Heuristic Space Diversity Measures for Population-based Hyper-heuristics

Stefan A.G. van der Stockt
Computational Intelligence Research Group (CIRG)
Department of Computer Science
University of Pretoria
Pretoria, South Africa
stefan.vanderstockt@gmail.com

Andries P. Engelbrecht
Department of Industrial Engineering
Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
engel@sun.ac.za

Christopher W. Cleghorn
Computational Intelligence Research Group (CIRG)
Department of Computer Science
University of Pretoria
Pretoria, South Africa
ccleghorn@cs.up.ac.za

***Abstract*—A hyper-heuristic is an optimization approach that continually selects the most appropriate heuristic(s) to apply to an optimization problem. Hyper-heuristics conduct a search in the space of heuristics, or heuristic space, for the most suitable heuristic to apply to candidate solutions in problem space. Traditionally, hyper-heuristics manage relatively simple low-level heuristics, which are often based on human domain intuition. Increasingly, hyper-heuristics are being used in conjunction with population-based meta-heuristics as the low-level heuristics. A heuristic space diversity measure helps practitioners understand the behavior of hyper-heuristics that manage population-based heuristics. This paper discusses existing measures to quantify heuristic space diversity, highlights shortcomings of these existing measures, and proposes a new heuristic space diversity entropy-based measure. Spatial and temporal volatility measures that characterize entity-to-heuristic assignments are also proposed.**

***Index Terms*—hyper-heuristic, meta-heuristic, heuristic space diversity, heuristic space volatility**

I. Introduction

Heuristic search-based optimization is the process of finding the combination(s) of input value(s) in a search space of possible input values that yield "optimal" output values for an optimization problem. A plethora of heuristic and meta-heuristic algorithms have been developed to solve optimization problems, be that static or dynamic problems that are single- or multi-objective, and that may or may not be constrained [1]. The field of *Computational Intelligence* (CI) comprises of many *Swarm Intelligence* (SI) and *Evolutionary Computation* (EC) meta-heuristics that are inspired by nature. Most SI and EC approaches rely on a population of candidate solutions. Candidate solutions may be modified by various types of SI (and even non-SI) algorithms, and are referred to as *entities* instead of *individuals*, *particles*, or other term.

Determining the best meta-heuristic to use on any particular problem is elusively hard. Wolpert and Macready [2] published the "No Free Lunch" (NFL) theorems for optimization in 1997. The NFL theorems imply that no algorithm can be considered "generally better" than any other algorithm in domains where the NFL theorems hold. Naturally, the NFL theorems cause a stir among researchers that develop algorithms that they hope to be "generally better" than other algorithms.

Uneasiness around the NFL theorems prompted researchers to further understand the conditions under which the NFL theorems hold. Schumacher *et al.* [3] show that any two algorithms will have the same performance over a set of fitness functions (for any performance measure) *if and only if* the set of fitness functions is closed under permutation. A given set of fitness functions (i.e. problems) is *closed under permutation* if, for every function f in the set, all possible rearrangements of mappings from search space values $\mathbf{x}_i$ to function values $f(\mathbf{x}_i)$ are also contained as another function in the set. Igel and Toussaint [4] show that, for problems of interest, the sets of fitness functions that are closed under permutation comprise of an infinitesimally small part of the whole.

Additionally, Auger and Teytaud [5] show that the NFL theorems do not hold generally in continuous domains. Alabert *et al.* [6] 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 theoretical edge cases which require additional technical conditions that are simply too restrictive to be found in practice. A recent algorithm selection survey by Kerschke *et al.* [7] echoes these findings of how the necessary conditions for the NFL theorems to hold are simply not found in problems of interest.

These practical perspectives on the theoretical underpinnings of optimization encourage research into control adaptation methods that can find the best algorithms for particular sets of problems. Much work has been done in this area by practitioners in the EC and SI fields. Well-known approaches include *parameter control* (PC) for EAs [8] [9], *adaptive operator selection* (AOS) [10], *adaptive memetic algorithms* (MA) [11], *self-learning PSO* [12] [13], *heterogeneous PSO* [14] [15], and *algorithm portfolios* [16].

The fields of *Operations Research*, *Computer Science*, and *Artificial Intelligence* have produced optimization methods called *hyper-heuristics*. Hyper-heuristics adapt the optimization process by choosing which low-level heuristics to apply to a problem over time [17]. A simpler definition is that hyper-heuristics are "heuristics to choose heuristics" [17]. Hyper-heuristics aim to improve performance above that of using any heuristics in isolation. What distinguishes hyper-heuristics from other control adaptation approaches is the separation between solving a *problem* and searching for a suitable *method* to solve a problem [18]. Burke *et al.* [19] distinguish between *selection* hyper-heuristics, which use predefined selection operators that choose a suitable existing heuristic to apply next, and *generative* hyper-heuristics, which iteratively evolve customized heuristics (mostly via *genetic programming*) that are tailored to a domain (or problem instance). This paper focuses on perturbative population-based selection hyper-heuristics.

Poli and Graff [20] show that the NFL theorems do not automatically apply to meta-search methods such as hyper-heuristics. Kerschke *et al.* [7] remark that increased performance is possible by exploiting the complementary strengths of a set of algorithms through automatically selecting the most appropriate algorithm that is expected to perform best on the problem set at hand (at any stage of the search). Kerschke *et al.* list hyper-heuristics as one such promising field to increase performance by solving the *online per-instance algorithm selection problem.*

When a hyper-heuristic manages meta-heuristics that use a population[1] of entities, the manner in which the hyper-heuristic assigns entities to specific meta-heuristics becomes important to understand. *Heuristic space diversity* (HSD) [21] represents the degree to which the entity-to-heuristic assignments made by the hyper-heuristic is balanced or unbalanced. Characterizing a hyper-heuristic's HSD over time is valuable to understanding how the hyper-heuristic makes heuristic selection decisions. Many selection operators, in turn, may rely on HSD to determine new entity-to-heuristic assignments. Grobler *et al.* [21] quantify HSD to guide entity-to-heuristic allocation, and found that this strategy increased performance in static environments.

This paper discusses methods that quantify the HSD of population-based selection hyper-heuristics, and outlines deficiencies in these existing approaches. An improved measure for HSD is presented, along with various measures to quantify the spatial and temporal volatility of entity-to-heuristic assignments made by a hyper-heuristic. Section II provides an overview of hyper-heuristics as an approach for solving optimization problems, outlines the distinction between heuristic space and problem space, and discusses selection hyper-heuristics for population-based meta-heuristics. Section III discusses methods to characterize behavior in heuristic space. Section IV presents an improved HSD measure and discusses how entity reassignment numbers, frequencies, and volatility analysis forms a core part of understanding the heuristic space search behavior of a hyper-heuristic. Section V demonstrates how the proposed heuristic space behavior measures can be used by practitioners. Section VI concludes the paper.

