Journal of the Operational Research Society (2013) 64, 1695–1724 © 2013 Operational Research Society Ltd. All rights reserved. 0160-5682/13

www.palgrave-journals.com/jors/

Hyper-heuristics: a survey of the state of the art

Edmund K Burke[1], Michel Gendreau[2], Matthew Hyde[3], Graham Kendall[4], Gabriela Ochoa[1*], Ender Özcan[4] and Rong Qu[4]

[1]University of Stirling, Scotland, UK; [2]University of Montreal, Montreal, Canada; [3]University of East Anglia, Norwich, UK; and [4]University of Nottingham, Nottingham, UK

Hyper-heuristics comprise a set of approaches that are motivated (at least in part) by the goal of automating the design of heuristic methods to solve hard computational search problems. An underlying strategic research challenge is to develop more generally applicable search methodologies. The term hyper-heuristic is relatively new; it was first used in 2000 to describe *heuristics to choose heuristics* in the context of combinatorial optimisation. However, the idea of automating the design of heuristics is not new; it can be traced back to the 1960s. The definition of hyper-heuristics has been recently extended to refer to *a search method or learning mechanism for selecting or generating heuristics to solve computational search problems*. Two main hyper-heuristic categories can be considered: heuristic *selection* and heuristic *generation*. The distinguishing feature of hyper-heuristics is that they operate on a search space of heuristics (or heuristic components) rather than directly on the search space of solutions to the underlying problem that is being addressed. This paper presents a critical discussion of the scientific literature on hyper-heuristics including their origin and intellectual roots, a detailed account of the main types of approaches, and an overview of some related areas. Current research trends and directions for future research are also discussed.

Journal of the Operational Research Society (2013) **64,** 1695–1724. doi:10.1057/jors.2013.71
Published online 10 July 2013

Keywords: Hyper-heuristics; evolutionary computation; metaheuristics; machine learning; combinatorial optimisation; scheduling

1. Introduction

Despite the success of heuristic methods and other search techniques in solving real-world computational search problems, there are still some difficulties in terms of easily applying them to newly encountered problems, or even new instances of similar problems. These difficulties arise mainly from the significant range of parameter or algorithm choices involved when using this type of approach and the lack of guidance as to how to select them. In addition, the scientific community's level of understanding of why different heuristics work effectively (or not) in different situations does not facilitate simple choices of which approach to use in which situation. Another drawback of current techniques is that state-of-the-art approaches for real-world problems tend to represent bespoke problem-specific methods which are expensive to develop and maintain. A key motivating goal for this area (but by no means the only one) is the challenge of automating the design and tuning of heuristic methods to solve hard computational search problems (Burke *et al*, 2003a, 2009; Ross, 2005). The main idea is to develop algorithms that are more generally applicable than many of the current implementations of search methodologies. When using hyper-heuristics, we are attempting to find the right method or sequence of heuristics in a given situation rather than trying to solve the problem directly. Hyper-heuristics could be regarded as 'off-the-peg' methods as opposed to 'made-to-measure' techniques. Therefore, an important goal is to design generic methods, which should produce solutions of acceptable quality, based on a set of easy-to-implement low-level heuristics. A hyper-heuristic can be seen as a (high-level) methodology which, when given a particular problem instance or class of instances, and a number of low-level heuristics (or its components), automatically produces an adequate combination of the provided components to effectively solve the given problem(s). The overall goal of this paper is to analyse and discuss the hyper-heuristic literature to date.

The term hyper-heuristics was first used in a peer-reviewed conference paper in 2001 (Cowling *et al*, 2000). The ideas in this first paper were further developed and applied to scheduling problems in Cowling *et al* (2001, 2002a, b, c). In these publications, a hyper-heuristic[1] was considered to be a high-level approach that, given a

Correspondence: Gabriela Ochoa, Department of Computing Sciences and Mathematics, School of Natural Sciences, University of Stirling, Room 4B104, Cottrell Building, Stirling, Scotland FK9 4LA, UK.
E-mails: gabriela.ochoa@cs.stir.ac.uk

[1]In these first appearances, the term was often written without a hyphen (ie *hyperheuristics*). Throughout this article, we have chosen to use the most widely used spelling of the term (with the hyphen).

particular problem instance and a number of low-level heuristics, can select and apply an appropriate low-level heuristic at each decision point. An earlier single appearance of the term can be found in a technical report (Denzinger *et al*, 1996), where it was used in a different context, to describe an approach that combines a range of Artificial Intelligence algorithms for automated theorem proving. This report only uses the term once and does not propose a definition of hyper-heuristics. The basic idea of automating the design and/or selection of heuristics is, however, much older. It can be traced right back to the early 1960s, as we will discuss in Section 2.

A number of introductory tutorial and review book chapters on hyper-heuristics have been published over the last few years. The first one appeared in 2003 (Burke *et al*, 2003a), where the authors discussed the idea of hyper-heuristics and stressed one of the key objectives; namely, to raise the level of generality at which optimisation systems can operate. The chapter also gives a brief history of the area and discusses in detail some representative examples published at the time. A tutorial article was later published by Ross (2005), which not only gives useful guidelines for implementing a hyper-heuristic approach, but it also discusses a number of relevant research issues and identifies promising application domains. A more recent publication (Chakhlevitch and Cowling, 2008) provides a classification and discussion of recent developments in hyper-heuristics, with an emphasis on real-world complex applications. The chapter presents three useful criteria as a definition of these approaches, which we rephrase here: a hyper-heuristic is (i) a higher level heuristic that manages a set of low-level heuristics, (ii) it searches for a good method to solve the problem rather than for a good solution, and (iii) it uses only limited problem-specific information. The authors regard the last criteria as the most crucial one. A recent overview and tutorial chapter (Burke *et al*, 2009) discusses methodologies to *generate* new heuristics from a set of potential heuristic components, in which Genetic Programming plays a prominent role. The chapter includes a detailed description of the steps needed to apply this approach, some representative case studies, a brief literature review of related work, and a discussion of relevant issues of this class of hyper-heuristic. Finally, Burke *et al* (2010d) present an overview of previous categorisations of hyper-heuristics and provide a unified classification and definition that captures the work that is being undertaken in this field. A hyper-heuristic is defined, there, as 'a search method or learning mechanism for selecting or generating heuristics to solve computational search problems'.

The next section discusses the intellectual roots and early hyper-heuristic approaches. Section 3 discusses our proposal for classifying hyper-heuristics (Burke *et al*, 2010d). Following this classification, we then provide a critical discussion of the scientific literature covering *heuristic selection* methodologies (Section 4), and *heuristic generation* methodologies (Section 5). Section 6 briefly overviews some related approaches that also seek to automate the design and tuning of search algorithms. Finally, Section 7 highlights the main research trends in hyper-heuristics and suggests some potentially interesting future research directions.

2. Origins and early approaches

The ideas behind hyper-heuristics are not new. They can be traced back to the early 1960s, and can be found across Operational Research, Computer Science and Artificial Intelligence. We describe below some relevant intellectual roots and early approaches developed before 2000. We have identified the following four types of early approach:

2.1. Automated heuristic sequencing

Fisher and Thompson (1963) and Crowston *et al* (1963) hypothesised that combining scheduling rules (also known as priority or dispatching rules) in production scheduling would be superior to using any of the rules separately. This pioneering work should be credited with laying the foundations of the current body of research into hyper-heuristic methods. The proposition was for a method of combining scheduling rules using 'probabilistic learning'. The main conclusions from this study are the following: '(1) an unbiased random combination of scheduling rules is better than any of them taken separately; (2) learning is possible' (Fisher and Thompson, 1963).

In the 1990s, these ideas were revisited: Storer *et al* (1992, 1995) clearly stated the problem of designing a good combination of problem-specific (fast) heuristics in job-shop scheduling as a search problem, and defined neighbourhoods within the heuristic space. The approach discussed in Fang *et al* (1993, 1994) employed a genetic algorithm to search a space of sequences of heuristic choices in the context of open-shop scheduling. Later on, Hart and Ross (1998) applied a variant of this idea to dynamic job-shop scheduling problems. Hart *et al* (1998) used a genetic algorithm-based approach to solve a real-world scheduling and delivery problem. The approach combined two genetic algorithms that evolved heuristic choices, one to manage the assignment of orders, and the second to schedule the arrival of deliveries. The approaches discussed above were all 'online'. That is, directed to find good sequences of heuristics to solve a given instance of a problem. In contrast, the work by Drechsler and Becker (1995); Drechsler *et al* (1996) in the domain of electronic chip design used an evolutionary algorithm to learn (from a set of previous examples) successful heuristics that can be applied to instances from a given problem after a learning phase.

2.2. Automated planning systems

Another body of work that inspired the concept of hyper-heuristics came from the Artificial Intelligence community. In particular, from work on automated planning systems and the problem of learning control knowledge. In Gratch *et al* (1993), Gratch and Chien (1996), the so-called COMPOSER system was used for controlling satellite communication schedules. The system can be characterised as a hill-climbing search in the space of possible control strategies. The approach is off-line in that a large supply of representative training problems are required in order to have an adequate estimate of the expected utility for various control strategies. This methodology employed domain-specific knowledge, and thus differs from modern hyper-heuristic approaches. Moreover, a planning problem differs from the general formulation of optimisation problems: the objective in planning is to find a prescription for actions to change the initial state into one that satisfies the goal.

