Assessing Hyper-Heuristic Performance

Nelishia Pillay[1], Rong Qu[2],
[1]Department of Computer Science, University of Pretoria, South Africa
[2] ASAP Group, University of Nottingham, UK

Hyper-heuristics is an emerging area of Computer Science and as such certain areas of the field need further development before it is established. One such area is assessing hyper-heuristic performance. Limited attention has been paid to assessing the generality performance of hyper-heuristics. The performance of hyper-heuristics has been predominately assessed in terms of optimality which is not ideal as the aim of hyper-heuristics is not to be competitive with state of the art approaches but rather to raise the level of generality. Furthermore from existing literature it is evident that different hyper-heuristics aim to achieve different levels of generality and need to be assessed as such. To cater for this the paper firstly presents a new taxonomy of four different levels of generality that can be attained by a hyper-heuristic. The paper then proposes a performance measure to assess the performance of different types of hyper-heuristics at the four levels of generality in terms of generality. Three case studies from the literature are used to demonstrate the application of the generality performance measure. The paper concludes by examining how the generality measure can be combined with other measures, such as optimality, to assess hyper-heuristic performance on more than one criterion.

***Index Terms*—hyper-heuristics, performance evaluation, generality, performance measure**

I. Introduction

Hyper-heuristics is an emerging technique that has proven to be effective at solving various problems including educational timetabling, vehicle routing, personnel scheduling and packing problems, amongst others [1]. The aim is to achieve a higher level of generality by using a general framework over a set of different problems or instances, rather than designing tailor-made problem specific algorithms which may perform well on some problems or instances but poorly on the others [2].

The generality of hyper-heuristics is achieved by exploring in a higher level search space of heuristics, rather than in a lower level space of direct solutions [3]. In the literature, most classic metaheuristics such as simulated annealing and evolutionary algorithms directly explore the search space of solutions to a problem. Hyper-heuristics adaptively search in the heuristic space at the higher level, thus achieving greater generality by leaving problem specific details to a set of low-level heuristics which operate upon the direct solution space. Note that most classic metaheuristics can also be employed by a hyper-heuristic to explore the high level heuristic space.

However, there has been no formalization of measurement on how well a hyper-heuristic performs in terms of generality. In some studies the performance of the hyper-heuristic is compared to that of state-of-the-art approaches [4], [5]. Such assessment is not appropriate against the main aim of a hyper-heuristic, to produce good results over a problem set rather than best results for certain problem instances without considering its general performance across others. Furthermore, depending on the type of hyper-heuristic, the performance expectations differ. For example, a generation constructive hyper-heuristic aims at creating new low-level constructive heuristics and as such cannot be expected to perform as well as state-of-the-art metaheuristics which are dedicated to improve complete solutions. A more suitable comparison would be with existing constructive heuristics for the problem domain.

Manuscript received December 1, 2012; revised August 26, 2015. Corresponding author: N.Pillay (email: npillay@cs.up.ac.za

The aim of this paper is to present a performance measure to assess the performance of hyper-heuristics in terms of generality. As a starting point the paper focuses specifically on the evaluation of hyper-heuristic performance in solving discrete optimisation problems. Future work will extend this to emerging application areas of hyper-heuristics such as continuous optimisation, multiobjective optimisation and automated design.

Section II defines the terminology used in the paper. Section III provides an overview of hyper-heuristics, which reveals that different research addresses hyper-heuristics of different levels of generality. These can range from performing well on a set of problem instances for a single problem [6] to achieving generality across multiple problem domains [7]. Based on this overview Section IV then presents a new taxonomy of generality and illustrates this using examples from the literature. Section V presents a performance measure to evaluate the performance of hyper-heuristics in terms of generality rather than optimality, and describes how this can be used to assess the performance of the different types of hyper-heuristics. Three case studies are used to illustrate the application of the performance measure. The section concludes by examining how the proposed generality measure can be combined with other criterion, such as optimality, to assess hyper-heuristic performance using more than one criterion. Finally, Section VI provides a summary of the findings of the paper and proposes future research directions. The main contribution of the paper is a generality performance assessment measure for hyper-heuristics.

This research aims to motivate and stimulate more advanced approaches based on existing diverse work in hyper-heuristics, by providing important complementary mechanisms with different classifications of general hyper-heuristics. The contributions of this research are as follows:

- A taxonomy defining different levels of generality a hyper-heuristic can attain.
- A performance measure to assess the performance of hyper-heuristics in terms of generality.

II. Terminology

This section defines the terms used in the paper in the context of the research presented.

Problem domain refers to a domain that may include various problems. Each problem domain has an underlying problem and the various problems are variations of the underlying problem. For example, the educational timetabling domain includes the school timetabling, university course timetabling and examination timetabling problems. Similarly, the packing problem domain includes one-dimensional(1D), two-dimensional(2D) and three-dimensional(3D) bin-packing.

Problem is one of the problems in a particular problem domain. Examples include the examination timetabling problem in the educational timetabling problem domain and the capacitated vehicle routing problem in the vehicle routing domain.

Problem instance refers to an instance of a particular problem. For example, *pur93* is a problem instance for the examination timetabling problem [8]. Similarly, *eil76* [9] is a problem instance for the symmetric travelling salesman problem.

Benchmark set refers to a set of problem instances for a particular problem. For example, the Toronto benchmark set for the examination timetabling problem and the Scholl benchmark set for the one-dimensional bin-packing problem.

III. Overview of Hyper-Heuristics

A hyper-heuristic aims to improve its generality to different problems by working in a heuristic space instead of a solution space [3], [10]. At the higher level, a hyper-heuristic carries out problem independent search on the space of heuristics, configuring problem specific low-level heuristics which operate on the search space of direct solutions. Based on the methods used at the high and low levels, four classes of hyper-heuristics have been defined, namely, selection constructive, selection perturbative, generation constructive and generation perturbative [1], [11], [12], [10].

