Assessing hyper-heuristic performance

Nelishia Pillay & Rong Qu

To cite this article: Nelishia Pillay & Rong Qu (2021) Assessing hyper-heuristic performance, Journal of the Operational Research Society, 72:11, 2503-2516, DOI: 10.1080/01605682.2020.1796538

To link to this article: https://doi.org/10.1080/01605682.2020.1796538

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY
2021, VOL. 72, NO. 11, 2503–2516
https://doi.org/10.1080/01605682.2020.1796538

THE OPERATIONAL RESEARCH SOCIETY

Taylor & Francis
Taylor & Francis Group

ORIGINAL ARTICLE

Check for updates

Assessing hyper-heuristic performance

Nelishia Pillay[a] and Rong Qu[b]

[a]Department of Computer Science, University of Pretoria, Pretoria, South Africa; [b]School of Computer Science, University of Nottingham, Nottingham, UK

ABSTRACT
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, i.e. the ability of a technique to produce good results for different problem instances or problems rather than the best results for some instances and poor results for others. Furthermore from existing literature in this area 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 based on a survey of the literature. 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 rather than optimality. 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 measures of other performance criteria, such as optimality, to assess hyper-heuristic performance on more than one criterion.

ARTICLE HISTORY
Received 8 August 2019
Accepted 12 July 2020

KEYWORDS
Hyper-heuristics; performance evaluation; generality; performance measure

1. 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 Burke et al. (2013). 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 Ross (2014).

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 Qu and Burke (2009). 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 Bader-El-Den et al. (2009); Burke et al. (2007). 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.

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 2 defines the terminology used in the paper. Section 3 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

CONTACT Nelishia Pillay npillay@cs.up.ac.za University of Pretoria, Pretoria, South Africa

problem Terashima-Marin, Zarate, Ross, and Valenzuela-Rendon (2006) to achieving generality across multiple problem domains Misir, Verbeeck, De Causmaecker, and Vanden Berghe (2013). Based on this overview Section 4 then presents a new taxonomy of generality and illustrates these using examples from the literature. Section 5 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 6 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 rather than optimality.

2. 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 Qu et al. (2009). Similarly, *eil76* Reinelt (1991) 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.

3. 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 Pillay and Qu (2018); Qu and Burke (2009). 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 Burke et al. (2013, 2019); Epitropakis and Burke (2018); Pillay and Qu (2018).

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 Burke et al. (2007); Qu et al. (2009). 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 Kendall and Hussin (2005).

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 Raghavjee and Pillay (2015).

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 Pillay (2016).

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, select a clause randomly, are combined with conditional operators to produce new heuristics Fukunaga (2008).

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 Koza (1992) and its variations, such as grammar-based genetic programming McKay et al. (2010) and grammatical evolution O'Neill and Ryan (2003), 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.

4. Levels of generality

In this section, we firstly present 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.

4.1. 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 McCollum et al. (2010) or the Faulkenauer benchmark set for the one-dimensional bin-packing problem Falkenauer (1996). The hyper-heuristic solves problem instances in a benchmark set for a single problem and hence its ability to generalise 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 Qu et al. (2009) and the ITC 2007 benchmark McCollum et al. (2010). Similarly, for the one-dimensional bin-packing domain the problems would include the Faulkenauer and Scholl benchmark sets Falkenauer (1996); Scholl et al. (1997). 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 generalise 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 Sabar et al. (2013, 2014). 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 generalise, with Level 1 being the lowest level of ability to generalise 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 generalise 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 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 5.

4.2. Level 1: Single problem-single benchmark

A fair amount of the studies on hyper-heuristics aims to achieve Level 1 generality, i.e. on a benchmark set of instances for a particular problem. One of the earlier studies Burke et al. (2006) employed a selection constructive hyper-heuristic to solve the examination timetabling problem for the Toronto benchmark. In Ferreira et al. (2015) a selection perturbative hyper-heuristic is implemented to solve the set covering problem using the OR-Library benchmark set. Similarly, in Pillay (2016) 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 Fukunaga (2008) 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 1 lists some examples of Level 1 generality hyper-heuristics.

Table 1. Examples of hyper-heuristics at level 1 generality.