2.3. Automated parameter control in evolutionary algorithms

Some early approaches to automatically set the parameters of evolutionary algorithms can also be considered as antecedents of hyper-heuristics. Several mechanisms for modifying parameters during the run in an 'informed', adaptive way were proposed early in the history of evolutionary algorithms (Eiben *et al*, 1999). Another idea is that of using an evolutionary algorithm to tune an evolutionary algorithm. This can be done using two evolutionary algorithms: one for problem solving and another one (so-called meta-evolutionary algorithm) to tune the first one (Grefenstette, 1986; Freisleben and Härtfelder, 1993). It can also be done by using a single evolutionary algorithm that tunes itself to a given problem while solving it. The notion of 'self-adaptation', first introduced within evolution strategies for varying the mutation parameters (Rechenberg, 1973; Schwefel, 1977), is an example in this category. Self-adaptation in evolutionary algorithms means that some parameters are varied during a run in a specific manner: the parameters are included in the chromosomes and co-evolve with the solutions. These approaches are related to the idea of searching over a space of possible algorithm configurations, and are, therefore, related to hyper-heuristics. For an overview and classification of approaches to parameter control in evolutionary algorithms, the reader is referred to Eiben *et al* (1999, 2007).

2.4. Automated learning of heuristic methods

An early approach to the automated generation of heuristic computer programs can be found in the domain of constraint satisfaction problems (Minton, 1996). In particular, the pioneering work of Minton (1996) presents a system for generating reusable heuristics for *Minimum Maximal Matching Problem*. The system modifies given elements of algorithm 'schema', which are templates of generic algorithms. The general idea is to automatically synthesise problem-specific versions of constraint satisfaction algorithms. The user provides a set of training instances that the system can experiment with during the configuration processes, and thus adapt to the particular instance distribution represented by the training instances. This study is unique in the literature, as the automated approach is compared against those produced by three NASA programmers, producing competitive results, and even often outperforming the human programmers. Another interesting learning approach in the mid-1990s was termed 'Teacher' (Wah *et al*, 1995; Wah and Ieumwananonthachai, 1999) (an acronym for TEchniques for the Automated Creation of HEuRistics), which was designed as a system for learning and generalising heuristics used in problem solving. The objective was to find improved heuristic methods as compared with existing ones, in applications with little or non-existent domain knowledge. The Teacher system employed a genetic-based machine learning approach, and was successfully applied to several domains such as: process mapping, load balancing on a network of workstations, circuit placement, and routing and testing.

3. A classification of hyper-heuristic approaches

As a framework for structuring this survey paper, we use the classification of hyper-heuristic approaches proposed in Burke *et al* (2010d) (Figure 1). This figure is reproduced here for completeness. This classification considers two dimensions: (i) the nature of the heuristics' search space, and (ii) the different sources of feedback information. According to the nature of the search space, we have (i) *heuristic selection*: methodologies for choosing or selecting existing heuristics, and (ii) *heuristic generation*: methodologies for

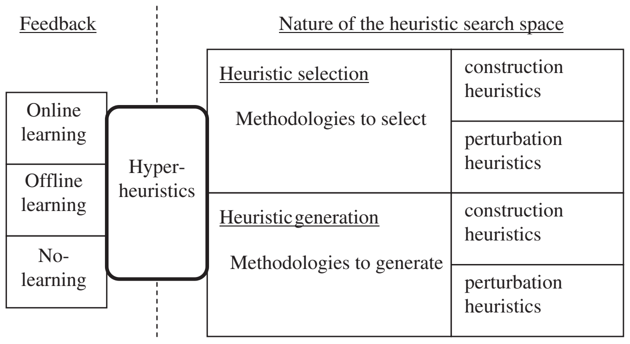

Figure 1 A classification of hyper-heuristic approaches, according to two dimensions: (i) the nature of the heuristic search space, and (ii) the source of feedback during learning (Burke *et al*, 2010d).

generating new heuristics from the components of existing ones. A second level in this dimension corresponds to the distinction between constructive and perturbative search paradigms (Hoos and Stützle, 2004). Perturbative methods work by considering complete candidate solutions and changing them by modifying one or more of their solution components, while constructive methods work by considering partial candidate solutions, in which one or more solution components are missing, and iteratively extending them.

A hyper-heuristic is a learning algorithm when it uses some feedback from the search process. According to the source of the feedback during learning, we can distinguish between *online* and *offline* learning. In online learning hyper-heuristics, the learning takes place while the algorithm is solving an instance of a problem, whereas in offline learning hyper-heuristics, the idea is to gather knowledge in the form of rules or programs, from a set of training instances, that will hopefully generalise to solving unseen instances.

These categories reflect current research trends. However, there are methodologies that can cut across categories. For example, we can see hybrid methodologies that combine constructive with perturbation heuristics (see eg Garrido and Riff, 2010), or heuristic selection with heuristic generation (Krasnogor and Gustafson, 2004; Maturana *et al*, 2010; Remde *et al*, 2012).

4. Heuristic selection methodologies

4.1. Approaches based on constructive low-level heuristics

These approaches build a solution incrementally. Starting with an empty solution, they intelligently select and use constructive heuristics to gradually build a complete solution. The hyper-heuristic framework is provided with a set of pre-existing (generally problem specific) constructive heuristics and the challenge is to select the heuristic that is somehow the most suitable for the current problem state. This process continues until the final state (a complete solution) has been reached. Notice that there is a natural ending to the construction process when a complete solution is reached. Therefore, the sequence of heuristic choices is finite and determined by the size of the underlying combinatorial problem. Furthermore, there is the interesting possibility of learning associations between partial solution stages and adequate heuristics for those stages.

Several approaches have recently been proposed to generate efficient hybridisations of existing constructive heuristics in domains such as timetabling, cutting and packing, production scheduling, constraint satisfaction, and vehicle routing problems (see Table 1). Both online and offline approaches, and different high-level strategies, or learning mechanisms have been investigated. The following subsections survey the approaches according to the application domain.

Table 1 Application domains of heuristic selection methodologies based on constructive low-level heuristics

Application domain	*Reference(s)*
Production scheduling	Fisher and Thompson (1963) Storer *et al* (1992, 1995) Dorndorf and Pesch (1995) Fang *et al* (1993, 1994) Norenkov and Goodman (1997) Hart and Ross (1998); Hart *et al* (1998) Vázquez-Rodríguez *et al* (2007a, b) Ochoa *et al* (2009b) Vázquez-Rodríguez and Petrovic (2010) Cano-Belmán *et al* (2010) Garcia-Villoria *et al* (2011)
Educational timetabling	Terashima-Marín *et al* (1999) Ahmadi *et al* (2003) Cheng *et al* (2003); Asmuni *et al* (2005) Ross *et al* (2004); Ross and Marín-Blázquez (2005) Burke *et al* (2005a, 2006b) Burke *et al* (2007c); Qu and Burke (2009) Ochoa *et al* (2009a) Li *et al* (2011) Pillay and Banzhaf (2007); Pillay (2008) Sabar *et al* (2011)
1D Packing	Ross *et al* (2002, 2003) Marín-Blázquez and Schulenburg (2007)
2D cutting and packing	Terashima-Marín *et al* (2006, 2007, 2009) Garrido and Riff (2007a, b) Lopez-Camacho *et al* (2010
Workforce scheduling	Remde *et al* (2007, 2009, 2012)
Constraint satisfaction	Terashima-Marín *et al* (2008) Ortiz-Bayliss *et al* (2010)
Vehicle routing	Garrido and Castro (2009); Garrido and Riff (2010)

4.1.1. Educational timetabling. There is a well-known analogy between a basic version of a timetabling problem and the graph colouring problem. Nodes can represent events and edges represent conflicts between events. Using this analogy, some timetabling algorithms in the literature are based upon graph colouring heuristics. These heuristics are criteria to select the next node to colour. Early approaches using evolutionary algorithms to evolve instructions for constructing an examination timetable rather than inducing the actual timetable were proposed in Ross *et al* (1997) and Terashima-Marín *et al* (1999). The idea was to use a non-direct chromosome representation based on evolving the configuration of constraint satisfaction methods for examination timetabling problems.

Ahmadi *et al* (2003) use a variable neighbourhood search algorithm to find good combinations of parameterised heuristics in examination timetabling. Several constructive heuristics are proposed based on a weighted decision function and the basic graph colouring heuristics. Low-level heuristics are used for exam selection, period selection, and room selection. The approach is used to solve real-world instance from the University of Nottingham, UK. A follow-up paper (Cheng *et al*, 2003) proposed a mixed-initiative approach to integrating human expertise, which supports the usefulness of hyper-heuristics in real-world problem-solving scenarios.

Asmuni *et al* (2005) investigate the use of a fuzzy system in solving course timetabling problems. The events (courses) to be scheduled are ordered by combining graph colouring heuristics. The fuzzy weight of an event is used to represent how difficult it is to schedule. A restricted form of exhaustive search is used to find the most appropriate shape for the fuzzy membership functions. The algorithm was tested on benchmark data sets with encouraging results.

Burke *et al* (2007c) propose a hyper-heuristic framework that implements commonly used graph colouring heuristics coupled with a random ordering heuristic. Tabu search is employed as the high-level search method for producing good sequences of the low-level heuristics. Each heuristic list produced by the tabu search algorithm is evaluated by sequentially using the individual heuristics to order the unscheduled events, and thus construct a complete timetable. This work also highlights the existence of two search spaces: the heuristic space and the problem solution space. The approach was tested on both course and exam timetabling benchmark instances with competitive results. A follow-up paper (Qu and Burke, 2009) compares the performance of several metaheuristics that operate on the search space of heuristics. Iterative techniques such as iterated local search and variable neighbourhood search were found to be more effective for traversing the heuristic search space. The study also implemented hybridisations of the hyper-heuristic framework with standard local search operating on the solution space, which was found to improve the performance of the overall system, making it competitive with state-of-the-art approaches on the studied benchmark instances. A further study (Ochoa *et al*, 2009a) uses the notion of fitness landscapes to analyse the search space of graph colouring heuristics. These landscapes are found to have a high level of neutrality (ie, the presence of plateaus). Furthermore, although rugged, they have the encouraging feature of a globally convex or *big valley* structure, which indicates that an optimal solution would not be isolated but surrounded by many local minima.