Selection constructive hyper-heuristics select low-level constructive heuristics to apply at each decision making point (i.e. assign a value to a decision variable) in creating a solution to the problem. For example, for the examination timetabling problem, different low-level constructive heuristics (i.e. largest degree, largest weighted degree, largest colour degree, largest enrolment and saturation degree) can be used to create a solution [5], [13]. These heuristics assess the difficulty of scheduling examinations, and assign them one by one accordingly to construct a solution, most difficult first. At each step during the high level search, the hyper-heuristic configures the low-level constructive heuristics to construct a solution.

Selection perturbative hyper-heuristics are used to choose a perturbative low-level heuristic at each point to iteratively improve a complete solution subject to a stopping condition. An initial solution can be created either randomly or using a constructive heuristic. For the examination timetabling problem, perturbative heuristics include moving an examination to another period, swapping the periods of two randomly selected examinations, deallocating an examination [14].

In a single point based hyper-heuristic search, moves are made based on the selected heuristic associated with an acceptance criterion.

Hyper-heuristics performing a multipoint search such as genetic algorithms are not composed of separate components for heuristic selection and move acceptance as the multipoint search technique by its nature performs both these tasks [15].

Generation constructive hyper-heuristics focus on creating new constructive heuristics by combining variables representing given low-level heuristics, components of existing low-level heuristics and problem characteristics using arithmetic and conditional operators. These are essentially a priority function such as an arithmetic function or arithmetic rule. In the case of university course timetabling, this could be an arithmetic function comprised of standard addition, subtraction, multiplication and division operators to configure and combine problem characteristics such as the number of students taking the lecture, the number of other conflicting lectures (with common students), the number of feasible timetable periods available to allocate a lecture to [16].

Like generation constructive hyper-heuristics, generation perturbative hyper-heuristics create new heuristics but these are perturbative. The new heuristics are comprised of the given low-level heuristics or components thereof and conditional and/or iterative operators. For example, for the Boolean satisfiability problem, low-level perturbative heuristics select and move variables and clauses. The components of low-level heuristics, such as net gain and select a clause randomly, are combined with conditional operators to produce new heuristics [17].

The new heuristics created by generation constructive and generation perturbative hyper-heuristics can be disposable or reusable. Disposable heuristics are created specifically for a particular problem instance, while reusable heuristics are effective when applied to new problem instances different from those used to create the heuristic. Genetic programming [18] and its variations, such as grammar-based genetic programming [19] and grammatical evolution [20], have chiefly been employed by hyper-heuristics to create new constructive and perturbative heuristics.

From the existing research conducted in this area, it is evident that hyper-heuristics are proposed to achieve different levels of generality based on the problem at hand. The following section provides a taxonomy to standardise the levels of generality based on the existing research in hyper-heuristics.

IV. Levels of Generality

In this section we firstly present a taxonomy for different levels of generality that must be achieved by hyper-heuristics.

This is based on applications of hyper-heuristics in the literature, as well as levels we see as needed to build an extensible taxonomy with future growth of the field. Examples are then provided from the literature to illustrate each level in the taxonomy.

A. Taxonomy for Hyper-Heuristic Generality

The taxonomy is comprised of the following four levels of generality:

- Level 1: Single problem, single benchmark - In this case the hyper-heuristic should produce good results for a particular benchmark set of instances for a particular problem. For example, the ITC 2007 benchmark set for the examination timetabling problem [21] or the Faulkenauer benchmark set for the one-dimensional bin-packing problem [22]. The hyper-heuristic solves problem instances in a benchmark set for a single problem and hence its ability to generalize is the lowest for this level.
- Level 2: Single problem, multiple benchmarks - The hyper-heuristic should perform well for different benchmark sets for a particular problem. For example, for the examination timetabling problem different benchmark sets would include the Toronto benchmark [13] and the ITC 2007 benchmark [21]. Similarly, for the one-dimensional bin-packing domain the problems would include the Faulkenauer and Scholl benchmark sets [22], [23]. The instances contained in the different benchmark sets must differ with respect to the problem constraints. For example, the Toronto and ITC 2007 benchmark sets differ with respect to the hard and soft constraints that must be met. For some problems, such as bin packing and the travelling salesman problem, there is just one constraint, so the benchmarks sets differ in terms of the problem features. For example, the Faulkenauer and Scholl benchmark sets differ in terms of the range for the size of items, number of items, bin capacities, etc. Hyper-heuristics in this category generalize better than Level 1 hyper-heuristics in that they can produce good solutions over more than one benchmark set with problem instances of differing constraints or features. In the case of problem features there must be a sufficient difference in features as in the Faulkenauer and Scholl benchmark sets. If the differences in features is small these will be equivalent to different problem instances in the same benchmark set rather than different benchmark sets.
- Level 3: Single domain, multiple problems - The hyper-heuristic must produce good solutions for different problems for a particular problem domain. The problem domain has a specific underlying problem, e.g. packing items in a bin, allocating educational events to timetable periods. The hyper-heuristic solves variations of the underlying problem. For example, one-dimensional, two-dimensional and three-dimensional bin-packing problems for the packing domain. There must be a single underlying problem, for example educational timetabling involves allocating educational events and variations include examination timetabling, university course timetabling and school timetabling for the educational timetabling domain.
- Level 4: Cross Domain - The hyper-heuristic is applied to problems across different problem domains. Each problem domain has a different underlying problem. For example, solving various discrete optimsation problems such as the symmetric travelling salesman problem, one-dimensional bin-packing problem and personnel scheduling problem [24], [25]. Similarly, solving continuous optimisation problems such as function optimisation and time series forecasting.

The levels distinguish between the hyper-heuristics in terms of the extent to which they generalize, with Level 1 being the lowest level of ability to generalize and Level 4 the highest level. Progression from one level to the next is not indicative of the problems being more challenging to solve at the higher levels but rather that the hyper-heuristic is able to generalize further. Also please note that there is no polymorphic relationship between the levels, i.e. a hyper-heuristic a Level 4 will not necessarily perform better than a hyper-heuristic at Level 1 in attaining Level 1 generality. The reason for this is that the aim of the hyper-heuristics at different levels are different and a hyper-heuristic that is developed to perform well across different problem domains for example, will not necessarily perform well for a particular problem or set of benchmark instances. These levels are based on the current state of the field of hyper-heuristics. However, the idea is to provide an extensible taxonomy that can be adapted according to the growth of the field.