II. Background on Hyper-heuristics

The distinction between the heuristic space and problem space is outlined bellow, followed by a deeper look at hyper-heuristics for population-based meta-heuristics.

[1]Such hyper-heuristics are classified as *multi-point search* hyper-heuristics by Burke *et al.* [18].

A. Problem Space versus Heuristic Space

A *hyper-heuristic* is defined by Chakhlevitch and Cowling [22] as a high-level control mechanism that uses limited problem-level knowledge to search a set of low-level heuristics for good *methods* and not good solutions. The classification of Burke *et al.* [18] discusses the *domain barrier* that separates the problems space from the heuristic space as follows:

- The *problem space* consists of the domain of the objective function. A point in the problem space represents a candidate solution to the optimization problem.
- The *heuristic space* is the search space of all applicable methods and their associated utility at time t during the optimization process. The domain of available heuristics is referred to as the *pool* of heuristics.

A point in the heuristic space represents a proposal of which heuristic(s) should be applied to improve the problem space solution vectors. This control flow is illustrated in figure 1.

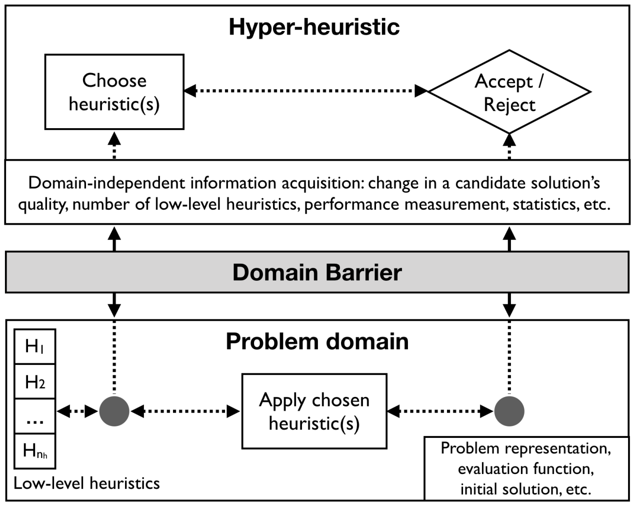

Hyper-heuristic
Choose heuristic(s)
Accept / Reject
Domain-independent information acquisition: change in a candidate solution's quality, number of low-level heuristics, performance measurement, statistics, etc.
Domain Barrier
Problem domain
H_1
H_2
...
H_{n_h}
Low-level heuristics
Apply chosen heuristic(s)
Problem representation, evaluation function, initial solution, etc.

Fig. 1: The conceptual framework for selection hyper-heuristics as presented by Burke *et al.* [18].

Ongoing research into hyper-heuristics continues to refine the working definition, goals, and constraints of what constitutes a hyper-heuristic approach. A growing number of researchers are revisiting the assumption that hyper-heuristics require a strict separation between the problem space and the heuristic space. Recently, Swan *et al.* [23] show that a "maximally restrictive" barrier between the hyper-heuristic and problem domain is counter-productive. The authors argue that certain types of a priori problem information (what the authors call "analytic information") can be exploited by a hyper-heuristic in a problem-independent manner to aid in the heuristic selection process. Such a broader view allows the incorporation of heuristic-to-domain mapping information, declarative domain descriptions, and constraint languages as data instead of code changes [23]. A research agenda has been defined to promote this view and gain consensus across the field as to what such an architectural vision for hyper-heuristics would look like [23] [24].

B. Hyper-heuristics for population-based meta-heuristics

The majority of hyper-heuristics in the literature employ *single-point search* techniques, where a single candidate solution is iteratively improved (or constructed) [17] [18]. Recent research trends focus increasingly more on *multi-point search* approaches, where a population of candidate solutions are assigned to heuristics by the hyper-heuristic. Multi-point search hyper-heuristics provide an opportunity to have the pool of heuristics comprise of more complex meta-heuristic techniques such as SI and EC algorithms. Burke *et al.* [17] review a number of studies where multi-point search hyper-heuristics are used to select which meta-heuristics should modify which entities in the population at time t.

Grobler *et al.* [25] [26] [27] present the *heterogeneous meta-hyper-heuristic* (HMHH) as a selection hyper-heuristic framework that manages a pool of population-based meta-heuristics. Each heuristic is a meta-heuristic algorithm configuration that comprises of specific 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. This approach is analogous to practitioners providing manually crafted domain-specific heuristics, except that the heuristics in HMHH are specific instances of meta-heuristics that are suitable in the problem domain. The intention is to have a pool of several different (yet specific) algorithm configurations with known behaviors.

Adaptation in HMHH occurs by assigning a specific heuristic to manage each entity in the population. Different heuristics become distinct behaviors that yield different outcomes when applied to any given candidate solution. Generally, the position and fitness of an entity is noticeably altered in different ways depending on which heuristic acts upon the entity. For example, an entity modified for one iteration by a DE algorithm variant will generally have a noticeably different output candidate solution compared to if the entity was, instead, modified by a PSO or GA variant. HMHH strives to give the most promising heuristics every opportunity to succeed by letting more entities be updated by the most suitable heuristics.

HMHH is shown in algorithm 1 with parts of the original notation adapted to align with notation in this paper. Every k iterations HMHH employs a selection operator, ς, to assign entities in the parent population E to heuristics. The performance feedback of each heuristic h_m, namely Q_{δ_m}, may be used by ς in determining entity-to-heuristic allocations.

The choice of selection operator allows HMHH to exhibit different types of heuristic allocation behavior. Van der Stockt and Engelbrecht [28] [29] [30] investigate the performance and behavior of various selection operators for HMHH applied to solving different types of dynamic optimization problems.