Li *et al* (2011) investigate two data mining techniques, artificial neural networks and binary logistic regression to find global patterns hidden in large data sets of heuristic sequences. With the trained classification rules, the performance of a resulting solution during the hyper-heuristic search can be predicted without the need to undertake the computationally expensive determination of the solution and calculation of the objective function. The approach was tested on the graph-colouring hyper-heuristic discussed above (Burke *et al*, 2007c), producing significant speed ups of the search process.

Pillay and Banzhaf (2007) and Pillay (2008) study the performance of evolutionary algorithms on a similar search space as that discussed above, namely, the space of combinations of graph colouring heuristics for examination timetabling. In the initial study (Pillay and Banzhaf, 2007), each element of the population is a variable length string, where each character represents a heuristic. The approach produced feasible examination timetables with soft constraints within the range of other search methods employed for this purpose, and outperformed previous hyper-heuristics on a number of the tested instances.

Ross *et al* (2004) and Ross and Marín-Blázquez (2005) apply a messy genetic algorithm (Goldberg *et al*, 1990) hyper-heuristic based on graph colouring heuristics to both class and exam timetabling problems. The idea is to learn associations between problem states and adequate heuristics for timetabling. Specifically, the system tries to discover a set of labelled points in the space of the problem states. Each label refers to a heuristic, and the algorithm works by repeatedly finding the nearest labelled point to the current condition and applies its label until a complete solution is built. Various different forms of problem-state description and methods of measuring the fitness were studied. The approach was able to generate fast and simple problem-solving algorithms that offer good performance over a range of exam and class timetabling problems.

Burke *et al* (2005a, 2006b) use a knowledge discovery technique, case-based reasoning (Leake, 1996), as a heuristic selector for solving both course and exam timetabling problems. A set of graph colouring heuristics and a hill-climbing procedure were selected as low-level heuristics. In Burke *et al* (2006b), tabu search is employed to discover the most relevant features used in evaluating the similarity between problem-solving situations. The objective was to choose the best heuristics from the most similar previous problem-solving situation to construct good solutions for the problem in hand.

Sabar *et al* (2011) utilise hierarchical hybridisations of four low-level graph colouring heuristics for producing even orderings. A combined difficulty index is calculated by considering all the orderings and events are scheduled according to this index. The approach produced competitive result on the studied benchmark instances.

4.1.2. Production scheduling. Dispatching rules are among the most frequently applied heuristics in production scheduling due to their ease of implementation and low time complexity. Whenever a machine is available, a dispatching rule inspects the waiting jobs and selects the job with the highest priority to be processed next. Dispatching rules differ from each other in the way that they calculate priorities. As discussed in Section 2, many early hyper-heuristic approaches were based on dispatching rules. More recently, Vázquez-Rodríguez *et al* (2007a) considered combinations of over a dozen different despatching rules to solve a multi-machine cardboard box shop scheduling problem. A standard genetic algorithm was employed as the high-level search strategy with successful results. A multi-objective job shop problem was studied in Vázquez-Rodríguez and Petrovic (2010) with a similar approach. Solutions are represented as sequences of dispatching rules that are called one at a time and used to sequence a number of operations onto machines. The approach simultaneously searches for the best sequence of rules, and the number of operations to be handled by each rule. On different variants of the multi-objective job shop, the method obtained better results on all the studied instances when compared with a previous hyper-heuristic based on dispatching rules, and a conventional genetic algorithm using a permutation representation.

A study of the search space composed by sequences of dispatching rules is presented in Vázquez-Rodríguez *et al* (2007b), where a formal definition and some properties of these spaces are discussed. The notion of a *decision block* is also introduced to refer to a set of decisions that are treated as a single unit (ie processed by a single heuristic). A further study (Ochoa *et al*, 2009b) conducts a landscape analysis of the dispatching rules search space. Two different objective functions and several hyper-heuristic representation sizes (with different block sizes) are considered. The study confirms the suitability of these heuristic search spaces for evolving solutions to production scheduling problems. Moreover, similarities between this search space and the space of sequences of graph-colouring heuristics for timetabling were found.

Cano-Belmán *et al* (2010) propose a hyper-heuristic embedded within a scatter search (Laguna and Martí, 2003) framework. The approach is applied to the problem of sequencing products (mixed-model) on a paced assembly line, and considers a set of 20 priority rules as low-level heuristics. These priority rules are used to select a product among a set of candidates, and are based on product and work station features such as demand, processing time, idle time, work overload, etc. Following the scatter search methodology, the so-called *reference set* contains sequences of priority rules, whose combination is based on a rule frequency matrix. The approach was tested over a wide range of instances from the literature. The solutions obtained were, in many cases, of better quality than those found by previous state-of-the art approaches.

4.1.3. Bin packing. Ross *et al* (2002) used an accuracy-based classifier system (Wilson, 1995), in the domain of one-dimensional bin-packing, to learn a set of rules that associate characteristics of the current state of a problem with different low-level constructive heuristics. A simplified description of the current state of the problem is proposed, which considers the number of items remaining to be packed and their size ranges. For the learning process, a large set of benchmark instances from the literature were used. The trained system showed good generalisation to unseen problems. Another study using a different type of classifier systems was also successfully applied to solve 1D bin-packing problems (Marín-Blázquez and Schulenburg, 2007). Ross *et al* (2003) use the messy genetic algorithm hyper-heuristic (described above in the context of timetabling) for learning associations between problem states and adequate heuristics for bin-packing. The approach is applied to the 1D bin-packing problem and overall the results were found a little better than those obtained with the classifier system, and on a larger set of benchmark problems.

Terashima-Marín *et al* (2006) applied the messy genetic algorithm hyper-heuristic to solve 2D regular cutting stock problems. For a 2D problem, two types of heuristics are required: for selecting the figures and objects, and for placing the figures into the objects. The state of the problem is described by the percentage of pieces that remain to be packed. The approach produced general heuristic-combination rules that efficiently solved unseen instances often with better performance than the best single heuristic for each instance.

A more extensive investigation of the messy genetic algorithm approach on 2D regular instances is presented in Terashima-Marín *et al* (2009). The study also extended the hyper-heuristic system to handle 2D irregular (convex polygonal) bin packing problems. Very encouraging results are reported for both types of problems. A recent study applies machine learning techniques to extract relevant features and improve the problem state representation of irregular packing problems (Lopez-Camacho *et al*, 2010).

Garrido and Riff (2007a, b) propose a genetic algorithm-based hyper-heuristic for solving the 2D strip packing problems. In this case, the system is online, that is solution methods are evolved for solving a single problem instance. The approach uses a variable length representation that considers a categorisation of the low-level heuristics according to their functionality: greedy, ordering and rotational. Very good results are reported that even outperform some specialised algorithms for the problem and benchmark instances studied.

4.1.4. Workforce scheduling problem. Remde *et al* (2007) propose a hybrid hyper-heuristic method to solve a complex real-world scheduling problem. The approach decomposes the problem into smaller parts solving each part using exact enumerative methods. Constructive heuristics are used and combined to first select a task, and then select potential resources such as time for the task. This combination produces a large number of low-level heuristics (over 200) that need to be handled by the hyper-heuristic. Variable neighbourhood search and other simple hyper-heuristics were successfully used for deciding the order in which to solve the sub-problems. In Remde *et al* (2009) a tabu search-based hyper-heuristic dynamically adapting the tabu tenures is applied to the same problem and framework. A comprehensive study on this framework is presented in Remde *et al* (2012), where several hyper-heuristics are compared against a Variable Neighbourhood and a Greedy Selection method with favourable results. The best performing hyper-heuristics depend on the allotted CPU time. When low to medium CPU time is available, hyper-heuristics based on ranking methods using adaptive reinforcement of low-level heuristics perform well. For medium to high CPU time, it is the adaptive tabu tenure hyper-heuristic approach (Remde *et al*, 2009) producing the best results.

4.1.5. Constraint satisfaction. Terashima-Marín *et al* (2008) use the messy genetic algorithms hyper-heuristic framework for solving the dynamic variable ordering problem within a constraint satisfaction framework. The proposed framework produces combinations of condition-action rules, after going through a learning process. The evolved rules produced encouraging results when tested with a large set of randomly generated benchmark problems. Ortiz-Bayliss *et al* (2010), explore patterns of regularities in the relative effectiveness of two heuristics for constraint satisfaction. The approach works in two stages. In a training stage information about the performance of the heuristics in different scenarios is gathered; and in the second stage, this information is used to generate a hyper-heuristic that decides which heuristic to apply in the constructive process to produce a solution.

Ortiz-Bayliss *et al* (2012) describe a model for choosing the right variable ordering heuristics while solving a constraint satisfaction instance. A hyper-heuristic is represented as a set of vectors that maps instance features to low-level heuristics, and a local search algorithm is used to search for such vectors.