In the following sections we provide an example for each level of generality from the literature. Note that we use *good* results here to indicate the expected performance at different levels of generality. A performance measure concerning different levels of generality is proposed and discussed in Section V.

B. Level 1: Single Problem-Single Benchmark

A fair amount of the studies on hyper-heuristics aim to achieve Level 1 generality, i.e. on a benchmark set of instances for a particular problem. One of the earlier studies [26] employed a selection constructive hyper-heuristic to solve the examination timetabling problem for the Toronto benchmark. In [27] a selection perturbative hyper-heuristic is implemented to solve the set covering problem using the OR-Library benchmark set. Similarly, in [16] a generation constructive hyper-heuristic is applied to the university course timetabling problem. The hyper-heuristic was applied to the ITC 2007 curriculum based course timetabling benchmark set. A further example is the generation perturbative hyper-heuristic employed by Fukunaga [17] to solve the Boolean satisfiability problem using the problem instances from SATLIB. Comparisons are usually conducted against other metaheuristics and evolutionary algorithms designed for the particular problem. Table I lists some examples of Level 1 generality hyper-heuristics.

TABLE I
EXAMPLES OF HYPER-HEURISTICS AT LEVEL 1 GENERALITY

Problem Domain	Type of Hyper-Heuristic	Benchmark Set	Reference
Examination timetabling	Selection constructive	Toronto	Pillay [28]
Constraint satisfaction	Selection constructive	Generated	Terashima-Marín et al. [29]
Travelling tournament problem	Selection perturbative	Easton	Chen et al. [30]
Set covering problem	Selection perturbative	ORLib	Ferreira et al. [27]
Examination timetabling	Selection perturbative	Toronto	Burke et al. [31]
Examination timetabling	Generation constructive	Toronto	Bader-El-Den et al. [4]
2D strip packing	Generation constructive	Generated	Burke et al. [32]
Multidimensional knapsack problem	Generation constructive	ORLib	Drake et al. [33]
One dimensional bin packing	Generation constructive	ORLib	Sim et al. [34]
3-SAT	Generation perturbative	SatLib	Bader-El-Den et al. [35]

C. Level 2: Single Problem-Multiple Benchmarks

Level 2 hyper-heuristics aim to solve problem instances in different benchmark sets for a problem. Benchmark sets usually differ in terms of characteristics. For example, for the examination timetabling problem different benchmark sets have different hard and soft constraints for the problem. Similarly, for the one-dimensional bin-packing problem different benchmark sets vary in the dimensions of items and bin capacities.

In the study conducted by Terashima-Marín et al. [6], a selection constructive hyper-heuristic is used to create solutions to the 2D regular cutting stock problem. The hyper-heuristic was evaluated on a combination of subsets of various 2-D regular cutting stock problems and randomly generated problem instances. In [36] a selection perturbative hyper-heuristic is used to solve a combination of problem instances from two benchmark sets for the examination timetabling problem, namely, the Toronto benchmark set and the Yeditepe benchmark set. In the study conducted by Burke et al. [32] a generation constructive hyper-heuristic is used to create new low-level constructive heuristics for the two dimensional strip packing problem. The hyper-heuristic is trained and tested on a combination of problem instances from different benchmark sets. In these studies the problem instances from the different benchmark sets are combined into a single set to which the hyper-heuristic is applied.

Some studies have applied the hyper-heuristic developed to different benchmark sets for a problem, but the performance of the hyper-heuristic is not evaluated on the combined set of problem instances from all the benchmark sets. Hence, the hyper-heuristic is applied and evaluated separately for each benchmark set. For example, in the study conducted by Sabar et al. [37], a selection constructive hyper-heuristic is implemented to solve the examination timetabling problem for the Toronto and ITC 2007 benchmark sets. In [15] a selection perturbative hyper-heuristic is used to solve problem instances in various benchmark sets for the school timetabling problem. Sim and Hart [38] have implemented a generation constructive hyper-heuristic to solve the one-dimensional bin-packing problem which was applied to 5 one-dimensional bin-packing benchmark sets. In this research, the hyper-heuristics developed are compared with other algorithms designed for the different benchmark data sets, respectively, to demonstrate its generality across different data sets of instances. Two different rankings, or comparison analyses, are used separately for the different data sets. Examples of Level 2 generality hyper-heuristics are listed in table II.

D. Level 3: Single Domain - Multiple Problems

Hyper-heuristics in this category solve different problems in a particular domain. For example, in the study presented in [43], a selection constructive hyper-heuristic is used to solve bin-packing problems. Problem instances from benchmark sets for the one dimensional bin packing problem, two dimensional bin packing problems and two dimensional bin packing problems with irregular concave polygons are combined to evaluate the hyper-heuristic.

In other studies, the hyper-heuristic is applied to the benchmark set for each problem separately, i.e. the problem instances for the different problems are not combined. Terashima-Marín et al. [44] employ a selection perturbative hyper-heuristic to solve packing problems. The hyper-heuristic solves both two dimensional regular and irregular problems. Burke et al. [5] also use a selection constructive hyper-heuristic for the educational timetabling domain. Examination timetabling problem instances and university course timetabling problem instances are solved by the hyper-heuristic. Similarly, Misir et al. [45] employ a selection perturbative hyper-heuristic to solve three healthcare scheduling problems, namely, home care scheduling, nurse rostering and patient admission scheduling. In this set of research, the hyper-heuristics developed are compared with other algorithms

