each of the heuristics as part of HMHH hyper-heuristic framework:

- **APSO and CPSO**: Both variants share the same charge, core radius, and perception limit parameters as recommended by Blackwell [78]. The hyper-heuristic maintains APSO's roughly 50% charge-to-neutral particle ratio during entity initialization and reassignment by keeping track of entity allocation as the population continually grows and shrinks. Entity charges are assigned accordingly. All particles

are initially assigned random positions and zero velocities.

- **QPSO**: Two QPSO variants with different values for r_{cloud} (see table 3) are used. Similar to APSO, QPSO maintains a roughly 50% charge-to-neutral particle ratio during entity initialization and reassignment by tracking entity allocation as the population grows and shrinks. Entity charges are assigned accordingly. All particles are initially assigned random positions and zero velocities.

- **DE:** Two versions of *rand/1/bin* DE are added to the heuristic pool using low and high values for P_r respectively. β was set to 0.5 as recommended by Storn *et al.* [65], Cruz *et al.* [68] and Ali *et al.* [79]. Additionally, diversity is introduced after a change in the environment occurs by probabilistically mutating individuals using Gaussian noise. (using a mutation probability of 0.1 as shown in table 3).

- **RIGA**: All parameters are set as proposed by Grefenstette *et al.* [69] and Grefenstette [80]. RIGA uses a replacement rate of 0.1, i.e. every iteration every entity in the population (except the population's best entity) has a 10% probability of being reinitialized. RIGA thus introduces a substantial amount of diversity. A mutation rate of 0.001 is used as originally proposed by Grefenstette.

Semantic decoration of reassigned entities

Additional context and state information is required when entities are assigned to a different heuristic than what the entity was assigned before. Each heuristic in this study has a different strategy to *semantically decorate* an entity with valid context and state information:

- **APSO, CPSO and QPSO**: A new particle is created that requires 1) a charge value, 2) a velocity value, 3) a personal best value and neighborhood best value, and 4) knowledge of the quantum radius (for QPSO). The hyper-heuristic maintains a roughly 50% charge-to-neutral particle ratio (for CPSO and QPSO) at all times. New particles are initialized with a positive or neutral charge accordingly, depending on the charge ratio of the swarm at time t. The velocity of a reassigned entity is initialized to zero. The particle's personal best is set to the entity's current fitness value, and the particle is made aware of the neighborhood best of the swarm (which may get updated).

- **DE and RIGA:** No special context information is required. A new individual is simply created using the position of the candidate solution of the reassigned entity.

Acknowledgments

This work is based on the research supported by the National Research Foundation (NRF) of South Africa (Grant Number 46712). The opinions, findings and conclusions or recommendations expressed in this article is that of the author(s) alone, and not that of the NRF. The NRF accepts no liability whatsoever in this regard.