4.1.6. Vehicle routing. Garrido and Castro (2009) use a hill-climbing-based hyper-heuristic to solve the capacitated vehicle routing problem. The approach incorporates both constructive and perturbative heuristics. Specifically, it searches the space of sequences of constructive-perturbative pairs of low-level heuristics. These sequences are applied in order to construct and improve partial solutions. The approach was tested using some standard state-of-the-art benchmarks and compared against several well-known methods proposed in the literature, with competitive results. In a follow-up paper, the authors use an evolutionary hyper-heuristic for solving the dynamic vehicle routing problem (Garrido and Riff, 2010). The framework includes three types of low-level heuristics: constructive, perturbative, and noise heuristics, and evolves a sequence of combinations of them, which are applied in order to construct and improve partial solutions. The approach is evaluated on a large set of instances with different topologies and degrees of dynamism, and produced competitive results when compared with some well-known methods proposed in the literature.

4.1.7. Summary and discussion. From the very early studies in production scheduling (see Section 2), it can be inferred that a combination or sequencing of several rules or constructive heuristics is advantageous over using just a single one. This fact has been recently confirmed within different domains, such as educational timetabling, bin packing and others. Approaches in the literature have used both online and offline machine learning. In the online approaches, the idea is to search (learn) for a good sequence of heuristics that learn while solving a single instance of the problem at hand. A feature of this type of approach is the clear existence of two search spaces, the space of sequences of heuristics, and the space of solutions to the underlying problems. An important research question is, then, to study the structure of these new heuristic search spaces; and the relationship between the two spaces. An analysis of the landscapes of heuristic sequences on both educational timetabling and production scheduling has revealed common features, such as the existence of plateaus (neutrality): many different local optima are located at the same level in the search (ie have the same value). These common landscape features can, in principle, be exploited by high-level search strategies. With respect to the mapping between the two spaces, the heuristic search space is generally smaller and covers only a subset of the solution search space (but well-distributed areas). The role of the high-level heuristic appears to be to search within the limited areas quickly and to explore as widely as possible the solution space by re-starting from different heuristic sequences within a limited computational time. This clearly invites the hybridisation of hyper-heuristics with standard local search techniques in the problem solution space. In other words, search can be simultaneously conducted over the two search spaces.

The offline machine learning approaches proposed so far have been based on learning classifier systems and messy genetic algorithms and have been mainly applied to several bin-backing problems. One fundamental research issue in this type of approach is the determination of a simplified, yet accurate, representation of the state space in the construction process, since the learning process is directed to link state-space descriptions to useful low-level heuristics. Other fundamental issues are the sensitivity of results in relation to the particular choice of low-level heuristics, and whether the use of randomised heuristics is advisable. The determination of efficient learning or search techniques to be used as high-level strategies also deserves further study.

Finally, the exploration of additional domains, in which constructive heuristics are available, is another worthwhile research direction.

4.2. Approaches based on perturbative low-level heuristics

These approaches aim to improve a candidate solution through a process of automatically selecting and applying a heuristic. Online and offline machine learning techniques are valuable to the heuristic selection strategies in order to make informed decisions regarding which heuristic to employ at a given step. Hyper-heuristic methodologies based on perturbative heuristics have been applied to a wide variety of combinatorial optimisation problems as summarised in Table 2.

There are a few studies on the hyper-heuristic methodologies to select perturbative heuristics that perform *multi-point search* (processing multiple solutions). The majority of previously proposed approaches conduct a *single point search*. In a single point search-based hyper-heuristic framework, an initial candidate solution goes through a set of successive stages repeatedly until termination. First, a heuristic (or a subset of heuristics) is selected from a set of low-level perturbative heuristics and then applied to a single candidate solution. Finally, a decision is made about whether to accept or reject the new solution.

A hyper-heuristic to select perturbative heuristics performing single-point search combines two separate components: (i) *heuristic selection method* and (ii) *move acceptance method* as identified in Bilgin *et al* (2006) and Özcan *et al* (2008). This component decomposition presents a high level of modularity, indicating that either one of these components can be replaced by another method generating a new hyper-heuristic. An instance of a single-point search-based hyper-heuristic will be denoted as

Table 2 Most studied application domains of methodologies to choose perturbative heuristics

Application domain	*References*
Personnel scheduling	Cowling *et al* (2000, 2002b, c) Cowling and Chakhlevitch (2003) Han and Kendall (2003) Burke *et al* (2003b) Bai *et al* (2012) Mısır *et al* (2010)
Educational timetabling	Cowling *et al* (2000, 2002c) Burke *et al* (2003b, 2005b) Bilgin *et al* (2006) Chen *et al* (2007) Bai *et al* (2012, 2007) Özcan *et al* (2009, 2010) Demeester *et al* (2012)
Space allocation	Burke *et al* (2005c) Bai and Kendall (2005); Bai *et al* (2008)
Cutting and packing	Dowsland *et al* (2007) Bai *et al* (2012)
Vehicle routing	Pisinger and Ropke (2007) Meignan *et al* (2010) Mısır *et al* (2011)
Sports scheduling	Mısır *et al* (2009) Gibbs *et al* (2010)
Cross-domain (HyFlex)	Burke *et al* (2010b, 2011a) Ochoa *et al*, 2012a, b) Özcan and Kheiri (2011) Walker *et al* (2012) Drake *et al* (2012) Mısır *et al* (2012) Ping-Che *et al* (2012) Chan *et al* (2012) Gaspero and Urli (2012)

Heuristic Selection—Move Acceptance from this point forward. Different combinations of heuristic selection and move acceptance methods have been explored within the context of hyper-heuristics.

4.2.1. Learning selection in hyper-heuristics performing single point search. The heuristic selection that does not use any type of learning mechanism is based on either a *random* or an *exhaustive* process. A *learning* mechanism can be introduced into the heuristic selection process to improve the decision-making process over a set of possible neighbourhoods. Thabtah and Cowling (2008) discuss an offline learning strategy to detect a rule from *seen* problem instances to choose a low-level heuristic at a given decision point for solving *unseen* problem instances.

The majority of the heuristic selection methods used within the single point search-based hyper-heuristic framework generate *online* score(s) for each heuristic based on their performances. Then these values are processed and/or combined in a systematic manner to select the heuristic to be applied to the candidate solution at each step. All score-based heuristic selection techniques require five main components to be implemented: (i) initial scoring, (ii) memory length adjustment, (iii) strategy for heuristic selection based on the scores, (iv) and (v) score update rules in case of improvement and worsening, respectively. All low-level heuristics are assigned an initial score. Depending on the mechanism used, these scores might affect the performance of a hyper-heuristic. In general, initial scores are set to the same value, typically zero. Memory length determines the effect of the previous performance of a heuristic while making the heuristic selection at a decision point. Given a set of scores, heuristic selection can be performed in many different ways. For example, *max* strategy selects the heuristic with the maximal score. On the other hand, *Roulette-wheel* (*score proportionate*) strategy associates a probability with each heuristic that is computed by dividing each individual score by the total score. Then, a heuristic is selected randomly based on these probabilities. A high score generates a higher probability of being selected.

One of the commonly used methods in hyper-heuristics is *reinforcement learning*, see Kaelbling *et al* (1996) and Sutton and Barto (1998) for more details. A reinforcement learning system interacts with the environment (or a *model* of the environment) and given a state, takes an action based on a *policy*. By trial and error, the system attempts to learn which actions to perform by evaluating state and action pairs through accumulated rewards. In the context of hyper-heuristics, rewarding and punishing each heuristic depending on their individual performance during the search is a scoring mechanism. If a low-level heuristic improves a solution, then it is rewarded and its score gets updated positively, while a worsening move causes punishment of a heuristic by decreasing its score. Different combination of operators can be designed for reward and punishment.

The acceptance strategy is an important component of any local search heuristic (operating on any search space). Two different types of acceptance strategies can be identified in the literature: *deterministic* or *non-deterministic*. Deterministic methods make the same decision for acceptance regardless of the decision point during the search using given current and new candidate solutions(s). A non-deterministic approach might generate a different decision for the same input. The decision process in almost all non-deterministic move acceptance methods requires additional parameters, such as the time (or current iteration).

Single point search-based hyper-heuristics will be covered in four distinct subsections considering the nature of their components as follows: (i) Hyper-heuristics using deterministic move acceptance, (ii) Hyper-heuristics using heuristic selection methods with no learning and non-deterministic move acceptance, (iii) Hyper-heuristics using heuristic selection methods with learning and non-deterministic move acceptance, and (iv) Comparison studies. Section 4.2.5 presents an overview of multi-point search-based hyper-heuristics. Finally, Section 4.2.7 provides a summary and discussion.

4.2.2. Hyper-heuristics using deterministic move acceptance. In Cowling *et al* (2000, 2002c), the authors proposed and compared a variety of the hyper-heuristic components on two real-world scheduling problems: a sales summit and a project presentation problem, respectively. A *Simple Random* heuristic selection method chooses a low-level heuristic at random at each step. *Random Gradient* is a variant of Simple Random, a randomly selected heuristic is repeatedly applied until no improvement is achieved. The same affect of Random Gradient can be achieved by modifying the operation of each heuristic as discussed and employing Simple Random. *Random Permutation* generates a random ordering of the low-level heuristics and at each step successively applies a low-level heuristic in the provided order. *Random Permutation Gradient* is a variant of Random Permutation that proceeds in the same manner as Random Gradient without changing the order of heuristics until no improvement is achieved. Berberoğlu and Uyar (2010) showed that the Random Permutation Gradient-based hyper-heuristic performs better than some other meta-heuristics for solving the unit commitment problem. *Greedy* exhaustively applies all low-level heuristics to a candidate solution and selects the one that generates the best *improved* solution. Greedy is a learning heuristic selection method with the shortest memory length. The heuristic that makes the best improvement as a feedback is used for the heuristic selection and then this information is discarded in the following step. Although Random Gradient and Random Permutation Gradient heuristic selection methods make use of a random component, they can still be considered as *intelligent* heuristic selection mechanisms that embed a reinforcement learning mechanism. Initial scores of all heuristics are set to 0, which is also the lower bound for the scores. The upper bound is set to 1. As a score update rule, score of an improving heuristic is increased by one (additive), otherwise it is punished by decreasing its score by 1 (subtractive). The scores are kept within the bounds; hence, the memory length is set to the shortest possible value for such a reinforcement learning scheme. This type of strategy can be useful if the search landscape is highly rugged and there are not many plateaus. With the exception of Greedy, the other heuristic selection methods execute fast.