TABLE II
EXAMPLES OF HYPER-HEURISTICS AT LEVEL 2 GENERALITY

Problem Domain	Type of Hyper-Heuristics	Benchmark Sets	Reference
One dimensional bin packing	Selection constructive	Scholl and Faulkenauer	Pillay [39]
One dimensional bin packing	Selection constructive	Faulkenauer and Technische Universitat Darmstadt	Ross et al. [40]
2D regular stock cutting	Selection constructive	Generated, ORLib, Martello and Vigo Berkey and Wang	Terashima-Marín et al. [6]
Examination timetabling	Selection constructive	Toronto and ITC 2007	Sabar et al. [37]
School timetabling	Selection perturbative	Abramson, Valouxis, Beligiannis, South African primary school South Africa high school	Raghavjee et al. [15]
Examination timetabling	Selection perturbative	Toronto and Yeditepe	Özcan et al. [36]
Vehicle routing	Generation constructive	Solomon and Sim and Hart	Sim et al. [41]
Constraint satisfaction	Generation	Generated, RLFAP-graphs et al. jobShop-e0ddr1	Sosa-Ascenio [42]
SAT	Generation perturbative	SatLib and Gottlieb	Fukunaga [17]

designed for the different benchmark problems, respectively, to demonstrate its generality across different problems in the same domain. Two different rankings, or comparison analyses, are used separately for the different data sets to assess the generality of the developed hyper-heuristics.

Table III presents examples of Level 3 generality hyper-heuristics.

E. Level 4: Cross Domain

Cross domain hyper-heuristics are applied across different problem domains. The concept of cross domain hyper-heuristics was initiated by the hyper-heuristic community in 2011 to increase the generality level of hyper-heuristics. The HyFlex Java framework [46] was developed and made publicly available to promote research on cross domain selection perturbative hyper-heuristics. The framework provides the methods for creating an initial solution, the low-level perturbative heuristics, the objective function and problem instances for six discrete optimization problems, namely, Boolean satisfiability, one dimensional bin packing, permutation flow shop, personnel scheduling, travelling salesman and vehicle routing. It is used to implement different selection perturbative hyper-heuristics for all six domains. Two challenges, CHeSC 2011 and CHeSC 2014, using this framework, were held to promote research in cross domain hyper-heuristics.

Hence, the research in this area has focused on using the HyFlex framework for cross domain hyper-heuristics. The early work includes that presented in [7], where an adaptive dynamic heuristic set (ADHS) strategy was used for heuristic selection, and an adaptive iteration limited list based threshold accepting (AILLA) approach was used for move acceptance. This hyper-heuristic has produced the best results in CHeSC 2011. The selection perturbative hyper-heuristic implemented by Chan et al. [47] takes an analogy from pearl hunting to explore the space of low-level pertubative heuristics. Cichowicz et al. [48] implemented a five-stage hyper-heuristic and a genetic hive hyper-heuristic for the cross domain challenge. Subsequent to the challenge, this field has grown rapidly with a number of attempts to produce better performing cross domain selection perturbative hyper-heuristics.

Examples of Level 4 generality hyper-heuristics are depicted in Table IV.

V. ASSESSING THE PERFORMANCE OF A HYPER-HEURISTIC

This section firstly introduces a measure for assessing the generality performance of hyper-heuristics. The application of this generality measure is then illustrated using case studies from the literature. The section ends by examining how the generality measure presented can be combined with other criteria to assess hyper-heuristic performance.

A. Measure for Assessing Hyper-Heuristic Performance

In the existing literature, the performance of a hyper-heuristic is assessed by comparing its performance to either existing low-level constructive or perturbative heuristics, other hyper-heuristics, or optimisation techniques on a set of problem instances, depending on the type of hyper-heuristic. The comparisons have been based on optimality which is not appropriate for the comparison of hyper-heuristic performance. Ranking has previously been used to perform this comparison for selection perturbative hyper-heuristics [46]. In this section we examine the use of ranking for comparing hyper-heuristic performance and introduce an alternative measure to assess the generality of hyper-heuristics.

In ranking, the approach being compared which produces a solution with the best objective value is assigned a rank of 1, the one with the next best objective value a rank of 2, and

TABLE III
EXAMPLES OF HYPER-HEURISTICS FOR LEVEL 3 GENERALITY

Problem Domain	Problems	Type of Hyper-Heuristics	Benchmark Sets	Reference
Educational timetabling	Examination timetabling and course timetabling	Selection constructive	Generated	Burke et al. [26]
Offline packing problems	1D single bin, 2D regular and 2D irregular single bin	Selection constructive	Generated, Scholl, Washer and Faulkenauer	Lopez-Camacho et al. [43]
Educational timetabling	Examination timetabling and course timetabling	Selection constructive	Toronto Metaheuristic network	Burke et al. [5]
Educational timetabling	Examination timetabling and course timetabling	Selection constructive	Toronto Metaheuristic network	Qu et al. [3]
2D stock cutting	Regular and irregular	Selection constructive	Generated, ORLib, Martello and Vigo Berkey and Wang	Terashima-Marín [44]

TABLE IV
EXAMPLES OF HYPER-HEURISTICS AT LEVEL 4 GENERALITY

Problem Domains	Type of Hyper-Heuristics	Benchmark Sets	Reference
Nurse rostering Course timetabling	Selection perturbative	Aickelin and Downsland Metaheuristic network	Burke et al. [49]
Examination timetabling Dynamic vehicle routing	Selection perturbative	ITC 2007, Taillard, Christofides, Fischer	Sabar et al. [25]
Examination timetabling Vehicle routing	Generation perturbative	ITC 2007, Golden, Christofides	Sabar et al. [24]
6 CHeSC problem domains	Selection perturbative	CHeSC benchmark sets	Misir et al. [7]
6 CHeSC problem domains	Selection perturbative	CHeSC benchmark sets	Chan et al. [47]
6 CHeSC problem domains	Selection perturbative	CHeSC benchmark sets	Cichowicz [48]
6 CHeSC problem domains	Selection perturbative	CHeSC subsets of benchmark sets	Lehrbaum [50]

so on. In the case of ties on objective values, the approaches are assigned the same rank. The ranks are then aggregated or averaged to compare the performance of the approaches. The approach with the lowest total or average is the best performing approach. Algorithm 1 presents a typical ranking algorithm.