Algorithm 1 Heterogeneous Meta-Hyper-Heuristic [26]

```
$E \leftarrow$ initialize parent population of $n_s$ solution entities.
$h_j(t) \leftarrow$ the heuristic algorithm applied to entity $e_j$ at iteration $t$.
$k \leftarrow$ algorithm iterations between heuristic assignments.$^1$
for all entities $e_j \in E$ do
    $h_j(1) \leftarrow$ choose random initial heuristic algorithm for $e_j$.
end for
$t = 1$.
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.
        $Q\delta_m(t) \leftarrow$ total improvement of entities assigned to $h_m$
                for the last $k$ iterations.$^2$
    end for
    for all entities $e_j \in E$ do
        $h_j(t+k) \leftarrow$ Select next heuristic for entity $e_j$ using
                selection operator $\varsigma(e_j, Q\delta_m(t))$.
    end for
    $t = t + k$.
end while
```

Notes:

1) $k = 5$ is used in [26].
2) In [26], $\varsigma(e_j, Q\delta_m(t))$ is rank-based tabu search [19]. Many other hyper-heuristic selection operators may be used with HMHH.

III. Measuring Diversity in Heuristic Space

Hyper-heuristics that manage population-based meta-heuristics continually need to assign the most suitable heuristics to entities. Good hyper-heuristics prevent a downward spiral called *heuristic space convergence* where one heuristic "takes over" by holding on to all assigned entities forever. If this happens, many heuristics may be unable to provide adequate feedback. Depending on the exact selection operator logic, the optimization process may devolve to using only that one dominating heuristic.

Insight into the diversity of the heuristic space provides insight into whether a hyper-heuristic is able to cope with the above situation. The diversity of entities-to-heuristic assignments may be assessed in a number of ways:

Entity allocation plots

A simple method to analyze how entities are allocated across heuristics is to plot the percentage of entities that is managed by each heuristic. Example entity allocation plots are presented in figure 2, which shows how many entities are allocated to each heuristic over all 1000 iterations. Nepomuceno and Engelbrecht [31] use behavior profile plots to illustrate the balance of how particle swarm optimization behaviors are allocated across particles. Van der Stockt and Engelbrecht [30] apply this technique to reveal how different hyper-heuristics assign entities across a pool of hyper-heuristics.

Entity allocation plots offer an intuitive understanding of how balanced entity assignments are across heuristics, but tend to become difficult to interpret when more than three heuristics are used. It is hard to compare the entity-to-heuristic assignments of multiple algorithm runs against each other to determine which runs show higher heuristic diversity than others. Comparisons of experiments that use a different number of heuristics affects the interpretation of the plot, since the point of equal balance of n_s entities across n_h heuristics occurs at n_s/n_h (which, visually, is located at different points on the y-axis for different values of n_h). Lastly, statistical comparisons are hard to perform when using this measure, since each plot yields a multi-variate time series of values that cannot be readily compared using established statistical tests.

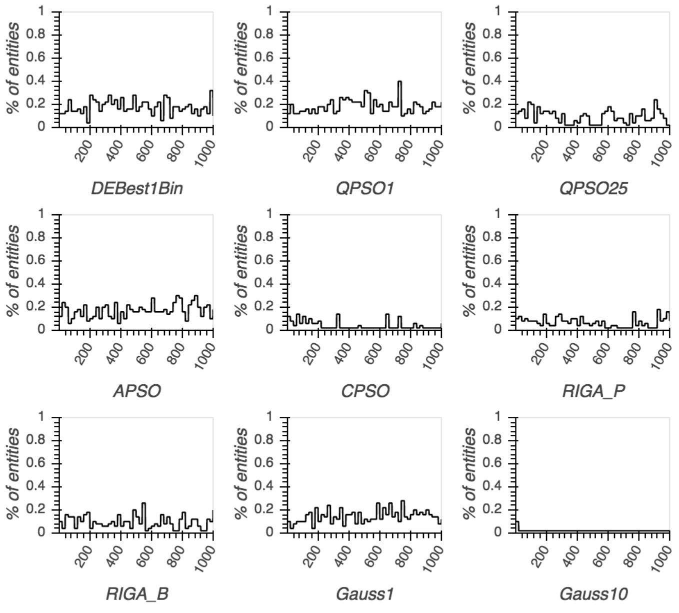


Fig. 2: Allocations of entities across nine heuristics.

Aggregated entity allocation plots

Viewing the aggregated statistics of entity allocations across time allows practitioners to obtain a more concise view of entity-to-heuristic allocations. Statistical measures of centrality, such as the mean or median, provides insight into which heuristics gets applied to the majority of entities most often. Statistical measure of spread, such as the standard deviation or inter-quartile range (IQR), give a sense of how much heuristic assignments vary over time. Visualizations such as box whisker plots allow this distribution analysis to be illustrated in concise form. Figure 3 shows an example of the entity-to-heuristic allocation distributions of the same algorithm run that was illustrated in figure 2.

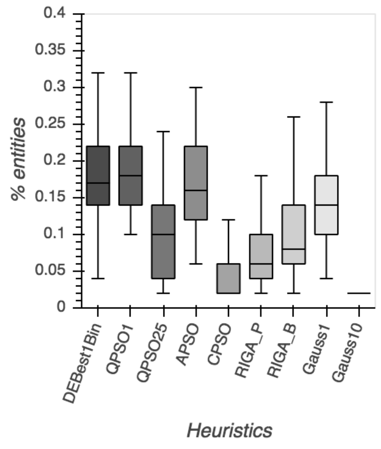


Fig. 3: Box whisker plot of entity-to-heuristic allocations of the same data that is used in figure 2.

Aggregated entity allocation plots improve upon entity allocation plots by producing scalar values for each heuristic instead of a time series values. However, each heuristic still yields a separate output value. Since the resulting distributions are all related, it is hard to compare different algorithms or experiment runs against each other (visually or statistically).

Heuristic space diversity

The HSD metric, $\mathcal{H}(t)$, proposed by Grobler and Engelbrecht [26] measures the spread of the population of entities across heuristics in the heuristic space as

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

where t is the current algorithm iteration, n_s is the number of entities, n_h is the number of heuristics, n_m is the number of entities assigned to heuristic h_m, α is a scaling factor (here $\alpha = 100$), and $T = n_s/n_h$. Values of $\mathcal{H}(t) \approx 1$ indicate that entities are balanced equally across all heuristics while $\mathcal{H}(t) \approx 0$ show that a few heuristics are controlling almost all of the entities. $\mathcal{H}(t)$ does not give an indication as to which heuristic has the greatest number of entities assigned, only that an imbalance is present.