The *Choice Function* heuristic selection method introduced in Cowling *et al* (2000) is a score-based learning approach. This method adaptively ranks each low-level heuristic with respect to a combined score based on the following: how well it has performed individually, how well its performance is as a successor of previously invoked heuristic and the elapsed time since it was last called. The first two components intensify recent performance, while the third provides an element of diversification. For implementing the heuristic selection, max and roulette wheel strategies were tested, with the former approach producing better performance. As the acceptance criteria, two deterministic approaches were considered: *All Moves* (AM) and *Only Improvements* (OI). The experimental results in Cowling *et al* (2000) show that the Choice Function—All Moves hyper-heuristic is promising. The best parameter set is obtained through a manual tuning process. Cowling *et al* (2001) introduce a parameter-less Choice Function. In Cowling *et al* (2002c), this variant was found to outperform the simple ones over the problems studied, and produced improved results when compared with a manually produced solution and a constructive approach. The design of this hyper-heuristic is further extended in Rattadilok *et al* (2005) by proposing a model for general-purpose low-level-heuristics and exploiting parallel computing frameworks for the hyper-heuristics.

Nareyek (2003) used *Reinforcement Learning* (RL) as a heuristic selection method attempting to learn how to select the promising heuristic at each decision point. The learning process is based on scores (weights) as described previously. Each heuristic starts with the same score and they are updated by a predetermined scheme during the move acceptance process. All Moves is used as an acceptance criterion. The approach is evaluated on the Orc Quest problem (Nareyek, 2001), and in a modified Logistics Domain, well known to the action-planning community. The results of the study suggest that combining a low rate of adaptation (additive update) for rewarding an improvement with a strong (root update) rate of adaptation for punishing a deterioration is a good choice. Moreover, choosing a heuristic with a max strategy at each step often generates better results when compared with choosing a heuristic with a roulette wheel scheme.

Burke *et al* (2003b) presented the Reinforcement Learning with Tabu Search heuristic selection method. In a similar way to the previous study of Nareyek (2003), the low-level heuristics are selected according to learned scores (ranks). The proposed hyper-heuristic also incorporates a dynamic tabu list of low-level heuristics that are temporarily excluded from the available heuristics in certain situations. This hyper-heuristic is evaluated on various instances of two distinct timetabling problems: university course timetabling and nurse rostering. The results were competitive with respect to those obtained using the state-of-the art problem-specific techniques. Burke *et al* (2005c) extended this methodology with a fixed size tabu list to be used in multi-objective optimisation. The hyper-heuristic maintains the scores of low-level heuristics for each objective separately. The results show that the proposed multi-objective hyper-heuristic framework guides the search towards the promising areas of the trade-off front over a set of space allocation and timetabling problems.

In Cowling and Chakhlevitch (2003), a range of hyper-heuristics were studied based on Simple Random and Greedy heuristic selection methods. According to the description of the Greedy method in Cowling *et al* (2000), worsening moves are never accepted. On the other hand, it is possible that all heuristics might worsen the quality of a candidate solution when the Greedy approach is used. In Cowling and Chakhlevitch (2003), such situations are allowed. This is a more general approach that allows the move acceptance component to deal with worsening moves, enriching the generation of different hyper-heuristics embedding different acceptance mechanisms. In this study, *Peckish* heuristic selection strategies (Corne and Ross, 1996) that use a Greedy method after reducing the number of low-level heuristics are also investigated along with four different Tabu Search based move acceptance strategies. These strategies accept an improving move and the related heuristic is removed from the tabu list if it is there. A non-improving move is accepted only if the employed heuristic is not in the tabu list. The hyper-heuristics utilise Only Improving, All Moves and a variant of All Moves that discards moves generating the same objective value as the current solution as move acceptance criterion. The approaches were evaluated on a real-world personnel scheduling problem with 95 low-level heuristics yielding promising results. However, the process for selecting a low-level heuristic to apply at each decision point is slow since it involves examining all heuristics from a large set. Therefore, in Chakhlevitch and Cowling (2005), two learning strategies were investigated for choosing the subset of the fittest low-level heuristics. At each step, the changes in the quality of a solution are compiled to reflect the total improvement due to a heuristic. Greedy—Tabu Search (event-based tabu list) that linearly reduces the number of the fittest low-level heuristics turned out to be the most promising hyper-heuristic.

Garcia-Villoria *et al* (2011) applied a number of different hyper-heuristic methods to an NP-hard scheduling problem, namely, response time variability problem. The authors experimented with constructive and dual stage improvement hyper-heuristics. The improvement hyper-heuristic evaluates the performance of each low-level heuristic during a learning stage and improves a solution based on the performance indicators for each heuristic obtained from the previous stage. Mixing local search heuristics using a roulette wheel heuristic selection strategy based on objective values generated by each heuristic during the learning stage performed better than a naive iterative selection strategy. The local search heuristics were then replaced by a set of metaheuristics within the framework. The cooperation of metaheuristics via the hyper-heuristic framework works better than the use of each individual metaheuristic for solving the problem.

McClymont and Keedwell (2011) applied a heuristic selection method modelled as a Markov chain to a well-known multi-objective continuous optimisation benchmark DTLZ. This is one of the rare studies on the application of selection hyper-heuristics for continuous optimisation (Kiraz *et al*, 2011; Köle *et al*, 2012) handling multi-objectives. The proposed approach is based on a Reinforcement Learning scheme, which maintains a set of weighted edges representing probabilities of transitioning from one heuristic to another. After each invocation, the edge weights are updated based on the performance of a heuristic. Given a set of four low-level heuristics, this selection method was incorporated into a Evolution Strategy framework and compared with Simple Random and a roulette wheel heuristic selection method using a tabu list, referred to as 'TSRoulWheel' in Burke *et al* (2005c). The Markov chain hyper-heuristic accepted *non-dominated* solutions yielding a matching performance to the best heuristic on the benchmark instances.

4.2.3. Hyper-heuristics using heuristic selection with no learning and non-deterministic move acceptance. In Ayob and Kendall (2003), a set of *Monte Carlo*-based non-deterministic move acceptance strategies which accept all improving moves and some non-improving moves with a certain probability was proposed. The authors explored a *Linear* (LMC), an *Exponential* probability function (EMC), and included their most sophisticated formulation based on the computation time and a counter of consecutive non-improvement iterations (EMCQ). The EMCQ formulation is similar to that of a simulated annealing approach (Kirkpatrick *et al*, 1983; Cerny, 1985). The difference is that it does not include a temperature parameter and thus a cooling schedule. Hyper-heuristics combining Simple Random and {Linear, Exponential, EMCQ} are applied to scheduling of electronic component placement on a printed circuit board. Their performances are compared with the combination of {Simple Random, Choice Function} and {All Move, Only Improving} hyper-heuristics. Simple Random-EMCQ delivered a superior performance as compared with the hyper-heuristics using deterministic acceptance with and without learning for the given problem instances. Although it appears as if no parameter tuning is necessary for Monte Carlo-based hyper-heuristics, more instructions will be executed in a unit time on a faster machine compared with a slower machine; hence, Monte Carlo-based hyper-heuristics will be producing different results given the same number of iterations.

In Kendall and Mohamad (2004a), a variant of the *Great Deluge* acceptance criteria (Dueck, 1993) was incorporated within a hyper-heuristic framework. In this acceptance strategy, at each iteration, any configuration is accepted which is not much worse than an expected objective value, referred to as *level*, which changes at a linear rate every step from an initial towards a target objective value within given number of iterations. Simple Random—Great Deluge generated competitive results as compared with a constructive heuristic and a genetic algorithm for solving channel assignment benchmark problems, a real–world problem from the mobile communications industry.

In another study, Kendall and Mohamad (2004b) extended the *Record-to-Record Travel* acceptance criteria of Dueck (1993) to be used in a hyper-heuristic. Any new candidate solution is accepted which is not much worse than the current one within a given fixed limit. A Simple Random—Record-to-Record Travel hyper-heuristic is also applied to benchmark instances of a channel assignment problem. The empirical results suggest that this hyper-heuristic is superior to using All Move, Only Improving and EMCQ move acceptance strategies, performing comparable to a constructive heuristic and a genetic algorithm.

A *Simulated Annealing* acceptance method in hyper-heuristics was investigated in Bai and Kendall (2005). The approach is studied on a shelf space allocation problem. In Simulated Annealing, the improving solutions are always accepted, and worsening moves are accepted according to the Metropolis criterion (Kirkpatrick *et al*, 1983). The temperature is decreased during the algorithm run using a *cooling schedule*. The authors discuss how to compute the relevant Simulated Annealing parameters automatically. Different approaches are allowed to improve an initial candidate solution that is generated by a greedy heuristic. The results show that the Simple Random—Simulated Annealing hyper-heuristics outperform Simple Random—Only Improving, Simple Random—All Moves, Greedy—Only Improving, Choice Function—All Moves, two conventional simulated annealing approaches each using a different single neighbourhood operator in all problem instances tested. Two strategies to decide the initial temperature are compared. One of them computes the initial temperature as a factor of the initial objective value, while the other one samples a set of random solutions and makes the computation based on the largest objective difference. The former scheme performs slightly better than the latter one.