Algorithm 1 Ranking algorithm

1: Apply approach a_i to problem instance p_j
2: Assign a rank r_j to approach a_i based on the objective value of the solution it has produced for problem instance p_j
3: The overall rank for each approach a_i is $R_i = \frac{\sum_{j=1}^{n} r_j}{n}$, where n is the number of problem instances, i.e. j = 1, ..., n.

Ranking favours the approach that produces the best objective value for most of the problem instances. As such it is not capable of measuring whether the approach performs well over the problem set with instances of different scales of objective values. We propose the *standard deviation of differences (SDD)* as a measure to assess how well a hyper-heuristic performs over a set of problem instances. Equation (1) presents the formula for *SDD*, where N is the number of problem instances.

$$SDD(H) = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N-1}} \tag{1}$$

$$x_i = 0 \qquad if\ o_i = 0\ and\ b_i = 0 \tag{2}$$

$$x_i = (|o_i - b_i|)/average(o_i, b_i)) * 100 \qquad otherwise \tag{3}$$

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{4}$$

SDD is the standard deviation of the percentage difference between o_i the objective value of the solution produced by the approach for problem instance i, and the best known objective value or the best objective value b_i among the approaches being compared. As indicated in equation 3 the percentage difference is calculated to be the percentage of the absolute value of the objective value and the best known objective value divided by the average value of both these values if both o_i and b_i are not zero. Alternatively, the maximum of both objective values could be taken instead of the average. If

both o_i and b_i are zero, the percentage difference x_i is zero as indicated in equation 2. A lower standard deviation indicates that there is less variance in the distance from the best known objective value over the set of problem instances and hence a better ability to generalize. Hence, if the difference from the optimum is the same for all problem instances, there is no variation and the SDD value is 0 indicating that the hyper-heuristic generalizes well. Thus, the best SDD value a hyper-heuristic can attain is 0. However, as can be seen from the literature and the examples presented below this is not very likely.

The aim of the SDD is to assess the variation of the difference from optima (or best known) over the set of problem instances. By definition this is what the standard deviation assesses. The less variation in the differences the better the hyper-heuristic can generalize. Thus if the difference from the optimum is the same there is no variation in the differences and SDD will be zero. The aim of SDD is to assess generality, rather than optimality (as the majority assessment used in the literature). For example, given three problem instances with o_i and b_i values 1025, 107, 29 and 1020, 102, 24 respectively. There is no variation in the difference from the optimum giving a value of 0 for SDD indicating that the hyper-heuristic has generalized (rather than optimised) well over the set of problem instances. Optimality can be added as an additional criterion as discussed below.

We illustrate both ranking(RAN) and SDD using two hyper-heuristics in [5], which aim to perform well over the problem set, and two approaches aiming to produce the best objective values for the problem instances in the benchmark set [51], [52]:

- GHH - The tabu search hyper-heuristic in [5].
- Multistage - The multistage version of GHH in [5].
- SB - The sequential allocation approach with backtracking in [51] for examination timetabling problems. This approach has produced the best objective values for more problem instances in the Toronto benchmark set than any other approach applied to this set.
- TS - In [52] a tabu search is applied to the examination timetabling problem, aiming to produce the best objective value for the problem instances. This approach is also applied to the Toronto benchmark set and has not produced the best objective value for any of the problem instances.

Table V displays the RAN and SDD values for the four approaches to the 11 problem instances in the Toronto benchmark set.

TABLE V
PERFORMANCE COMPARISON USING RAN AND SDD

Approach	*RAN*	*SDD*
SB	1.8	14.66
GHH	1.91	5.99
Multistage	2.73	8.3
TS	3.36	10.54

From Table V it can be seen that the sequential allocation approach with backtracking (SB) by [51] has the best RAN value, and produces the best objective value for more of the problem instances in the benchmark set. In terms of SDD, however, both the hyper-heuristics, GHH and Multistage, perform better than the approaches which aim to produce the best results for the problem instances. SB has the worst SDD value, and produces the best results for some of the problems instances but performs poorly on the other problem instances. This indicates that it does not generalize as well over the problem set compared to the other approaches. While TS does not produce the best results for any of the problem instances, it generalizes better over the problem set than SB. With SDD the measurement of generality reveals more information on how the approaches generlize across different problem instances.

B. Different Types of Hyper-Heuristics and Generality Levels

This section firstly examines how to assess the performance of the different types of hyper-heuristics, namely, selection constructive, selection perturbative, generation constructive and generation perturbative, against the four levels of generality defined in section III. Then performance evaluation for the four levels of generality is explained.

Selection constructive hyper-heuristics

Selection constructive hyper-heuristics choose a low-level constructive heuristic to apply at each point in constructing a solution to a problem. Hence, the overall aim of these hyper-heuristics is the same as that of low-level constructive heuristics, namely, to create a good initial solution which can be optimised further using other techniques. Hence, the performance of these hyper-heuristics, using the SDD performance measure, can be assessed by comparing the results obtained to :

1) The existing low-level constructive heuristics that are used to create initial solutions for the problem domain.
2) Other selection constructive hyper-heuristics that have been applied to the benchmark set of problems.

Selection perturbative hyper-heuristics

Selection perturbative hyper-heuristics aim to improve an initial solution created randomly or using a constructive heuristic. These heuristics essentially select a move operator to apply in the solution space. The hyper-heuristics can be evaluated by comparing their performance, using the SDD performance measure, to:

- Other selection perturbative hyper-heuristics applied to the same benchmark sets.
- Other optimisation techniques applied to the same benchmark sets.

Generation constructive hyper-heuristics

Generation constructive hyper-heuristics create new low-level heuristics and hence their performance is measured in terms of the new heuristics. The induced heuristics are used to create an initial solution to the problem. As such they are evaluated by comparing their performance, using the SDD performance measure, to:

- Existing low-level constructive heuristics for the problem domain.
- Other generation constructive hyper-heuristics applied to the same domain.