Problem Domain	Type of Hyper-Heuristic	Benchmark Set
Examination timetabling Pillay (2012b)	Selection constructive	Toronto
Constraint satisfaction Terashima-Marin et al. (2008)	Selection constructive	Generated
Travelling tournament problem Chen et al.,(2007)	Selection perturbative	Easton
Set covering problem Ferreira et al.,(2015)	Selection perturbative	ORLib
Examination timetabling Burke et al.,(2012)	Selection perturbative	Toronto
Examination timetabling Bader-El-Den et al.,(2009)	Generation constructive	Toronto
2D strip packing Burke et al.,(2010)	Generation constructive	Generated
Multidimensional knapsack problem Drake et al.,(2014)	Generation constructive	ORLib
One dimensional bin packing Sim et al.,(2015)	Generation constructive	ORLib
3-SAT Bader-El-Den & Poli,(2008)	Generation perturbative	SatLib
Combinatorial test generation Ahmed et al.,(2020)	Selection perturbative	Train control management system

4.3. 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.

Table 2. Examples of hyper-heuristics at level 2 generality.

Problem Domain	Type of Hyper-Heuristics	Benchmark Sets
One dimensional bin packing Pillay,(2012a)	Selection constructive	Scholl and Faulkenauer
One dimensional bin packing Ross et al. (2002)	Selection constructive	Faulkenauer and Technische Universitat Darmstadt
2D regular stock cutting Terashima-Marin et al. (2006)	Selection constructive	Generated, ORLib, Martello and Vigo Berkey and Wang
Examination timetabling Sabar et al.,(2012)	Selection constructive	Toronto and ITC 2007
School timetabling Raghavjee & Pillay, (2015)	Selection perturbative	Abramson, Valouxis, Beligiannis, South African primary school South Africa high school
Examination Timetabling Özcan et al., (2010)	Selection perturbative	Toronto and Yeditepe
Vehicle routing Sim & Hart, (2016)	Generation constructive	Solomon and Sim and Hart
Constraint Satisfaction Sosa-Ascencio et al. (2016)	Generation	Generated, RLFAP-graphs
SAT Fukunaga,(2008)	Generation perturbative	SatLib and Gottlieb
Software clustering Kumari & Srinivas,(2016)	Selection perturbative	Mancoridis et al. and Barros

In the study conducted by Terashima-Marín et al. (2006), a selection constructive hyper-heuristic is used to create solutions to the 2 D 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 Özcan et al. (2010) 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. Burke et al. (2010) 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. Sabar et al. (2012), a selection constructive hyper-heuristic is implemented to solve the examination timetabling problem for the Toronto and ITC 2007 benchmark sets. In Raghavjee and Pillay (2015) a selection perturbative hyper-heuristic is used to solve problem instances in various benchmark sets for the school timetabling problem. Sim and Hart Sim and Hart (2013) 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 2.

4.4. 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 Lopez-Camacho et al. (2014), 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., and (2010) employ a selection perturbative hyper-heuristic to solve packing problems. The hyper-heuristic solves both two dimensional regular and irregular problems. Burke et al. Burke et al. (2007) 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. M, K, P, & G (2013) 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 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 3 presents examples of Level 3 generality hyper-heuristics.

Table 3. Examples of hyper-heuristics for level 3 generality.

Problem Domain	Problems	Type of Hyper-Heuristics	Sets
Educational timetabling Burke et al.,(2006)	Examination timetabling and course timetabling	Selection constructive	Generated
Offline packing problems Lopez-Camacho et al.,(2014)	1D single bin, 2D regular and 2D irregular single bin	Selection constructive	Generated, Scholl Washer and Faulkenauer
Educational timetabling Burke et al. (2007)	Examination timetabling and course timetabling	Selection constructive	Toronto Metaheuristic network
Educational timetabling Qu & Burke,(2009)	Examination timetabling and course timetabling	Selection constructive	Toronto Metaheuristic network
2D stock cutting Terashima-Marín et al.,(2010)	Regular and irregular	Selection constructive	Generated, ORLib Martello and Vigo Berkey and Wang
Combinatorial interaction testing Jia et al. (2015)	Synthetic models with and without constraints	Selection pertubative	Garvin, Cohen Seagull, Kuhn Siemens
Interaction *t-way* testing Zamli et al. (2016)	Covering arrays and mixed covering arrays	Selection pertubative	Garvin, Cohen Ahmed

Table 4. Examples of hyper-heuristics at level 4 generality.