Overall, the $\mathcal{H}(t)$ HSD measure offers a better description of how entities are distributed across heuristics than raw or aggregated entity allocation plots. The measure is normalized so that values of 0.0 or 1.0 are, respectively, always associated with imbalanced or balanced entity assignments, regardless of the number of heuristics. The $\mathcal{H}(t)$ measure can be used as a time series to directly inspect the HSD at a specific time t, or to compare the diversity of multiple hyper-heuristics across time (i.e., for vector-based or time series based analysis). The $\mathcal{H}(t)$ measure can also be aggregated over to produce a single scalar value that is easier to incorporate into existing statistical testing procedures (such as the average, median, or standard deviation of HSD values, among other type of aggregation). Grobler *et al.* [21] use $\mathcal{H}(t)$ to guide entity-to-heuristic assignments as part of the selection operator logic of the hyper-heuristic.

IV. Holistic Heuristic Space Behavior Measures

An improved HSD measure is outlined below, as well as proposed measures to characterize the number, frequency, and volatility of the entity assignments made by a hyper-heuristic.

A. An Improved Heuristic Space Diversity Measure

The $\mathcal{H}(t)$ measure, as defined in equation (1), has a drawback in that the value of $\mathcal{H}(t)$ can become less than zero: if there are more than four heuristics, and all n_s entities are assigned to a single heuristic, then equation (1) simplifies to $\alpha(1 - 2(n_h - 1)/1.5n_h)$. To avoid this situation, any hyper-heuristic must maintain $n_m > 1$ at all times to ensure that $\mathcal{H}(t) > 0$. This places an artificial constraint on the hyper-heuristic that is undesirable.

Another option is to use a different method to calculate the disparity between the entity assignments of different heuristics. Budescu and Budescu [32] discuss a measure called *normalized entropy* that is based on a measure of diversity of populations of people by Teachman [33], which uses Shannon's entropy theory [34]. Normalized entropy is extended below as a measure called $\mathcal{N}(t)$ that can be used as an alternative to $\mathcal{H}(t)$, where $\mathcal{N}(t)$ is defined as

$$\mathcal{N}(t) = -\sum_{m=1}^{n_h} \frac{p(t)\log_2(p(t))}{log_2(n_h)} \tag{2}$$

where

$$p(t) = \begin{cases} \dfrac{n_m(t)}{n_s} & \text{if } n_m(t) > 0 \\ \epsilon & \text{otherwise} \end{cases}$$

where ϵ is a very small positive constant. Similar to $\mathcal{H}(t)$, values of $\mathcal{N}(t) \approx 1$ when entity-to-heuristic allocations are balanced, and $\mathcal{N}(t) \approx 0$ if one heuristic dominates. The $\mathcal{N}(t)$ measure improves upon $\mathcal{H}(t)$ since $\mathcal{N}(t)$ can never be negative, and $\mathcal{N}(t)$ is resilient in cases where a heuristic has zero entities assigned (i.e. $n_m(t) = 0$) which removes the artificial constraint that $\mathcal{H}(t)$ placed on practitioners.

B. Entity Reassignment Behavior

HSD gives insight into the state of entities-to-heuristic allocations, but does not portray the full dynamics behind entity reassignments. The assignment of entities to heuristics by a hyper-heuristic can be analyzed along a number of dimensions, namely

- the *balance* of entity assignments across the pool of heuristics at time t,
- the *number* of entities that are reassigned to new heuristics by a hyper-heuristic at time t,
- the *frequency* (or temporal volatility) with which a hyper-heuristic assigns entities to new heuristics over a run, and
- the (spatial) *volatility* with which a hyper-heuristic disrupts existing entity allocations over a run.

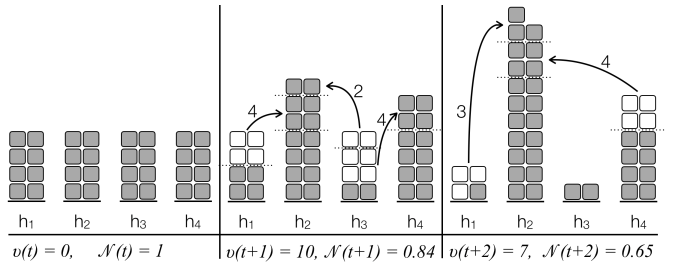
The *balance* of entity assignments is measured using $\mathcal{N}(t)$, which is calculated after every iteration using equation (2). The resulting series of HSD values over time may or may not vary, depending on the actions taken by the hyper-heuristic.

The *number* of entities that are reassigned to different heuristics at time t expresses the magnitude of entity reassignments that occur. Various hyper-heuristics perform entity reassignments in different ways, depending on the selection logic employed. The subset $E^*(t) \subset E$ contains those entities in the population E that are reassigned new heuristics at time t. The number of entities that are reassigned to new heuristics at time t, $\upsilon(t)$, is simply the size of the subset $E^*(t)$, i.e.

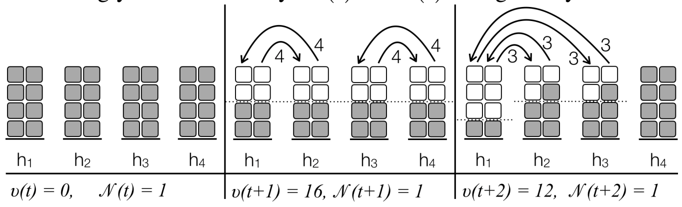
$$\upsilon(t) = |E^*(t)| \tag{3}$$

The minimum value of $\upsilon(t)$ is zero, which occurs when no entities are reassigned to new heuristics. The maximum value of $\upsilon(t)$ is $n_s - n_f$, where $n_s = |E|$ is the total number of entities in the HMHH parent population E, and n_f is the sum of the minimum allowed entity counts for each heuristic.

It is possible for two hyper-heuristics to have similar $\mathcal{N}(t)$ values over a period of time, yet have vastly different $\upsilon(t)$ values. Figure 4 illustrates how $\mathcal{N}(t)$ and $\upsilon(t)$ are related using an example. Consider two hyper-heuristics that each manage


4
2
4
4
3
h_1
h_2
h_3
h_4
$\upsilon(t) = 0,\ \mathcal{N}(t) = 1$
$\upsilon(t+1) = 10,\ \mathcal{N}(t+1) = 0.84$
$\upsilon(t+2) = 7,\ \mathcal{N}(t+2) = 0.65$

(a) The hyper-heuristic reassigns entities across heuristics in an increasingly unbalanced way. $\mathcal{N}(t)$ and $\upsilon(t)$ change every iteration.


4
4
4
4
3
3
3
3
h_1
h_2
h_3
h_4
$\upsilon(t) = 0,\ \mathcal{N}(t) = 1$
$\upsilon(t+1) = 16,\ \mathcal{N}(t+1) = 1$
$\upsilon(t+2) = 12,\ \mathcal{N}(t+2) = 1$

(b) The hyper-heuristic reassigns entities in a balanced way. $\mathcal{N}(t)$ remains constant while $\upsilon(t)$ changes every iteration.