Generation perturbative hyper-heuristics

Generation perturbative hyper-heuristics create new low-level perturbative heuristics. These are essentially local search operators. These hyper-heuristics are evaluated by comparing their performance, using the SDD performance measure, to:

- Existing local search operators for the problem domain. Each generated perturbative heuristic and existing local search operator is applied for a number of iterations to an initial solution. A set time or number of iterations can be used as a termination criterion. Alternatively, this iterative process can be terminated when there is no further improvement in the objective value of the resulting solution.
- Other generation perturbative hyper-heuristics all applied to the same benchmark set of problems.

In the case of Level 1 hyper-heuristics, SDD is calculated over the benchmark set to show the generality of the hyper-heuristic. For hyper-heuristics aiming to attain Level 2 generality, i.e. performing well over *n* benchmark sets for a particular problem, SDD must be calculated over the instances in all *n* benchmark sets. For example, suppose that two benchmark sets containing 12 and 8 problem instances are used to evaluate the hyper-heuristic, SDD is calculated over the 20 problems. Similarly, for Level 3 and Level 4 hyper-heuristics these measures must be calculated over all the problem instances from the different benchmark sets for the problem domain and the instances from the benchmark sets for the different domains, respectively.

C. Case Studies

This section illustrates and demonstrates the use of the SDD performance measure established in Section V-A in assessing the performance of hyper-heuristics in three studies from the literature, namely:

- A selection constructive Level 1 hyper-heuristic [39]
- A selection perturbative Level 4 hyper-heuristic [50]
- A generation constructive Level 3 hyper-heuristic [53]

The study in [39], [54] employs a selection constructive hyper-heuristic to solve the Toronto benchmark examination timetabling set of problems. Evolutionary algorithm hyper-heuristics are investigated to explore the space of heuristic combinations. Each low-level constructive heuristic in the combination is applied in sequence to select the next examination to allocate to the timetable. The study compares the performance of four evolutionary algorithm selection constructive hyper-heuristics:

- FHC - The heuristic combinations are of fixed length, equal to the number of examinations to be allocated.
- VHC - The heuristic combinations are of variable length between one and a specified maximum value.
- NHC - The heuristic combination is not comprised of just characters representing the low-level constructive heuristics, but integer-character pairs. The integer specifies the number of times the paired heuristic will be used to select an examination to schedule to the timetable.
- CEA - The heuristic combinations that make up the population in the evolutionary algorithm include all three representations, i.e. the representations used by FHC, VHC and NHC.

The paper presents just the results obtained by the four hyper-heuristics so a performance comparison with the existing low-level heuristics is not possible. Table VI presents the SDD values calculated to assess the performance of the four hyper-heuristics. Both CEA and VHC outperform the other hyper-heuristics.

TABLE VI
PERFORMANCE COMPARISON OF HYPER-HEURISTICS IN [39]

Hyper-Heuristic	***SDD***
CEA	8.39
VHC	88.31
NHC	88.45
FHC	89.61

Lehrbaum [50] presents a Level 4 selection perturbative hyper-heuristic, HAHA, to solve discrete optimisation problems for different domains in HyFlex, namely, Boolean satisfiability, one dimensional bin packing, personnel scheduling, flow shop scheduling, vehicle routing and the travelling salesman problem. Table VII presents SDD for HAHA and five other selection perturbative hyper-heuristics entered in the cross domain challenge, which also aim to achieve Level 4 generality across the six problem domains.

TABLE VII
PERFORMANCE COMPARISON OF SELECTION PERTURBATIVE HYPER-HEURISTICS FROM [50]

Hyper-Heuristic	*SDD*
AdaptHH	10.56
VNS-TW	39.5
HAHA	43.04
ML	43.81
SA-ILS	60.03
DynILS	61.09

Applying the SDD, we can see that in Table VII, AdaptHH outperforms the other hyper-heuristics. It is interesting to note that this hyper-heuristic was the winner of the cross domain challenge. VNS-TW performs better than the remaining hyper-heuristics with DynILS achieving the least level of generality across the problems.

The study presented in [53] compares the performance of two generation constructive hyper-heuristics to generate new low-level constructive heuristics for educational timetabling problems. AHH employs genetic programming to evolve arithmetic low-level heuristics. The second hyper-heuristic, HHH-GA10, uses a genetic algorithm to evolve hierarchical low-level constructive heuristics. The study evaluates both of these hyper-heuristics separately on examination timetabling and curriculum-based university course timetabling problems to

determine how they perform in the domain of educational timetabling. The Toronto and the ITC 2007 examination timetabling track benchmark sets have been used to assess the hyper-heuristics for examination timetabling and the ITC 2007 curriculum-based course timetabling track benchmark set for course timetabling. The study reveals that AHH performs better for examination timetabling and HHH-GA10 better for curriculum based course timetabling.

We examine the performance of both hyper-heuristics over both problems to assess their performance for the domain of educational timetabling. The SDD value for each of the hyper-heuristics over three benchmark sets, i.e. Toronto, ITC 2007 examination timetabling and ITC 2007 curriculum-based course timetabling, is listed in Table VIII. HHH-GA10 generalizes better for the domain of educational than AHH, performing well over all three benchmark sets.

TABLE VIII
PERFORMANCE COMPARISON OF GENERATION CONSTRUCTIVE HYPER-HEURISTICS FROM [53]

Hyper-Heuristic	*SDD*
AHH	78.9
HHH-GA10	62.93

D. Combining the Generality Measure with Other Criteria

In some instances the researcher may want to assess hyper-heuristic performance in terms of other criteria in addition to generality. For example, in addition to generality, the researcher may want to assess hyper-heuristic performance in terms of optimality and runtime. One approach that can be used is scalarization [55] such as taking the weighted sum of the different criteria. However, in order to take a weighted sum it is necessary that the values in the weighted sum are of the same dimension. Various normalization techniques can be used for this purpose [56]. The simplest is to divide the value by the known optimum in instances where this optimum is not zero. The most appropriate multiobjectve function to use is beyond the scope of the paper and future research will examine this further.