Problem Domains	Type of Hyper-Heuristics	Benchmark Sets
Nurse rostering Course timetabling Burke et al.,(2003)	Selection perturbative	Aickelin and Downsland Metaheuristic network
Examination timetabling Dynamic vehicle routing Burke et al.,(2003)	Selection perturbative	ITC 2007, Taillard, Christofides, Fischer
Examination timetabling Vehicle routing Sabar et al.,(2013)	Generation perturbative	ITC 2007, Golden, Christofides
6 CHeSC problem domains Misir et al.,(2013)	Selection perturbative	CHeSC benchmark sets
6 CHeSC problem domains Chan et al.,(2012)	Selection perturbative	CHeSC benchmark sets
6 CHeSC problem domains Cichowicz et al.,(2012)	Selection perturbative	CHeSC benchmark sets
6 CHeSC problem domains Lehrbaum,(2011)	Selection perturbative	CHeSC subsets of benchmark sets

4.5. 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 Ochoa et al. (2012) 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 Misir et al. (2013), 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. Chan et al. (2012) takes an analogy from pearl hunting to explore the space of low-level pertubative heuristics. Cichowicz et al. Cichowicz et al. (2012) 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 4.

5. 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.

5.1. 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 Ochoa et al. (2012). 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 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. $j = 1, \dots, n$.
```

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.

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

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

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

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

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 generalised (rather than optimised) well over the set of problem instances.

We illustrate both ranking(*RAN*) and *SDD* using two hyper-heuristics in Burke et al. (2007), 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 Caramia et al. (2001); Gaspero and Schaerf (2000):

- GHH - The tabu search hyper-heuristic in Burke et al. (2007).
- Multistage - The multistage version of GHH in Burke et al. (2007).
- SB - The sequential allocation approach with backtracking in Caramia et al. (2001) for examination timetabling problems. This approach has

Table 5. 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

produced the best objective values for more problem instances in the Toronto benchmark set than any other approach applied to this set.

- TS - In Gaspero and Schaerf (2000) 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 5 displays the *RAN* and *SDD* values for the four approaches to the 11 problem instances in the Toronto benchmark set.

From Table 5 it can be seen that the sequential allocation approach with backtracking (SB) by Caramia et al. (2001) 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 generalise 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 generalises better over the problem set than SB. Box plots for the rank values and the SDD values for the four methods are depicted in Figures 1 and 2 respectively. As these measures have different scales they have not been plotted on the same chart.

5.2. 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 3. Then performance evaluation for the four levels of generality is explained.

5.2.1. 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.

5.2.2. 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.

5.2.3. 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.

5.2.4. 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

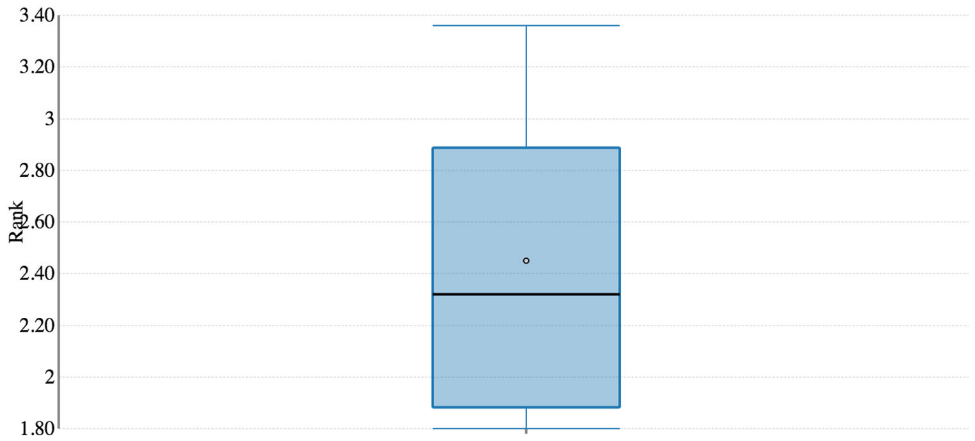

Figure 1. Box plot for the rank values.

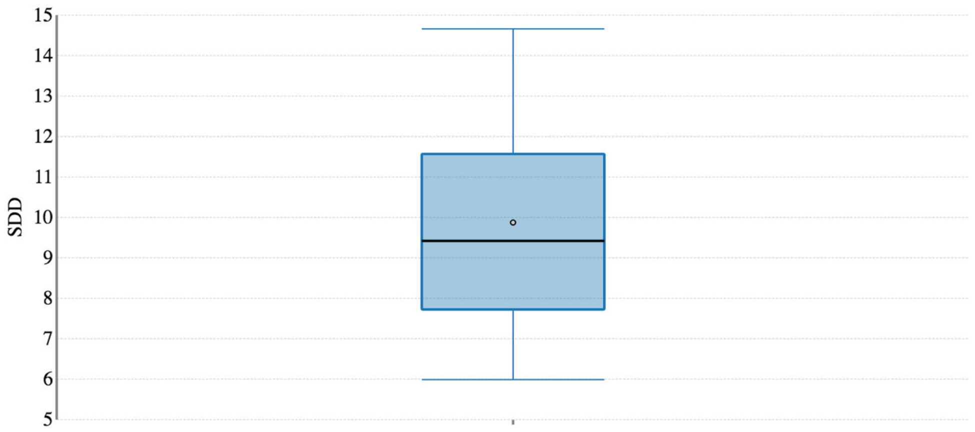

Figure 2. Box plot for the SDD values.

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.

5.3. Case studies

This section illustrates and demonstrates the use of the *SDD* performance measure established in Section 5.1 in assessing the performance of hyper-heuristics in three studies from the literature, namely:

- A selection constructive Level 1 hyper-heuristic Pillay (2012a)
- A selection perturbative Level 4 hyper-heuristic Lehrbaum (2011)

A generation constructive Level 3 hyper-heuristic Pillay and Özcan (2019)40The study in Pillay (2010, 2012a) 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-

Table 6. Performance comparison of hyper-heuristics in Pillay (2012a).

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

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 6 presents the *SDD* values calculated to assess the performance of the four hyper-heuristics. Both CEA and VHC outperform the other hyper-heuristics.