Antunes *et al* (2009) described a multi-objective Simple Random—Simulated Annealing hyper-heuristic for solving a power compensation problem in electricity distribution networks. Deciding the location of network nodes and the size of capacitors to be installed for reactive power compensation requires two conflicting objectives to be achieved, namely, cost and power loss. Six low-level heuristics are designed to make a move from one feasible solution to another. Simulated Annealing makes its acceptance decision based on the dominance between the new solution and the archived solutions. The weighted sum of two objective values is used in the acceptance probability function whenever needed. The results indicate that this hyper-heuristic performs slightly worse than a multi-objective genetic algorithm.

Late Acceptance method (Burke and Bykov, 2008) is a memory-based technique that maintains the history of objective values from the previous solutions in a list of given size, L. The new solution is compared with a previous solution obtained at the Lth step and the acceptance decision is made accordingly. Demeester *et al* (2012) used Simple Random-based hyper-heuristics focusing on different move acceptance method for examination timetabling. Improving or Equal, Great Deluge, Simulated Annealing, Late Acceptance and its variant, referred to as *Steepest Descent Late Acceptance*, were used as move acceptance criteria. Steepest Descent Late Acceptance first acts the same as Only Improving for a given solution and the incumbent solution before applying the generic Late Acceptance. The Simple Random—Simulated Annealing hyper-heuristic improved on a number of best results from the literature over the Toronto benchmark data set and performed well over another data set provided by the authors.

Mısır *et al* (2011) proposed a move acceptance method that adaptively sets the threshold value based on history. Simple random combined with the new acceptance method, referred to as *Adaptive Iteration Limited List-based Threshold Acceptance* (AILLA), is tested on a ready-mixed concrete delivery problem. The performance comparison of AILLA, Late Acceptance, Simulated Annealing, Great Deluge and Improving and Equal acceptance criteria showed that AILLA and Late acceptance outperform the rest if Simple Random is used for heuristic selection. It is observed that if the execution time is increased, then Simple Random—AILLA starts to outperform Simple Random—Late Acceptance.

4.2.4. Hyper-heuristics using heuristic selection with online learning and non-deterministic move acceptance. In Dowsland *et al* (2007), a variant of *Reinforcement Learning with Tabu Search* (RLTS) was hybridised with a *Simulated Annealing with Reheating* move acceptance strategy. In particular, RLTS is modified to employ a *batch learning* mechanism updating the performance of a heuristic based on the best objective value obtained after a number of iterations at each decision point. In addition, an undulating cooling schedule based on a geometric function is proposed as a means to deal with the effects of having different neighbourhood sizes (given by the pool of low-level heuristics used).

A reheating scheme is employed after a rejected move and the required acceptance rate is reduced periodically as discussed in Thompson and Dowsland (1996) at every given number of iterations. In a way, the updates of scores for low-level heuristics in Reinforcement Learning with Tabu Search and reductions of the acceptance rate in Simulated Annealing with Reheating are performed together. The proposed hyper-heuristic is applied to a packing problem of determining shipper sizes for storage and transportation. Real-world data from a cosmetics company are used as a base for generating experimental data. The Reinforcement Learning with Tabu Search—Simulated Annealing with Reheating hyper-heuristic is superior in performance than a simpler local search strategy (random descent). Bai *et al* (2012) presented a different hyper-heuristic scheme that was possibly inspired from the studies provided in Burke *et al* (2003b), Bai and Kendall (2005), Dowsland *et al* (2007). The proposed hyper-heuristic uses a reinforcement learning mechanism with a short-term memory as a heuristic selection component. Each heuristic receives a weight (score) that is updated periodically. In this study, a different Simulated Annealing with Reheating scheme is used as a move acceptance method which executes switching between annealing and reheating phases during the search. The proposed hyper-heuristic is tested on nurse rostering, course timetabling and bin packing problems and comparisons to previously proposed approaches show that it is competitive. On the other hand, Burke *et al* (2010a) show that this hyper-heuristic does not perform better than the hyper-heuristics using {Simple Random, Greedy, Choice Function} heuristic selection methods for examination timetabling. This study also shows that the hyper-heuristics based on Simulated Annealing and its reheating variant perform significantly better than the ones based on EMCQ move acceptance.

In Pisinger and Ropke (2007), a competent unified methodology was presented for solving different vehicle routing problems. The proposed approach extended the large neighbourhood search framework presented in Shaw (1998) with an adaptive layer. This layer adaptively chooses among a number of insertion and removal heuristics to intensify and diversify the search, according to scores for each heuristic accumulated during the iterations. The hyper-heuristic combines the adaptive heuristic selection mechanism with a standard Simulated Annealing acceptance strategy based on a linear cooling rate. A large number of tests were performed on standard benchmarks from the literature covering five variants of the vehicle routing problem. The results proved highly promising, as the methodology was able to improve on the best known solutions on some instances.

Özcan *et al* (2009) investigated different heuristic selection methods that would perform the best in combination with the Late Acceptance method. The results show that Simple Random performs the best when combined with Late Acceptance as compared with other hyper-heuristic approaches involving online learning. The delay within the acceptance strategy seems to deceive the learning mechanisms.

Bhanu and Gopalan (2008) combined a genetic algorithm, and a set of its hybrids with simulated annealing, tabu search and hill climbing, respectively under a Greedy—Great Deluge hyper-heuristic to schedule jobs in a grid environment. The hyper-heuristic performs better than each individual low-level metaheuristic over a small set of problems. Özcan *et al* (2010) showed that Reinforcement Learning—Great-Deluge and Simple Random—Late Acceptance were promising hyper-heuristics for solving the Toronto and Yeditepe examination timetabling problems.

Mısır *et al* (2009) introduced a hyper-heuristic combining a simple heuristic selection method based on a learning automaton. This method updates the probabilities of each low-level heuristic being chosen in an online manner and utilises a new move acceptance method, namely; *Iteration Limited Threshold Accepting* (*ILTA*). The experiments over a set of Traveling Tournament Problem instances from the US National League Baseball and the Super 14 Rugby League delivered promising performance. A two-phase hyper-heuristic based on ILTA was applied to the Eternity II puzzle yielding successful results (Vancroonenburg *et al*, 2010).

Mısır *et al* (2010) describe a tabu-based learning strategy to reduce the number of low-level heuristics for a number of phases using a quality index (*QI*) for each low-level heuristic (where $1 \leqslant QI \leqslant$ number-of-heuristics). *QI* reflects the quality of a heuristic at a phase and helps to compare performance differences and a low-level heuristic is selected randomly from the reduced set. The authors also introduce an adaptive variant of *ILTA* as a new move acceptance strategy. The resultant hyper-heuristic managing six perturbative heuristics performs better than Simple Random—Improving and Equal for home care scheduling.

Blazewicz *et al* (2011) studied Choice Function and Reinforcement Learning-based selection hyper-heuristics and their variants to predict DNA sequences. A set of low-level heuristics were defined that manipulate a given DNA sequence via insertion, deletion, swap and shift operations. Simulated Annealing and Accept All Moves were used as the move acceptance criteria. Using a base set of six low-level heuristics, Roulette Wheel-based Selection—Simulated Annealing outperformed all other hyper-heuristics. The tests were performed using different sets of low-level heuristics. The learning hyper-heuristics delivered similar performance to bespoke metaheuristics from the literature for DNA sequencing.

4.2.5. Other studies on selection hyper-heuristics. There are an increasing number of comparison studies on hyper-heuristics illustrating their success on different problem domains. After experimenting with 35 hyper-heuristics, Bilgin *et al* (2006) reported that although none of the hyper-heuristics dominated the others for benchmark function optimisation. According to the empirical results, Choice Function—Improving and Equal produces a slightly better mean performance. Moreover, Choice Function—Simulated Annealing and Simple Random—Great Deluge produce better quality solutions for exam timetabling. Özcan *et al* (2006) investigated the performance of different hyper-heuristic frameworks over a set of benchmark functions. The low-level heuristics are classified as either a *hill climber* (*meme*) that aims to generate an improved solution or a *mutational* heuristic that perturbs a candidate solution without considering whether the resulting solution will be improved or not. In this study, three alternative iterated local search inspired hyper-heuristic frameworks to the standard framework are proposed, which separate and enforce the hill-climbing process explicitly and deliver promising performances. The success of a hyper-heuristic based on a framework distinguishing between mutational and hill climbing heuristics is also confirmed across different problem domains in later studies (Burke *et al*, 2010b; Berberoğlu and Uyar, 2010). Özcan *et al* (2008) extended the studies in Bilgin *et al* (2006) and Özcan *et al* (2006) and illustrated that the choice of hill climber affected the performances of the relevant frameworks. Choice Function—Improving and Equal based on a general iterated local search framework with multiple perturbative neighbourhood operators and a prefixed hill climber performs well and its performance is similar to a generic memetic algorithm. The experimental results show that a memetic algorithm embedding a Simple Random—Improving and Equal hyper-heuristic, which is categorised as a static external-level adaptation mechanism in Ong *et al* (2006), also delivers a good performance. More on memetic algorithms utilising hyper-heuristics to choose hill climbers can be found in Ersoy *et al* (2007). Bai *et al* (2008) studied the performance of a set of *fast* approaches for solving a fresh produce inventory and shelf space allocation problem and compares against a variety of approaches (Bai and Kendall, 2008). The empirical results show that the Simulated Annealing-based hyper-heuristics perform better than Reinforcement Learning with Tabu Search—All Moves and they deliver a similar performance to a generic simulated annealing approach and GRASP (Feo and Resende, 1995). Gibbs *et al* (2010) reported that Reinforcement learning heuristic selection performed well in combination with Great Deluge and Simulated Annealing for producing a football fixture schedule for the holiday periods. On the other hand, Berberoglu and Uyar (2011) compared the performance of 24 learning and non-learning selection hyper-heuristics managing seven mutational and hill-climbing heuristics on a short-term electrical power generation scheduling problem. Random Permutation Descent—Only Improving performed the best on this problem.