VI. CONCLUSION AND FUTURE RESEARCH

Hyper-heuristics aim at attaining generality in providing solutions to problems. Given that this is an emerging area existing performance measures from optimization such us optimality and ranking have been used to assess hyper-heuristic performance which is not ideal. It is also evident from the research in the field that different hyper-heuristics aim to achieve different levels of generality. The paper presents a taxonomy to classify hyper-heuristics in terms of the generality level it aims to attain. A performance measure to assess hyper-heuristic performance in terms of generality is presented and illustrated using three case studies.

The study presented in this paper has focused on hyper-heuristics for solving discrete optimisation problems. Future work will investigate applying the measure to continuous and multiobjective optimisation. Furthermore, the use of hyper-heuristics for the automated design of machine learning and search algorithms has been shown to be effective. Future extensions of this work will also investigate this performance measure for assessing the performance of hyper-heuristics to this rapidly developing area of hyper-heuristics. Future work will also identify the most appropriate multiobjective function for assessing the performance of hyper-heuristics using more the one criterion such as optimality and generality. Generality and optimality may be contradictory in some instances and need to be weighed carefully. Such a multiobjective evaluation function must be extensible to cater for new dimensions/objectives as the field of hyper-heuristics develops further.

REFERENCES

[1] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, and E. Ozcan, "Hyper-heuristics: A survey of the state of the art," *Journal of Operational Research Society*, vol. 64, pp. 1695–1724, 2013.

[2] P. Ross, "Hyper-heuristics," in *Search Methodologies*. Springer, 2014, pp. 611–638.

[3] R. Qu and E. K. Burke, "Hybridisations within a graph based hyper-heuristic framework for university timetabling problems," *Journal of Operational Research Society*, vol. 60, pp. 1273–1285, 2009.

[4] M. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using grammar-based genetic programming hyper-heuristic framework," *Memetic Computing*, vol. 1, pp. 205–219, 2009.

[5] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, pp. 177–192, 2007.

[6] H. Terashima-Marin, C. F. Zarate, P. Ross, and M. Valenzuela-Rendon, "A GA-based methd to produce generalized hyper-heuristics for the 2D-regular cutting stock problem," in *Proceedings of the 8th Annual Conference on Genetic Programming and Evolutionary Algorithms*. ACM, 2006, pp. 591–598.

[7] M. Misir, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe, "A new hyper-heuristic as a general problem solver: An implementation in hyflex," *Journal of Scheduling*, vol. 16, no. 3, pp. 291–311, 2013.

[8] R. Qu, E. Burke, B. McCollum, L. Merlot, and S. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling*, vol. 12, no. 1, pp. 55–89, 2009.

[9] G. Reinelt, "Tsplib-a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, November 1991.

[10] N. Pillay and R. Qu, *Hyper-Heuristics: Theory and Applications*, ser. Natural Computing Series. Springer, 2018.

[11] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, Ö. Ender, and W. J. R., "A classifiction of hyper-heuristic approaches: Revisted," in *Handbook of Metaheuristics*. Springer, 2019.

[12] M. Epitropakis and E. K. Burke, "Hyper-heuristics," in *Handbook of Metaheuristics*. Springer, 2018.

[13] R. Qu, E. K. Burke, and B. McCollum, "Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems," *European Journal of Operational Research*, vol. 198, no. 2, pp. 392–404, 2009.

[14] G. Kendall and N. M. Hussin, "An investigation of a tabu-search-based hyper-heuristic for examination timetabling," in *Multidisciplinary Scheduling: Theory and Applications*, E. Burke, S. Petrovic, and M. Gendreau, Eds. Springer, 2005, pp. 309–328.

[15] R. Raghavjee and N. Pillay, "A selection perturbative hyper-heuristic for solving the school timetabling problem," *ORiON*, vol. 31, no. 1, pp. 39–60, 2015.

[16] N. Pillay, "Evolving construction heuristics for the curriculum based university course timetabling problem," in *Proceedings of the IEEE 2016 Congress on Evolutionary Computation (CEC 2016)*, Y.-S. Ong, Ed. IEEE, 2016, pp. 4437–4443.

[17] A. S. Fukunaga, "Automated discovery of local search heuristics for satisfiability testing," *Evolutionary Computation*, vol. 16, no. 1, pp. 31–61, 2008.

[18] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, 1st ed. MIT, 1992.

[19] R. I. McKay, N. X. Hoai, P. Whigham, and M. O'Neill, "Grammar-based genetic programming: A survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3, pp. 365–396, 2010.

[20] M. O'Neill and C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Springer, 2003.

[21] B. McCollum, P. McMullan, B. Paechter, R. Lewis, A. Schaerf, L. DiGaspero, A. Parkes, R. Qu, and E. Burke, "Setting the research agenda in automated timetabling: The second international timetabling competition." *INFORMS Journal of Computing*, vol. 22, no. 1, pp. 120–130, 2008.

[22] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, no. 1, pp. 5–30, June 1996.

[23] A. Scholl, R. Klein, and C. Jurgens, "Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem," *Computers and Operations Research*, vol. 24, no. 7, pp. 626–645, 1997.

[24] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Grammatical evolution hyper-heuristic for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 840–861, September 2013.

[25] ——, "A dynamic multi-armed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 217–228, 2014.

[26] E. K. Burke, S. Petrovic, and R. Qu, "Case-based heuristic selection for timetabling problems," *Journal of Scheduling*, vol. 9, no. 2, pp. 115–132, 2006.

[27] A. S. Ferreira, A. T. R. Pozo, and R. A. Goncalves, "An ant colony based hyper-heuristic approach for the set covering problem," *Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 4, p. http://dx.doi.org/10.14201/ADCAIJ201541121, 2015.

[28] N. Pillay, "A study of evolutionary algorithm hyper-heuristics for the one-dimensional bin-packing problem," *South African Computer Journal*, vol. 48, pp. 31–40, June 2012.