Lehrbaum Lehrbaum (2011) 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 7 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.

Applying the *SDD*, we can see that in Table 7, 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 Pillay and Özcan (2019) 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.

Table 7. Performance comparison of selection perturbative hyper-heuristics from Lehrbaum (2011).

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

Table 8. Performance comparison of generation constructive hyper-heuristics from Pillay and Özcan (2019).

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

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 8. HHH-GA10 generalizes better for the domain of educational than AHH, performing well over all three benchmark sets.

5.4. 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 Guantara (2018) 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 Mausser (2006). The simplest is to divide the value by the known optimum in instances where this optimum is not zero. The most appropriate normalization technique and multiobjectve function to use is beyond the scope of the paper and future research will examine this further. However, in this section we will look at an example normalization technique and example functions for combining the SDD (representing generality) and ranking (representing optimality).

Table 9. Normalized *RAN* and *SDD*.

Approach	*RAN*	*SDD*	*RAN-N*	*SDD-N*
SB	1.8	14.66	0.07	0
GHH	1.91	5.99	0.60	0.27
Multistage	2.73	8.3	0	1
TS	3.36	10.54	1	0.53

Table 10. Example functions.

Approach	Min	Sum	Weight-G	Weight-O
SB	0	1	0.6	0.4
GHH	0	0.07	0.03	0.04
Multistage	0.27	0.86	0.4	46
TS	0.52	1.52	0.71	0.81

Min-max normalization Freedman et al. (2007) can be used to normalize SDD and the average or sum of ranks to be in the range [0, 1]. The lowest value x_{min}, i.e. the minimum, is mapped to a value of zero. The maximum value x_{max} is mapped to a value of 1. A value x between the minimum and maximum values is mapped to a value between 0 and 1 using the following formula:

$$\frac{x - x_{min}}{x_{max} - x_{min}} \tag{5}$$

Table 9 depicts the normalized *SDD* and *RAN* values for the examination timetabling example that we did a comparison of *SDD* and *RAN* for above. The values have been normalized by min-max normalization and labelled as *RAN-N* and *SDD-N*.

Table 10 examines the effect of different functions to combine both *SDD-N* and *RAN-N*. If both generality and optimality are equally important in solving the problem at hand, then taking the minimum of *RAN-N* and *SDD-N* is a possible combining function. Using the Min function the order of the methods in terms of performance is SB and GHH, Multistage and then TS. Both SB and GHH are tied as the best performing methods. This makes sense if both generality and optimality are important, as SB is the best performing method if performance is measured in terms of optimality and GHH is the best performing method if performance is measured in terms of generality. The Multistage hyper-heuristic does not perform as well as SB and GHH using the Min function, but outperforms TS which is what we want if optimality and generality are equally important.

If the Sum of *SDD-N* and *RAN-N* is taken, the order of the methods in terms of performance is GHH, Multistage, SB and TS. In this instance, methods that achieved better generality appear to be favoured over methods that achieved better optimality with the exception of SB and TS, where SB performs better than TS if the Sum is taken instead of using just *SDD*. Hence, the Sum is a combined measure of optimality and generality performance (Figure 3).

Taking the weighted sum of the normalized values will allow for the importance of generality and optimality to be controlled on a scale from 0 to a 100 percent. In Table 10 Weight-G represents the weighted sum with generality being given a little more importance than optimality, with generality contributing 60% and optimality 40%. This has the same effect as taking the sum but with a smaller difference between SB and TS. Weight-O in Table 10 represents the weighted sum with optimality given more importance (60%) than generality (40%). Given this weighting the order of the methods in terms of performance is GHH, SB, Multistage and TS. It is also interesting to note that irrespective of ~~the function or the weighting GHH always performs the best.~~

An alternative function is a comparative function to decide which of two methods performs better. An example of such a function, which we will refer to as the *Pareto* function, is depicted in algorithm 5.4.

Algorithm 2. Pareto function