Fig. 4: Illustrative example of measuring heuristic space diversity using $\mathcal{N}(t)$ and the entity reassignment rate using $\upsilon(t)$ for four heuristics, h_1, h_2, h_3, and h_4, and 32 entities over three time steps t, $t+1$, and $t+2$.

32 entities. The first hyper-heuristic in figure 4a shows steadily decreasing HSD values as more entities are progressively assigned to a single heuristic (namely h_2). The second heuristic in figure 4b actually reassigns a larger number of entities to new heuristics each iteration (i.e., $\upsilon(t)$ is greater for the second hyper-heuristic), yet HSD values remain constant at $\mathcal{N}(t) = 1$.

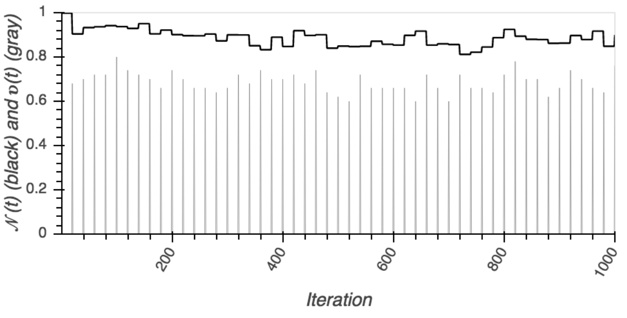
The balance and number of entity assignments as measured, respectively, by $\mathcal{N}(t)$ and $\upsilon(t)$, need to be considered independently. By itself, $\mathcal{N}(t)$ only gives insight into how balanced the entity assignments are across the heuristics, regardless of the quantity or frequency of entity reassignments. On its own, $\upsilon(t)$ measures the churn (or agitation) of entities by showing how many entities were reassigned to new heuristics, regardless of the balance or frequency of entity assignments. High $\upsilon(t)$ values do not always imply large changes in $\mathcal{N}(t)$ values. Figure 5 shows the $\mathcal{N}(t)$ and $\upsilon(t)$ values over time for the same roulette wheel selection operator depicted in figures 2 and 3, but using different logic to trigger heuristic selection.

The *frequency* with which a hyper-heuristic reassigns entities to new heuristics is the proportion of all HMHH algorithm iterations T where entities are reassigned to new heuristics. The entity reassignment frequency, δ, is defined as

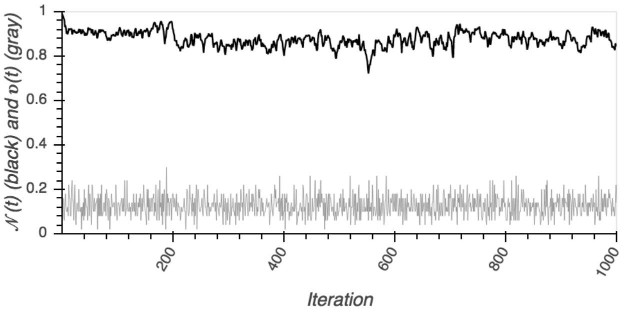
$$\delta = \frac{|T'|}{|T|} \tag{4}$$

where $T' \subseteq T$ is the subset of all algorithm iterations T where at least one entity reassignment occurs (i.e. where $\upsilon(t) > 0$). The δ measure provides a sense of the temporal volatility of hyper-heuristic assignments.

The *volatility* with which a hyper-heuristic makes adjustment to entity allocations is the mean number of entities that



(a) Performing heuristic selection for all entities simultaneously every k iterations.



(b) Performing heuristic selection for entities at random iterations with a per-entity reassignment probability of $P = 1/k$.

Fig. 5: $\mathcal{N}(t)$ (black) and $\upsilon(t)$ (gray) over time for the roulette wheel selection operator. Different selection logic yields comparable $\mathcal{N}(t)$ values, but drastically different $\upsilon(t)$ values.

are reassigned in an algorithm run. The mean is computed relative to the proportion of iterations where entity reassignments occur (i.e. T') as follows:

$$\varphi = \frac{\sum_{t' \in T'} \upsilon(t')}{|T'|} \tag{5}$$

In other words, algorithm iterations that do not contain any entity reassignments do not skew the mean calculation. The φ measure reports how large the average disruption to entity allocations is when heuristic allocation changes are made.

V. Visualizing Heuristic Space Behavior

The proposed HSD measure ($\mathcal{N}(t)$), the entity reassignment frequency (δ), and the volatility of heuristic allocation changes (φ) are used in an experimental setting to characterize the behavior of various HMHH selection operators. The term "HMHH selection operator" and hyper-heuristic are used interchangeably in the remainder of this paper.

A. Experimental setup

A pool of heuristics, H, is constructed that consists of $n_h = 9$ different EC and SI meta-heuristics. HMHH is used to manage a population of entities, E. The population size is set to $n_s = 50$. Each entity, $e_j \in E$, is assigned to a specific heuristic $h_m \in H$ by the HMHH selection operator. Each heuristic h_m manages $n_m(t)$ entities at time t. For each entity, the probability of the hyper-heuristic selecting heuristic h_m at time t is $P_m(t)$, where $m = \{1, \dots, n_h\}$. Various selection operators determine each $P_m(t)$ differently. The following HMHH selection operators are used:

- **Simple random (Rand)** selection always assigns each entity $e_j \in E$ to heuristic h_m with equal probability. The probabilities of selection of each heuristic remain constant, i.e. $P_m(t) = \frac{1}{n_h}$.
- **Roulette wheel (RoulM)** selection assigns each entity $e_j \in E$ to heuristic h_m with a probability relative to the mean fitness of entities assigned to h_m, i.e.