[29] H. Terashima-Marin, J. Ortiz-Bayliss, P. Ross, and M. Valenzuela-Rendon, "Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problem," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*. ACM, 2008, pp. 571–578.

[30] P.-C. Chen, G. Kendall, and G. V. Berghe, "An ant based hyper-heuristic for the travelling tournament problem," in *IEEE Symposium on Computational Intelligence in Scheduling (SCIS '07)*, 2007, p. doi: 10.1109/SCIS.2007.367665.

[31] E. K. Burke, G. Kendall, M. Misir, and E. Özcan, "Monte carlo hyper-heuristics for examination," *Annals of Operations Research*, vol. 196, no. 1, pp. 73–90, 2012.

[32] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "A genetic programming hyper-heuristic approach for evolving two dimensional strip packing heuristics," *IEEE Transactions on Evolutionary Computation*, pp. 942–958, 2010.

[33] J. H. Drake, M. Hyde, K. Ibrahim, and E. Özcan, "A genetic programming hyper-heuristic for the multidimensional knapsack problem," *Kybernetes*, vol. 43, no. 9/10, pp. 1500–1511, 2014.

[34] K. Sim, E. Hart, and B. Paechter, "A lifelong learning hyper-heuristic method for bin packing," *Evolutionary Computation*, vol. 23, no. 1, pp. 37–67, 2015.

[35] M. Bader-El-Den and R. Poli, "Generating SAT local-search heuristics using a GP hyper-heuristic framework," in *Artificial Evolution: International Conference on Artificial Evolution*. Springer, 2008, pp. 37–49.

[36] E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A reinforcement learning-great-deluge hyper-heuristic for examination timetabling," *International Journal of Applied Metaheuristic Computing*, vol. 1, no. 1, pp. 39–59, January 2010.

[37] N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph colouring constructive hyper-heuristic for examination timetabling problems," *Applied Intelligence*, vol. 37, no. 1, pp. 1–11, July 2012.

[38] K. Sim and E. Hart, "Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, 2013, pp. 1549–1556.

[39] N. Pillay, "Evolving hyper-heuristics for the uncapacitated examination timetabling problem," *Journal of Operational Research Society*, vol. 63, no. 47-58, 2012.

[40] P. Ross, S. Schulenburg, J. G. Marin-Blazquez, and E. Hart, "Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*. Morgan Kaufman Publishers, 2002, pp. 942–948.

[41] K. Sim and E. Hart, "A combined generative and selective hyper-heuristic for the vehicle routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '16)*. ACM, 2016, pp. 1093–1100.

[42] A. Sosa-Ascencio, G. Ochoa, H. Terashima-Marin, and S. E. Conant-Pablos, "Grammar-based generation of variable-selection heuristics for constraint satisfaction problems," *Genetic Programming and Evolvable Machines*, vol. 17, no. 2, pp. 119–144, 2015.

[43] E. Lopez-Camacho, H. Terashima-Marin, P. Ross, and G. Ochoa, "A unified hyper-heuristic framework for solving bin packing problems," *Expert Systems with Applications*, vol. 41, pp. 6876–6889, 2014.

[44] H. Terashima-Marín, P. Ross, E. López-Camacho, and M. Valenzuela-Rendón, "Generalized hyper-heuristics for solving 2d regular and irregular packing problems," *Annals of Operations Research*, vol. 179, pp. 369–392, 2010.

[45] M. M, V. K, D. C. P, and V. B. G, "An investigation on the generality level of selection hyper-heuristics under different empirical conditions," *Applied Soft Computing*, vol. 13, no. 2013, pp. 3335–3353, 2013.

[46] G. Ochoa, M. Hyde, T. Curtois, J. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, and E. Burke, "Hyflex: A benchmark framework for cross-domain heuristic search," in *Evolutionary Computation in Combinatorial Optimization (EvoCOP 2012), Lecture Notes in Computer Sceince*, vol. 7245, 2012, pp. 136–147.

[47] C. Chan, F. Xue, W. Ip, and C. Cheung, "A hyper-heuristic inspired by pearl hunting," in *Learning and Intelligent Optimization, Lecture Notes in Computer Science*, Y. Hamadi and M. Schoenauer, Eds., vol. 7219, 2012, pp. 349–353.

[48] T. Cichowicz, M. Drozdowski, M. Frankiewicz, Grzegorz, F. Rytwiń, and J. Wasilewski, "Five phase genetic hive hyper-heuristics for the cross-domain search," in *Learning and Intelligent Optimization, Lecture Notes in Computer Science*, Y. Hamadi and M. Schoenauer, Eds., vol. 7219. Springer, 2012, pp. 354–339.

[49] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, pp. 451–470, 2003.

[50] A. Lehrbaum, "A new hyper-heuristic algorithm for cross-domain search problems," Master's thesis, Faculty of Informatics, Vienna University of Technology, 2011.

[51] M. Caramia, P. D. Olmo, and G. Italiano, "New algorithms for examination timetabling," in *4th International Workshop on Algorithm Engineering (WAE 2000), Leture Notes in Computer Science*, S. Naher and D. Wagner, Eds., vol. 1982, 2001, pp. 230–241.

[52] L. D. Gaspero and A. Schaerf, "Tabu search techniques for examination timetabling," in *Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes of Computer Science*, E. K. Burke and W. Erben, Eds., vol. 2079, 2000, pp. 104–117.

[53] N. Pillay and E. Özcan, "Automated generation of constructive ordering heuristics for education timetabling," *Annals of Operations Research*, pp. 1–28, September 2017.

[54] N. Pillay, "Evolving hyper-heuristics for a highly constrained examination timetabling problem," in *Proceedings of the international conference on the Practice and Theory of Automated Timetabling (PATAT 2010)*, 2010, pp. 336–346.

[55] N. Guantara, "A review of multi-objective optimization: Methods and applications," *Electrical and Electronic Engineering*, vol. 5, 2018.

[56] H. Mausser, "Normalization and other topics in multi-objective optimization," in *Proceedings of the Field-MITACS Industrial Problems Workshop*, 2006, pp. 89–101.