```
1: if $((SDD\text{-}N_{m_1}<SDD\text{-}N_{m_2})\&\&(RAN\text{-}N_{m_1}<RAN\text{-}N_{m_2}))$ then
2:     $m_1$ outperforms $m_2$
3: end if
4: if $((SDD\text{-}N_{m_1}<SDD\text{-}N_{m_2})\&\&(RAN\text{-}N_{m_1}<RAN\text{-}N_{m_2}))$ then
5:     $m_1$ outperforms $m_2$
6: end if
7: if $((SDD\text{-}N_{m_1}<SDD\text{-}N_{m_2})\&\&(RAN\text{-}N_{m_1}>RAN\text{-}N_{m_2}))$ then
8:     if $RAN\text{-}N_{m_1}<SDD\text{-}N_{m_2}$ then
9:         $m_1$ performs better than $m_2$
10:    else
11:        $m_2$ performs better than $m_1$
12:    end if
13: end if
14: if $((SDD\text{-}N_{m_1}>SDD\text{-}N_{m_2})\&\&(RAN\text{-}N_{m_1}<RAN\text{-}N_{m_2}))$ then
15:    if $SDD\text{-}N_{m_1}<RAN\text{-}N_{m_2}$ then
16:        $m_1$ performs better than $m_2$
17:    else
18:        $m_2$ performs better than $m_1$
19:    end if
20: end if
```

If both *SDD-N* and *RAN-N* are lower for one of the two methods being compared then this method is the better performing method. However, if *SDD-N* is lower for one method and *RAN-N* is lower for the other method, then the method with the lower greater value is the better performer. Table 11 performs these comparisons for the SB, GHH, Multistage and TS methods. In the comparison of GHH and Multistage and TS, it is clear that GHH outperforms both these methods as it has a lower *SDD-N* and *RAN-N* value in both instances. Similarly, Multistage outperforms TS with a lower *SDD-N* and *RAN-N* value. In the comparison of GHH and SB, GHH has a lower *SDD-N* value and SB a lower *RAN-N* value. However, the

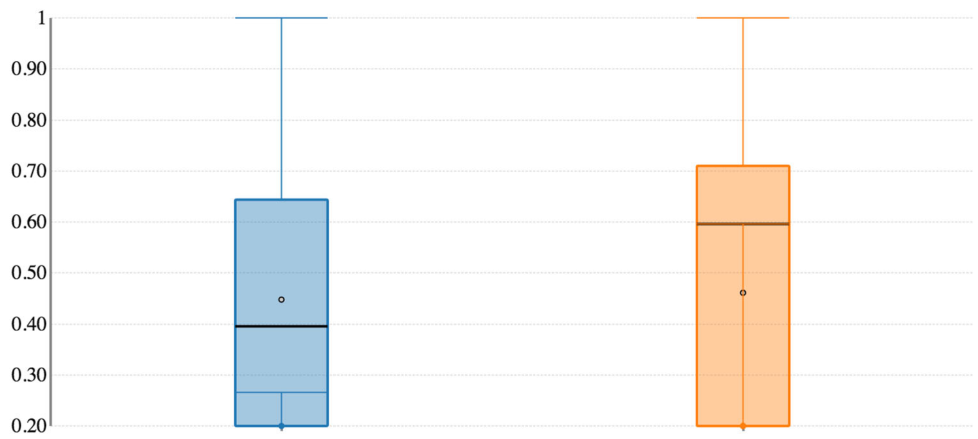

Figure 3. Box plot for the normalized SDD and Rank Values: SDD (first plot), Rank (second plot).

Table 11. Illustration of the Pareto function.

Comparison	Better Performer
GPHH + Multistage	GPHH
GPHH + SB	GPHH
GPHH + TS	GPHH
Multistage + SB	Multistage
Multistage + TS	Multistage
SB + TS	SB

RAN-N value for GHH is lower than the *SDD-N* value for SB, hence GHH outperforms SB. Similarly, Multistage outperforms SB and SB outperforms TS.

Future work will investigate other options for normalizing *SDD* and *RAN* and alternative functions for combining these measures including any additional measures that a particular study may want to assess hyper-heuristic performance on, e.g. computational effort or runtimes.

5.5. Statistical significance

The SDD has been presented as an alternative to assessing hyper-heuristic performance intstead of optimality using the objective function. In the previous section examples of how the SDD can be combined with other measures assessing hyper-heuristic performance such as optimality has been highlighted. In assessing the statistical significance of comparisons of different hyper-heuristics assessed according to SDD or a combination of SDD and other performance measures such as ranking, the same approach can be taken as has been done when using the objective function to compare hyper-heuristic performance, namely, a minimum of thirty runs are performed for each approach and the value of the measure for the *n* runs performed form the series for a nonparametric test, such as the Friedman test, to assess statistical significance of the comparison.

6. 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.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Nelishia Pillay http://orcid.org/0000-0003-3902-5582
Rong Qu http://orcid.org/0000-0001-8318-7509

References

Ahmed, B., Enolu, E., Afzal, W. & Zamli. (2020). An evaluation of monte carlo-based hyper-heuristic for interaction testing of industrial embedded software applications. *Soft Computing*, https://doi.org/10.1007/s00500-020-04769-z.

Bader-El-Den, M., & Poli, R. (2008). *Generating SAT local-search heuristics using a GP hyper-heuristic framework* [Paper presentation]. Artificial Evolution: International Conference on Artificial Evolution (pp. 37–49). Springer.