$$\mu_m(t) = \frac{\sum_{j=1}^{n_m} f_j(t)}{n_m} \tag{6}$$

  The probability of selection of each heuristic h_m is set to $P_m(t) = \frac{\mu_m(t)}{\sum_{l=1}^{n_h} \mu_l(t)}$.
- **Entity tournament (ETour)** selection employs tournament selection from EC [1]. For every entity $e_j \in E$, a tournament set of entities, $T_j \in E$, is randomly drawn. The entity $e_w \in T_j$ with the best fitness is considered the winner. The entity e_j under consideration is assigned the same heuristic as e_w, while e_w remains unchanged. Tournament sizes may be in the range $\{2, \dots, n_s\}$.
- **Ant-inspired fitness proportional (AProp)** selection [14] is inspired by the fundamental version of the ant colony optimization meta-heuristic (ACO-MH) [35] [36]. Each heuristic h_m is assigned a *pheromone concentration* $\rho_m(t)$ that is used to calculate $P_m(t)$ as

$$P_m(t) = \frac{\rho_m(t)}{\sum_{l=1}^{n_h} \rho_l(t)} \tag{7}$$

  Roulette wheel selection assigns entities to heuristics relative to $P_m(t)$. Initially, each heuristic has a pheromone concentration $\rho_m(1) = \frac{1}{n_h}$. Pheromone levels are updated based on the size of the fitness improvement of entities. For function maximization $\rho_m(t)$ is updated as

$$\rho_m(t) = \rho_m(t-1) + \sum_{j=1}^{n_m} \max\{0, f_j(t) - f_j(t-1)\} \tag{8}$$

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

$$\rho_m(t) \leftarrow \frac{\sum_{l=1, l \neq m}^{n_h} \rho_l(t)}{\sum_{l=1}^{n_h} \rho_l(t)} \times \rho_m(t) \tag{9}$$

  Ant-inspired fitness proportional selection emphasizes the magnitude of the raw fitness value improvements.
- **Frequency improvement (Freq)** selection is based on Nepomuceno and Engelbrecht's *frequency-based heterogeneous PSO* behavior selection scheme (FB-HPSO) [15]. FB-HPSO selects new particle behaviors based on the frequency with which each behavior improved the fitness of particles over the previous k iterations. FB-HPSO is adapted here to select the best heuristic for each entity. A frequency score, $\chi_m(t)$, is calculated for each heuristic h_m based on the number of times each entity

e_j assigned to h_m improved its fitness since the current heuristic was assigned, i.e.

$$\chi_m(t) = \sum_{i=1}^{k}\sum_{j=1}^{n_m} \begin{cases} +1 & \text{if } f_j(t-i) \text{ improved} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

A maximum of $k = 10$ prior iterations are considered to prevent historical feedback from overshadowing more recent feedback, as per the guidance of Nepomuceno and Engelbrecht [15]. Tournament selection using $\chi_m(t)$ is applied to determine the winning heuristic, with tournament sizes in the range $\{2, \ldots, n_h\}$.

- **Frequency improvement reinforcement learning (RL-Freq)** selection uses a reinforcement learning approach similar to Narayek [37] and Burke *et al.* [19]. A rank score, $r_m(t)$, is maintained for each heuristic, and $r_m(1) = 0$ initially. Changes in rank, $\Delta r_m(t)$, are based on how many times the fitness of entities assigned to h_m improved versus remaining the same or stagnating, i.e.

$$\Delta r_m(t) = \sum_{j=1}^{n_m} \begin{cases} +1 & \text{if } f_j(t) \text{ improved} \\ -1 & \text{if } f_j(t) \text{ otherwise} \end{cases} \tag{11}$$

Ranks are updated as $r_m(t) = r_m(t-1) + \Delta r_m(t)$. The maximum rank is $r_{max} = n_s$ and the minimum rank is $r_{min} = -n_s$. 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 [19]). Rank ties are broken randomly.

- **Difference proportional (DProp)** selection [38] by Spanevello and Montes de Oca probabilistically assigns entities to that heuristic h_b that contains the fittest entity $e_b \in E$. The probability $P_b(t)$ of reassigning entity e_j from the entity's currently assigned heuristic h_j to h_b is

$$P_b(t) = \frac{1}{1 + \exp\left(-\beta \frac{f_b(t) - f_j(t)}{|f_b(t)|}\right)} \tag{12}$$

where $\beta = 5$ as recommended by Spanevello and Montes de Oca. The function $P_b(t)$ has a sigmoidal shape that enables difference proportional selection to increase the probability of assigning poor performing entities to h_b, and lower the probability of reassigning well-performing entities to different heuristics.

The moving peaks benchmark (MPB) by Branke [39] is used create 27 unique classes of dynamic optimization problems (DOPs) as identified by Duhain and Engelbrecht [40]. HMHH is configured with each of the selection operators outlined above. Each configuration is run for 1000 iterations on 71 random instances[2] of each type of DOPs, resulting in 1917 algorithm runs. Heuristic changes are triggered randomly for every entity with a probability set relative to 20% of the time between MPB landscape changes. The values of δ, φ, and the median and IQR values of $\mathcal{N}(t)$ are subsequently calculated for each run. Figure 6 shows the result of the analysis.

B. Results

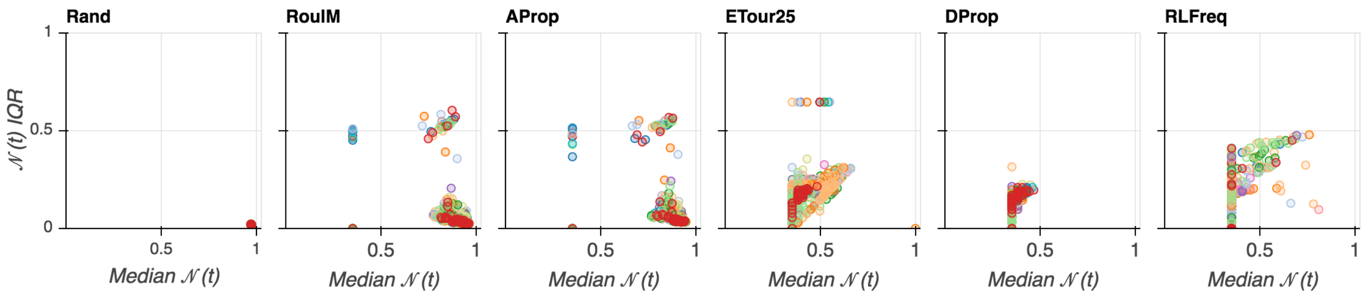
Scatter plots of the median and IQR of $\mathcal{N}(t)$, and δ versus φ, respectively, are a natural way[3] to visualize this information. Figure 6a shows the median and IQR values of $\mathcal{N}(t)$. These non-parametric measures make fewer assumptions about the distribution of the data, and the mean and standard deviation could potentially also be used. Figure 6b highlights the entity assignment volatility of each hyper-heuristic, both in spatial and temporal terms by using φ and δ, respectively.