In order to generate better learning schemes that will improve the decision-making process during heuristic selection, one should always remember that a learning process involves memory. Memory length can be handled in different ways. For example, the scores for low-level heuristics are updated based on the entire historical information in Burke *et al* (2003b) and Dowsland *et al* (2007). On the other hand, the minimum and maximum scores are bounded in Nareyek (2003). This approach serves as some type of a forgetting mechanism for successive improving or non-improving moves. The empirical study carried out in Bai *et al* (2007) on university course timetabling shows that hyper-heuristics using a short-term memory produce better results than both an algorithm without memory and an algorithm with infinite memory. The memory length is found to be sensitive to different problem instances.

There is a growing number of studies on multi-point (population)-based hyper-heuristics. These hyper-heuristics are either existing metaheuristics or used in/for cooperative search. In Cowling *et al* (2002b), an indirect genetic algorithm for solving a personnel scheduling problem was proposed. The approach can be regarded as a hyper-heuristic that uses a GA as the heuristic selection mechanism. Han and Kendall (2003) extend this study with adaptive length chromosomes and guided operators, producing promising results on the trainer scheduling problem when compared with both a direct encoding genetic algorithm and a memetic algorithm.

The ant colony algorithm was used as a hyper-heuristic in Burke *et al* (2005b) and Chen *et al* (2007) to address a personnel scheduling and a sports timetabling problem, respectively, with promising results. Similarly, Ren *et al* (2010) discussed an ant-based hyper-heuristic for solving the p-median problem. Different type of frameworks have been proposed to enable the use of multi-point-based search methods. For example, Vrugt and Robinson (2007) introduced the *AMALGAM* approach for continuous multi-objective optimisation that manages a set of population-based multi-objective approaches while producing a new population of solutions yielding an improved performance, eventually. The number of new solutions produced by each low-level approach is decided proportional to the percentage of previously created individuals that remains in the working population at each stage.

Cobos *et al* (2011) mixed different variants of evolutionary approaches for document clustering and tested different heuristic selection and move acceptance methods under a multi-point-based search framework. One of the initial studies hybridising generation and selection of heuristics was provided by Kampouridis *et al* (2012). In this study, a genetic programming approach is used to create decision trees for financial forecasting in order to make decisions whether to buy or not. During the multi-stage evolutionary process, the candidate solutions are improved through a reinforcement learning-based hyper-heuristic that manage a set of perturbative low-level heuristics.

Grobler *et al* (2012) mixed a set of metaheuristics including a genetic algorithm, particle swarm optimisation variants, CMA-ES and variants of differential evolution under a hyper-heuristic framework. The authors investigated different ways of employing local search in combination with those low-level heuristics over a set of benchmark functions.

Tsai *et al* (2012) presented a simple random-based hyper-heuristic framework which is able to mix single and multi-point-based metaheuristics for data clustering.

Crainic and Toulouse (2003) classified *cooperative search methods* as type 3 parallel strategies that allowed multiple search methodologies to guide the search process via information sharing in a multi-thread environment. These strategies can be thought of as parallel/distributed hyper-heuristics which have been increasingly used to combine multiple low-level (meta-)heuristics.

Biazzini *et al* (2009) combine several algorithms for numerical optimisation such as differential evolution and random search in a distributed framework in an island model.

Meignan *et al* (2010) presented a self-adaptive and distributed approach based on agents and hyper-heuristics. Several agents concurrently explore the search space using a set of operators. The approach is applied to vehicle routing.

Ouelhadj and Petrovic (2010) proposed an agent-based cooperative hyper-heuristic framework composed of a population of heuristic agents and a cooperative hyper-heuristic agent. Computational experiments on a set of permutation flow shop benchmark instances illustrated the superior performance of the cooperative hyper-heuristic framework over sequential hyper-heuristics.

Dynamic environment problems represent a challenging set of problems in which the environment changes over time during the search process. Successful approaches are highly adaptive and can react rapidly whenever a change occurs (Branke, 2002; Cruz *et al*, 2011). Kiraz and Topcuoglu (2010) hybridised an evolutionary algorithm with a selection hyper-heuristic for solving dynamic generalised assignment problem. Kiraz *et al* (2011) experimented with hyper-heuristics using different heuristic selection methods on moving peaks benchmark. Choice Function—Improving and Equal performed the best managing seven parameterised Gaussian-based mutation operators across a variety of change scenarios. Köle *et al* (2012) showed that Simple Random choice is a viable strategy if the environment changes fast and there is noise on *The Open Racing Car Simulator* (TORCS). Uludag *et al* (2012) presented a framework hybridising Estimation Distribution Algorithms and hyper-heuristics for solving discrete dynamic environment problems. This approach uses multi-population combining offline and online learning to deal with random as well as cyclic dynamic environments.

4.2.6. The HyFlex benchmark framework and the Cross Domain Heuristic Challenge (CHeSC) 2011. HyFlex (Hyper-heuristic Flexible framework) (Ochoa *et al*, 2012a) is a software framework for the development of hyper-heuristics and cross-domain search methodologies. The framework features a common software interface for dealing with different combinatorial optimisation problems, and provides the algorithm components that are problem specific. In this way, the algorithm designer does not require a detailed knowledge of the problem domains, and thus can concentrate his/her efforts on designing adaptive general-purpose optimisation algorithms. In an initial implementation, HyFlex provided four combinatorial problems implemented in Java, namely: boolean satisfiability, one-dimensional bin packing, permutation flow shop and personnel scheduling, including for each problem a set of heuristic/search operators, initialisation routines. These four domains represented the training benchmark that supported an international research competition: the first Cross-Domain Heuristic Search Challenge (Ochoa and Hyde, 2011) that attracted significant international attention. The challenge is analogous to the athletics Decathlon event, where the goal is not to excel in one event at the expense of others, but to have a good general performance on each. Competitors submitted one Java class file using HyFlex representing their hyper-heuristic or high-level search strategy. This ensures that the competition is fair, because all of the competitors must use the same problem representation and search operators. Moreover, due to the common interface, the competition considered not only hidden instances, but also two hidden domains. Two additional domains were later implemented, namely: the travelling salesman and vehicle routing problems. The competing algorithms were compared (on both the training and hidden domains) and ranked using a simple point mechanism inspired by the Formula 1 scoring system. More details about the competition scoring system, experimental setting, and best performing algorithms can be found in Ochoa *et al* (2012a).

We give here a brief overview of publications so far based on the HyFlex framework and using the CHeSC 2011 benchmark software. The first article implementing hyper-heuristics using HyFlex was published in 2010 (Burke *et al*, 2010b), where several hyper-heuristics combining two heuristic selection mechanism and three acceptance criteria were compared. A multiple neighbourhood iterated local search was also implemented and found to outperform the other approaches as a general optimiser. This iterated local search hyper-heuristic contains a perturbation stage, during which a neighborhood move is selected uniformly at random (from the available pool of mutation and ruin-recreate heuristics) and applied to the incumbent solution, followed by a greedy improvement stage (using all the local search heuristics). The approach is extended in Burke *et al* (2011a) by substituting the uniform random selection of neighbourhoods in the perturbation stage by online learning strategies, significantly improving the performance. This implementation was the best performing hyper-heuristic before the competition started.

Özcan and Kheiri (2011) implemented a multi-stage hyper-heuristic, combining a greedy stage with a random descent stage, followed by a simple solution acceptance mechanism. This relatively simple approach produces very good results when compared with previous HyFlex hyper-heuristics (before the competition).

Walker *et al* (2012) describe in detail the design of one of the CHeSC 2011 hidden domains, namely the vehicle routing problem with time windows. The article also implements a new multiple neighbourhood iterated local search algorithm that includes adaptive mechanisms for both the perturbation and improvement stages. This implementation outperformed all the CHeSC competitors in the vehicle routing domain.

Drake *et al* (2012) describe a variant of the choice function heuristic selection with a simple new initialisation and update scheme for the weights of diversification and intensification.This approach ranks the 20th while its modified version ranks the 12th among the hyper-heuristics proposed by the CHeSC competitors (emphasising the importance of tuning).

Mısır *et al* (2012) implement an approach including two stages: heuristic selection and solution acceptance. Heuristic selection is done by learning dynamic heuristic sets, and effective pairs of heuristics. The algorithm also incorporates adaptation of the heuristic parameters, and an adaptive threshold acceptance. This approach was the winner of the CHeSC competition.

Ping-Che *et al* (2012) implement a variable neighbourhood search algorithm that orders perturbation heuristics according to strength. It includes two stages: diversification and intensification and incorporates adaptive techniques to adjust the strength of the local search heuristics. This approach obtained the second place in the competition.

Chan *et al* (2012) implement a hyper-heuristic that can assemble different iterated local search algorithms. The authors use the metaphor of pearl hunting; there is a diversification stage (surface and change target area) and an intensification stage (dive and find pearl oysters). The algorithm also uses offline learning to identify search modes. This approach obtained the fourth place in the competition, and, interestingly, was able to find new best-known solutions for the personnel scheduling problem (Ochoa *et al*, 2012a).