Bader-El-Den, M., Poli, R., & Fatima, S. (2009). Evolving timetabling heuristics using grammar-based genetic programming hyper-heuristic framework. *Memetic Computing*, *1*(3), 205–219. https://doi.org/10.1007/s12293-009-0022-y

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, *64*(12), 1695–1724. https://doi.org/10.1057/jors.2013.71

Burke, E. K., Hyde, M., Kendall, G., & Woodward, J. (2010). A genetic programming hyper-heuristic approach for evolving two dimensional strip packing heuristics. *IEEE Transactions on Evolutionary Computation*, *14*(6), 942–958. https://doi.org/10.1109/TEVC.2010.2041061

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ender, Ö., & R, W. J. (2019). A classifiction of hyper-heuristic approaches: Revisted. In *Handbook of metaheuristics*. Springer.

Burke, E. K., Kendall, G., & Soubeiga, E. (2003). A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, *9*(6), 451–470. https://doi.org/10.1023/B:HEUR.0000012446.94732.b6

Burke, E. K., Kendall, G., Misir, M., & Özcan, E. (2012). Monte carlo hype-heuristics for examination. *Annals of Operations Research*, *196*(1), 73–90. https://doi.org/10.1007/s10479-010-0782-2

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, *176*(1), 177–192. https://doi.org/10.1016/j.ejor.2005.08.012

Burke, E. K., Petrovic, S., & Qu, R. (2006). Case-based heuristic selection for timetabling problems. *Journal of Scheduling*, *9*(2), 115–132. https://doi.org/10.1007/s10951-006-6775-y

Caramia, M., Olmo, P. D., & Italiano, G. (2001). New algorithms for examination timetabling. In S. Naher & D. Wagner (Eds.), *4th international workshop on algorithm engineering (WAE 2000), Lecture Notes in Computer Science* (Vol. 1982, pp. 230–241). Springer.

Chan, C., Xue, F., Ip, W., & Cheung, C. (2012). A hyper-heuristic inspired by pearl hunting. In Y. Hamadi & M. Schoenauer (Eds.), *Learning and intelligent optimization, Lecture Notes in Computer Science* (Vol. 7219, pp. 349–353). Springer.

Chen, P.-C., Kendall, G., & Berghe, G. V. (2007). *An ant based hyper-heuristic for the travelling tournament problem* [Paper presentation]. *IEEE symposium on computational intelligence in scheduling (scis '07)* (p. https://doi.org/10.1109/SCIS.2007.367665)[10.1109/SCIS.2007.367665]

Cichowicz, T., Drozdowski, M., Frankiewicz, M., Grzegorz, Rytwiń, F., & Wasilewski, J. (2012). Five phase genetic hive hyper-heuristics for the cross-domain search. In Y. Hamadi & M. Schoenauer (Eds.), *Learning and intelligent optimization, Lecture Notes in Computer Science* (Vol. 7219, pp. 354–439). Springer.

Drake, J. H., Hyde, M., Ibrahim, K., & Özcan, E. (2014). A genetic programming hyper-heuristic for the multidimensional knapsack problem. *Kybernetes*, *43*(9/10), 1500–1511. https://doi.org/10.1108/K-09-2013-0201

Epitropakis, M., & Burke, E. K. (2018). Hyper-heuristics. In *Handbook of metaheuristics*. Springer.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, *2*(1), 5–30. https://doi.org/10.1007/BF00226291

Ferreira, A. S., Pozo, A. T. R., & Goncalves, R. A. (2015). An ant colony based hyper-heuristic approach for the set covering problem. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *4*(1), 1. https://doi.org/10.14201/ADCAIJ201541121

Freedman, D., Pisani, R., & Purves, R. (2007). *Statistics: Fourth international student edition*. W.W. Norton and Company.

Fukunaga, A. S. (2008). Automated discovery of local search heuristics for satisfiability testing. *Evolutionary Computation*, *16*(1), 31–61. https://doi.org/10.1162/evco.2008.16.1.31

Gaspero, L. D., & Schaerf, A. (2000). Tabu search techniques for examination timetabling. In E. K. Burke & W. Erben (Eds.), *Selected papers from the 3rd international conference on the practice and theory of automated timetabling, Lecture Notes in Computer Science* (Vol. 2079, pp. 104–117). Springer.