Practitioners may typically analyze figure 6 as follows:

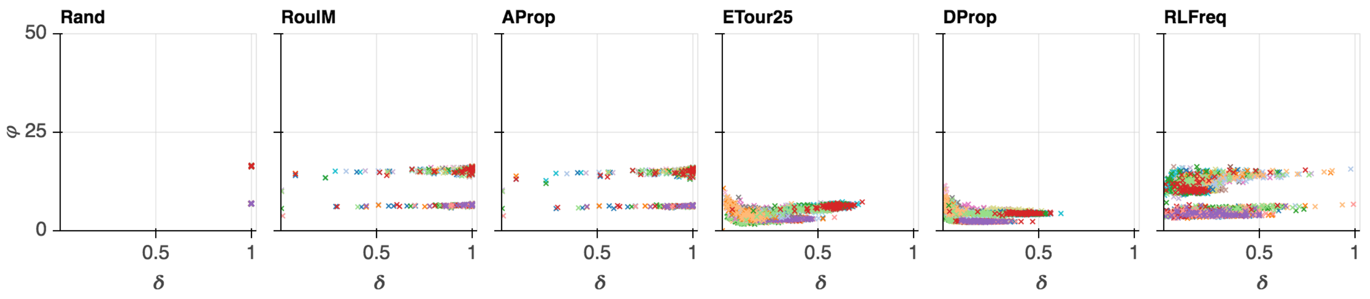
- **Rand:** Figure 6a shows that each sample had $\mathcal{N}(t) \approx 1$ with no variation (indicated by low IQR values), regardless of the problem type. Figure 6b reveals that a consistent number of entities were assigned new heuristics every iteration (i.e., φ was high and $\delta = 1$). The discrepancy in the two groups of samples is due to the fact that the HMHH parameter k was dependent on the cycle length parameter of the MPB, which resulted in higher proportions of entities to be reassigned in the one group.
  Figure 6 confirms that **Rand** was impartial to reassigning entities across heuristics, always created balanced assignments with high HSD, and was unaffected by the type of problem or sample being addressed.
- **RoulM:** A wider distribution of $\mathcal{N}(t)$ values across problem types and samples shows that **RoulM** was more selective than **Rand**. Certain runs showed completely balanced HSD values with low variance (very similar to **Rand**). Other runs showed that **RoulM** frequently assigned most entities to only a few heuristics. The wide range of different IQR values for $\mathcal{N}(t)$ when the median $\mathcal{N}(t)$values was less than one reveals that **RoulM** could selectively maintain high or low HSD for different runs. In contrast to **Rand**, the wide distribution of δ values reveals that **RoulM** was selective in entity reassignments, sometimes choosing not to reassign any entities at certain algorithm iterations of a run.
- **AProp:** The results for **AProp** appear very similar to **RoulM**, which indicates that the behavior of **AProp** was indistinguishable from random heuristic assignment. The results may help practitioners to investigate the causes, i.e., perhaps pheromone updates using equation (8) were too similar to the outcome of equation (6), or pheromone evaporation using equation (9) was too strong each iteration.
- **ETour25:** The hyper-heuristic showed an approximately triangular relationship between median and IQR values of $\mathcal{N}(t)$. Within a single algorithm run, the hyper-heuristic

[2]Derrac *et al.* [41] and Garcia *et al.* [42] recommend that the number of samples, s, used for a nonparametric Friedman test-based analysis must satisfy $2a \leq s \leq 8a$, where a is the number of algorithms being compared. The choice of $s = 71$ samples satisfies the requirement for the analysis of the nine stand-alone heuristics (i.e. $a = 9$ satisfies $s \leq 8 \times 9$), the analysis of 6 hyper-heuristics (i.e. $a = 6$ yields $s \geq 2 \times 6$), as well as any joint analysis of all heuristics and hyper-heuristics (i.e. $a = 15$ results in $s \geq 2 \times 15$).

[3]Other multi-dimensional visualization techniques may of course be used as well, such as spiderweb plots or parallel coordinate plots.



(a) Median versus IQR for $\mathcal{N}(t)$ for 71 algorithm runs (by selection operator).



(b) φ versus δ for 71 algorithm runs (by selection operator).

Fig. 6: Measuring the balance and volatility of a hyper-heuristic over 71 algorithm samples.

showed evidence of always assigning all entities to a single heuristic (lower left corner of the triangle), fluctuating between assigning all entities to a single heuristic and assigning entities across heuristics in a more balanced manner (top left corner of the triangle), and also showed a tendency to maintain lop-sided assignments (where a large proportion of entities are assigned to few heuristics, indicated by $\mathcal{N}(t) \approx 0.65$) with high variation in the balance of assignments.

Generally, φ values were much lower and more uniform across problem types and samples for **ETour25** than for **Rand** or **RoulM**, indicating markedly different volatility in entity assignments. Large variations in δ values can be seen for **ETour25**. Certain runs resulted in few to no entity-to-heuristic changes at all in a run (i.e., heuristic space convergence may have occurred), but other runs showed that up to 70% of algorithm iterations contained heuristic reassignments. These plots are useful for practitioners to understand what the source of the variation is, i.e. is the type of problem correlated with δ or φ values.

- **DProp:** The HSD behavior of **DProp** was dominated by allocating all entities to a single heuristic across most iterations of each run, as indicated by low median and IQR values for $\mathcal{N}(t)$. **DProp** was less volatile than **Rand** or **RoulM** and showed φ values similar to **ETour25**. The δ value distribution for **DProp** reveals that, similar to **ETour25**, the hyper-heuristic was highly adaptive to the specific problem instance being solved. Practitioners can now isolate specific samples and further investigate which problem types or cases caused **ETour25** to display higher variation in HSD behavior than **DProp**.
- **RLFreq:** The straight vertical line in figure 6a shows that **RLFreq** allocated all entities to a single heuristic, but was able to rapidly re-diversify entity-to-heuristic allocations when needed (as shown by the large range of IQR values). This finding is supported by figure 6b which shows how **RLFreq** showed large variations in both φ and δ across samples. Practitioners might investigate if there was any correlation between these samples ()where median $\mathcal{N}(t)$ values were greater than 0.35) and specific problem types and/or φ and δ values.

VI. Conclusion

The heuristic space behavior of a hyper-heuristic that manages population-based meta-heuristics is critical to understanding how the hyper-heuristic operates. The newly proposed heuristic space diversity measure, $\mathcal{N}(t)$, improves upon previous measures by placing fewer algorithm parameter constraints on the practitioner. Additionally, this paper proposes volatility measures, namely φ and δ, that characterize the amount of spatial and temporal disruption that the hyper-heuristic selection logic causes.