Guantara, N. (2018). A review of multi-objective optimization: Methods and applications. *Electrical and Electronic Engineering*, *5*.doi:10.1080/2331916.2018

Jia, Y., Cohen, M., Harman, M., & Petke, J. (2015). Learning combinatorial interaction test generation strategies using hyperheuristic search. In *Proceedings of the 37th international conference on software engineering (ICSE 2015)* (Vol. 1, p. 540–550).

Kendall, G., & Hussin, N. M. (2005). An investigation of a tabu-search-based hyper-heuristic for examination timetabling. In E. Burke, S. Petrovic, & M. Gendreau (Eds.), *Multidisciplinary scheduling: Theory and applications* (pp. 309–328). Springer.

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection* (1st ed.). MIT.

Kumari, A. C., & Srinivas, K. (2016). Hyper-heuristic approach for multi-objective software module clustering. *Journal of Systems and Software*, *117*, 384–401. https://doi.org/10.1016/j.jss.2016.04.007

Lehrbaum, A. (2011). *A new hyper-heuristic algorithm for cross-domain search problems* [Unpublished master's thesis]. Faculty of Informatics, Vienna University of Technology.

Lopez-Camacho, E., Terashima-Marin, H., Ross, P., & Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, *41*(15), 6876–6889. https://doi.org/10.1016/j.eswa.2014.04.043

Mausser, H. (2006). Normalization and other topics in multi-objective optimization. In *Proceedings of the Field-Mitacs Industrial Problems Workshop* (p. 89–101).

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., & Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, *22*(1), 120–130. https://doi.org/10.1287/ijoc.1090.0320

McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., & O'Neill, M. (2010). Grammar-based genetic programming: A survey. *Genetic Programming and Evolvable Machines*, *11*(3–4), 365–396. https://doi.org/10.1007/s10710-010-9109-y

Misir, M., Verbeeck, K., Causmaecker, P. D., & Berghe, G. V. (2013). A new hyper-heuristic as a general problem solver: An implementation in hyflex. *Journal of Scheduling*, *16*(3), 291–311. https://doi.org/10.1007/s10951-012-0295-8

Mısır, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G. (2013). An investigation on the generality level of selection hyper-heuiristics under different empirical conditions. *Applied Soft Computing*, *13*(7), 3335–3353. https://doi.org/10.1016/j.asoc.2013.02.006

O'Neill, M., & Ryan, C. (2003). *Grammatical evolution: Evolutionary automatic programming in an arbitrary language*. Springer.

Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J., Walker, J., Gendreau, M., & Burke, E. (2012). Hyflex: A benchmark framework for cross-domain heuristic search. In *Evolutionary computation in combinatorial optimization (EVOCOP 2012), Lecture Notes in Computer Science* (Vol. 7245, pp. 136–147). Springer.

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

Pillay, N. (2010). 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)* (p. 336–346).

Pillay, N. (2012a). Evolving hyper-heuristics for the uncapacitated examination timetabling problem. *Journal of the Operational Research Society*, *63*(1), 47–58. (https://doi.org/10.1057/jors.2011.12

Pillay, N. (2012b). A study of evolutionary algorithm hyper-heuristics for the one-dimensional bin-packing problem. *South African Computer Journal*, *48*, 31–40. https://doi.org/10.18489/sacj.v48i1.87

Pillay, N. (2016). Evolving construction heuristics for the curriculum based university course timetabling problem. In Y.-S. Ong (Ed.), *Proceedings of the IEEE 2016 congress on evolutionary computation (cec 2016)* (pp. 4437–4443). IEEE.

Pillay, N., & Özcan, E. (2019). Automated generation of constructive ordering heuristics for education timetabling. *Annals of Operations Research*, *275*(1), 181–128. https://doi.org/10.1007/s10479-017-2625-x

Pillay, N., & Qu, R. (2018). *Hyper-heuristics: Theory and applications*. Springer.

Qu, R., & Burke, E. K. (2009). Hybridisations within a graph based hyper-heuristic framework for university timetabling problems. *Journal of the Operational Research Society*, *60*(9), 1273–1285. https://doi.org/10.1057/jors.2008.102

Qu, R., Burke, E. K., & McCollum, B. (2009). Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, *198*(2), 392–404. https://doi.org/10.1016/j.ejor.2008.10.001

Qu, R., Burke, E., McCollum, B., Merlot, L., & Lee, S. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, *12*(1), 55–89. https://doi.org/10.1007/s10951-008-0077-5

Raghavjee, R., & Pillay, N. (2015). A selection perturbative hyper-heuristic for solving the school timetabling problem. *ORiON*, *31*(1), 39–60. https://doi.org/10.5784/31-1-158

Reinelt, G. (1991). Tsplib-a traveling salesman problem library. *ORSA Journal on Computing*, *3*(4), 376–384. https://doi.org/10.1287/ijoc.3.4.376

Ross, P., Schulenburg, S., Marin-Blazquez, J. G., & Hart, E. (2002). Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In *Proceedings of the genetic and evolutionary computation conference (GECCO 2002)* (pp. 942–948). Morgan Kaufman Publishers.

Ross, P. (2014). Hyper-heuristics. In *Search methodologies* (pp. 611–638). Springer.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2013). Grammatical evolution hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, *17*(6), 840–861. https://doi.org/10.1109/TEVC.2013.2281527

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2014). A dynamic multi-armed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, *45*(2), 217–228. https://doi.org/10.1109/TCYB.2014.2323936

Sabar, N. R., Ayob, M., Qu, R., & Kendall, G. (2012). A graph colouring constructive hyper-heuristic for examination timetabling problems. *Applied Intelligence*, *37*(1), 1–11. https://doi.org/10.1007/s10489-011-0309-9

Scholl, A., Klein, R., & Jurgens, C. (1997). Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers and Operations Research*, *24*(7), 626–645. https://doi.org/10.1016/S0305-0548(96)00082-2

Sim, K., & Hart, E. (2013). *Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model* [Paper presentation]. Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (pp. 1549–1556). https://doi.org/10.1145/2463372.2463555

Sim, K., & Hart, E. (2016). *A combined generative and selective hyper-heuristic for the vehicle routing problem* [Paper presentation]. Proceedings of the Genetic and Evolutionary Computation Conference (Gecco '16) (pp. 1093–1100). ACM. https://doi.org/10.1145/2908812.2908942

Sim, K., Hart, E., & Paechter, B. (2015). A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation*, *23*(1), 37–67. https://doi.org/10.1162/EVCO_a_00121

Sosa-Ascencio, A., Ochoa, G., Terashima-Marin, H., & Conant-Pablos, S. E. (2016). Grammar-based generation of variable-selection heuristics for constraint satisfaction problems. *Genetic Programming and Evolvable Machines*, *17*(2), 119–144. https://doi.org/10.1007/s10710-015-9249-1

Terashima-Marín, H., Ross, P., Farías-Zárate, C. J., López-Camacho, E., & Valenzuela-Rendón, M. (2010). Generalized hyper-heuristics for solving 2D regular and irregular packing problems. *Annals of Operations Research*, *179*(1), 369–392. https://doi.org/10.1007/s10479-008-0475-2

Terashima-Marín, H., Zarate, C. F., Ross, P., & Valenzuela-Rendon, M. (2006). *A GA-based methd to produce generalized hyper-heuristics for the 2D-regular cutting stock problem* [Paper presentation]. Proceedings of the 8th Annual Conference on Genetic Programming and Evolutionary Algorithms (pp. 591–598). ACM. https://doi.org/10.1145/1143997.1144102

Zamli, K. Z., Alkazemi, B. Y., & Kendall, G. (2016). A Tabu Search hyper-heuristic strategy for t-way test suite generation. *Applied Soft Computing*, *44*, 57–74. https://doi.org/10.1016/j.asoc.2016.03.021