Future work should characterize the behavior of different hyper-heuristics on the same set of benchmark problems, and compare the results to determine if there are any correlations between certain types of behaviors and increased or decreased performance. Comparison of the behavior of a hyper-heuristic against control groups (such as random selection) will be valuable to understand if and how intelligent selection mechanisms improve upon simply maintaining algorithmic diversity.

References

[1] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. USA: Wiley Publishing, 2007.

[2] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[3] C. Schumacher, M. D. Vose, and L. D. Whitley, "The no free lunch and problem description length," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2001, pp. 565–570.

[4] C. Igel and M. Toussaint, "On classes of functions for which no free lunch results hold," *Information Processing Letters*, vol. 6, no. 86, pp. 317–321, 2003.

[5] A. Auger and O. Teytaud, "Continuous lunches are free!" in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2007, pp. 916–922.

[6] A. Alabert, A. Berti, R. Caballero, and M. Ferrante, "No-free-lunch theorems in the continuum," *Theoretical Computer Science*, vol. 600, pp. 98–106, 2015.

[7] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *Evolutionary Computation*, vol. 27, no. 1, pp. 3–45, 2019, mIT Press.

[8] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith, "Parameter control in evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms*, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 19–46.

[9] F. G. Lobo, C. F. Lima, and Z. Michalewicz, *Parameter Setting in Evolutionary Algorithms*, 1st ed. Springer Publishing Company, Incorporated, 2007, vol. 54.

[10] J. Maturana, F. Lardeux, and F. Saubion, "Autonomous operator management for evolutionary algorithms," *Journal of Heuristics*, vol. 16, no. 6, pp. 881–909, Dec 2010.

[11] N. Krasnogor and 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*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 987–994.

[12] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 627–646, 2012.

[13] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4515–4538, 2011, special Issue on Interpretable Fuzzy Systems.

[14] F. V. Nepomuceno and A. P. Engelbrecht, "A self-adaptive heterogeneous PSO inspired by ants," in *Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß, and T. Stützle, Eds., vol. 7461. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 188–195.

[15] ——, "A self-adaptive heterogeneous PSO for real-parameter optimization," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*. IEEE Press, Piscataway, NJ, June 2013, pp. 361–368.

[16] B. A. Huberman, R. M. Lukose, and T. Hogg, "An economics approach to hard computational problems," *Science*, vol. 275, no. 5296, pp. 51–54, 1997.

[17] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[18] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds. Boston, MA: Springer US, 2010, pp. 449–468.

[19] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, Dec 2003.

[20] R. Poli and M. Graff, "There is a free lunch for hyper-heuristics, genetic programming and computer scientists," in *Genetic Programming*, L. Vanneschi, S. Gustafson, A. Moraglio, I. De Falco, and M. Ebner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 195–207.

[21] J. Grobler, A. P. Engelbrecht, G. Kendall, and V. S. S. Yadavalli, "Heuristic space diversity management in a meta-hyper-heuristic framework," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, July 2014, pp. 1863–1869.

[22] K. Chakhlevitch and P. Cowling, "Hyperheuristics: Recent developments," in *Adaptive and Multilevel Metaheuristics*, C. Cotta, M. Sevaux, and K. Sörensen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–29.

[23] J. Swan, P. De Causmaecker, S. Martin, and E. Özcan, "A re-characterization of hyper-heuristics," in *Recent Developments in Metaheuristics*, L. Amodeo, E.-G. Talbi, and F. Yalaoui, Eds. Cham: Springer International Publishing, 2018, pp. 75–89.

[24] J. Swan, S. Adriaensen, M. Bishr, E. Burke, J. Clark, P. De Causmaecker, J. Durillo, K. Hammond, E. Hart, C. Johnson, Z. Kocsis, B. Kovitz, K. Krawiec, S. Martin, J. Merelo, L. Minku, E. Özcan, G. Pappa, E. Pesch, P. Garcia-Sànchez, A. Schaerf, K. Sim, J. Smith, T. Stützle, V. Stefan, S. Wagner, and X. Yao, "A research agenda for metaheuristic standardization," in *Proceedings of the XI Metaheuristics International Conference*, Agadir, United Kingdom, 2015, pp. 1–3.

[25] J. Grobler, A. P. Engelbrecht, G. Kendall, and V. S. S. Yadavalli, "Multi-method algorithms: Investigating the entity-to-algorithm allocation problem," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*. IEEE Press, Piscataway, NJ, June 2013, pp. 570–577.

[26] J. Grobler, A. P. Engelbrecht, G. Kendall, and V. S. S. Yadavalli, "Heuristic space diversity control for improved meta-hyper-heuristic performance," *Information Sciences*, vol. 300, pp. 49–62, 2015.

[27] ——, "Alternative hyper-heuristic strategies for multi-method global optimization," in *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 2010, pp. 1–8.

[28] S. A. G. Van der Stockt and A. P. Engelbrecht, "Analysis of hyper-heuristic performance in different dynamic environments," in *Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*. IEEE Press, Piscataway, NJ, Dec 2014, pp. 1–8.

[29] ——, "Analysis of global information sharing in hyper-heuristics for different dynamic environments," in *Proceedings of the 2015 IEEE Congress on Evolutionary Computation*, Sendai, Japan, 2015, pp. 822–829.

[30] ——, "Analysis of selection hyper-heuristics for population-based meta-heuristics in real-valued dynamic optimization," *Swarm and Evolutionary Computation*, vol. 43, pp. 127–146, 2018.

[31] F. V. Nepomuceno and A. P. Engelbrecht, "Behavior changing schedules for heterogeneous particle swarms," in *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, Sep. 2013, pp. 112–118.

[32] D. V. Budescu and M. Budescu, "How to measure diversity when you must." *Psychological Methods*, vol. 17, no. 2, pp. 215–227, 2012.

[33] J. D. Teachman, "Analysis of population diversity: Measures of qualitative variation," *Sociological Methods & Research*, vol. 8, no. 3, pp. 341–362, 1980.

[34] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[35] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[36] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, vol. 2. IEEE Press, Piscataway, NJ, July 1999, pp. 1470–1477.

[37] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in *Metaheuristics: Computer Decision-Making*. Boston, MA: Springer US, 2004, pp. 523–544.

[38] P. Spanevello and M. A. Montes de Oca, "Experiments on adaptive heterogeneous PSO algorithms," in *Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, 2009, pp. 36–40.

[39] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, vol. 3. IEEE Press, Piscataway, NJ, July 1999, pp. 1875–1882.

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

[41] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[42] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010, special Issue on Intelligent Distributed Information Systems.