Gaspero and Urli (2012) used reinforcement learning for heuristic selection and explored several variants for the rewards, policy and learning functions. Different ways of modelling the agents' states and actions were also explored.

Ochoa *et al* (2012b) present a number of extensions to the HyFlex framework that enable the design of more effective adaptive heuristics. The article also demonstrates that adaptive evolutionary algorithms can be implemented within the framework, and that the use of crossover and a diversity metric produced improved results, including a new best-known solution on the studied vehicle routing problem.

Another software framework, Hyperion, was proposed in Swan *et al* (2011) that provides a general recursive framework supporting the development of any type of (meta-)heuristic, selection hyper-heuristic and their hybrids.

4.2.7. Summary and discussion. Most of the existing hyper-heuristics to select perturbative heuristics are designed based on a single point search framework. Initial studies concentrate on utilising different mechanisms to manage a set of low-level heuristics. As hyper-heuristics were initially defined as 'heuristics to choose heuristics', almost no emphasis is given to the acceptance mechanisms during these initial studies. Later, it has been observed that by using more sophisticated move acceptance criteria, the performance can be improved substantially. After the initial studies, there has been a rapid growth in the usage of well-known acceptance methods in different hyper-heuristic frameworks. A range of heuristic selection methods have been investigated such as the Choice Function (Cowling *et al*, 2000, 2002c) and reinforcement learning variants (Nareyek, 2003; Burke *et al*, 2003b; Dowsland *et al*, 2007; Gibbs *et al*, 2010; Mỳsỳr *et al*, 2010). Simulated annealing (Bai and Kendall, 2005; Bilgin *et al*, 2006; Dowsland *et al*, 2007; Bai *et al*, 2012), late acceptance (Özcan *et al*, 2009; Demeester *et al*, 2012) and variants of threshold acceptance (Kendall and Mohamad, 2004a; Bilgin *et al*, 2006; Mısır *et al*, 2009; Özcan *et al*, 2009; Mısır *et al*, 2011) turn out to be appropriate choices as move acceptance components to be used within hyper-heuristics to select perturbative heuristics. It appears that the choice of move acceptance component is slightly more important than the choice of heuristic selection.

In order to observe how well the proposed hyper-heuristics generalise, they need to be applied to several problem domains. We can expect more comparative studies in the future. One of the goals of hyper-heuristic research is raising the level of generality. In this context, it is often the case that a hyper-heuristic does not aim to outperform a custom-made solver for a given problem. In such an environment, applicability over a wide range of problem domains is more crucial. For this reason, comparison measures across different problems are of interest. Selection hyper-heuristics are highly adaptive search methodologies. There is strong empirical evidence that they can handle not only static optimisation problems, but also dynamic environments (Kiraz and Topcuoglu, 2010; Kiraz *et al*, 2011; Uludag *et al*, 2012). There is a vast literature on dynamic environments. Both communities can benefit from interaction. The current studies attempt to bring hyper-heuristics into dynamic environments, while the use of existing techniques in the field of dynamic environments would also be beneficial in the development of adaptive selection hyper-heuristics.

The theoretical study on selection hyper-heuristics is extremely limited. In a recent study by He *et al* (2012), a theoretical comparison was performed between a pure strategy using a single mutation operator and a mixed strategy using multiple mutation operators within the framework of $(1+1)$ EA based on a performance measure, referred to as *asymptotic hitting time*. The authors showed that the asymptotic hitting time of the $(1+1)$ EA with a mixed strategy using a set of mutation operators based on a prefixed distribution is not worse than the $(1+1)$ EA with the worst pure strategy using a single operator from that set. This type of studies is important for bridging the gap between theory and practice. It is crucial to have theoretical support motivating the development of selection hyper-heuristics.

CHeSC 2011 set an interesting benchmark for selection hyper-heuristics. We expect that there will be more studies on hyper-heuristics extending the features of the interface and even introducing new benchmarks based on HyFlex (Ochoa *et al*, 2012a) and Hyperion (Swan *et al*, 2011).

5. Heuristic generation methodologies

The previous section covered heuristic selection methodologies. In contrast, this section will review another class of hyper-heuristics: heuristic *generation* methodologies. The defining feature of this class is that the hyper-heuristic searches a space of heuristics constructed from components rather than a space of complete, pre-defined, heuristics. While both classes output a solution at the end of a run, a heuristic generator also outputs the new heuristic that produced the solution, and this heuristic can be potentially reused on new problem instances.

Genetic programming (Koza, 1992; Koza and Poli, 2005) is an evolutionary computation technique that evolves a population of computer programs, and is the most common methodology used in the literature to automatically generate heuristics. In the case that the evolved programs are heuristics, genetic programming can be viewed as a hyper-heuristic to generate heuristics. However, genetic programming is not inherently a hyper-heuristic, as the evolved programs can also directly represent problem solutions. For example, in symbolic regression, the solution is a formula, and the 'programs' in the population are candidate formulas, which are not used as heuristics. As another example, genetic programming can be employed to evolve programs which construct the design of artefacts such as bridges, circuits, and lenses. These programs are not heuristics, but a series of deterministic instructions.

Automatically generated heuristics may be 'disposable' in the sense that they are created for just one problem, and are not intended for use on unseen problems. This terminology was first used by Bader-El-Den and Poli (2007). Alternatively, the heuristic may be created for the purpose of reusing it on new unseen problems of a certain class. It is generally the case that all heuristics generated by a hyper-heuristic can technically be defined as reusable, as they *can* be applied to a new instance to produce a legal solution. However, they may not perform well on new instances if the particular hyper-heuristic methodology has not been designed with reusability in mind. For a generated heuristic to be successful when reused, the hyper-heuristic would usually train it offline, on a set of representative problem instances.

There are a number of potential advantages of automatically generating heuristics. The characteristics of problem instances vary, and obtaining the best possible result for an instance would ideally require a new heuristic specialised to that instance, or a specialised variation of a previously created heuristic. It is inefficient for a human analyst to specialise heuristics on a per-instance basis. As such, human created heuristics are rarely successful on only one problem instance; they are usually designed to be effective on a given class of problems. In contrast, an automated heuristic design process makes it potentially feasible and cost effective to design a heuristic for each problem instance. As the process is automated, it is less demanding on human resources and time. As it is more specialised, a generated heuristic could even produce a better solution than that which can be obtained by any current human-created heuristic, and many such examples are discussed in this section.

For example, 'best-fit' is a human-created heuristic for one-dimensional bin packing, which performs well on a wide range of bin packing instances. It was created as a general heuristic for all bin packing problems, and no heuristic is superior in both the average and worst case (Kenyon, 1996). However, over a narrower set of bin packing problems with piece sizes defined over a certain distribution, best-fit can be outperformed by automatically generated heuristics which are 'tailored' to the distribution of piece sizes (Burke *et al*, 2007b).

Table 3 Application domains of heuristic generation methodologies

Application domain	*References*
Production scheduling	Jakobovic *et al* (2007) Ho and Tay (2005) Tay and Ho (2008) Dimopoulos and Zalzala (2001) Geiger *et al* (2006)
Cutting and packing	Burke *et al* (2006a, 2007a, b) Poli *et al* (2007) Kumar *et al* (2008) Allen *et al* (2009) Burke *et al* (2010c, e, 2011b) Özcan and Parkes (2011) Burke *et al* (2012) Sim *et al* (2012)
Satisfiability	Fukunaga (2002, 2004, 2008) Bader-El-Den and Poli (2007, 2008) Lokketangen and Olsson (2010)
Travelling salesman problem	Keller and Poli (2007a, b, 2008a, b, c) Oltean and Dumitrescu (2004) Runka (2009)
Function optimisation	Oltean (2005) Oltean and Grosan (2003) Tavares *et al* (2004)
Timetabling and scheduling	Pillay (2009) Bader-El-Den *et al* (2009)
Additional domains	Drechsler and Becker (1995) Drechsler *et al* (1996) Minton (1996) Schmiedle *et al* (2002) Stephenson *et al* (2003) Oltean and Grosan (2003) Tavares *et al* (2004) Oltean (2005) DiGaspero and Schaerf (2007) Kumar *et al* (2009) Pappa and Freitas (2009) Nguyen *et al* (2011) Elyasaf *et al* (2012) van Lon *et al* (2012)

Table 3 presents a summary of papers that involve the automatic generation of heuristics. The rest of the section is organised by application area as follows: production scheduling (Section 5.1), cutting and packing (Section 5.2), SAT (Section 5.3), the travelling salesman problem (Section 5.4), and timetabling and scheduling (Section 5.5). These are the domains most widely studied in the literature. A summarising discussion is provided in Section 5.6.

5.1. Production scheduling

Genetic programming has rarely been used to solve production scheduling instances directly, because of the difficulty of encoding solutions. However, it is highly suitable for encoding scheduling *heuristics* (Jakobovic *et al*, 2007). The evolution of dispatching rules is the most common application of hyper-heuristics in this domain.

Ho and Tay (2005) and Tay and Ho (2008) employ genetic programming to evolve composite dispatching rules for the flexible job shop scheduling problem. Jakobovic *et al* (2007) employ a similar technique for the parallel machine scheduling problem. The evolved dispatching rules are functions, which assign a score to a job based on the state of the problem. When a machine becomes idle, the dispatching rule function is evaluated once for each job in the machine's queue, and each result is assigned to the job as its score. The job in the queue with the highest score is the next job to be assigned to the machine.

The hyper-heuristic combines components from previously published dispatching rules. The results obtained by the best evolved dispatching rule are better on over 85% of the instances. This shows that genetic programming can combine and rearrange heuristic components to create heuristics superior to those which have been created by humans. Importantly, the heuristics are shown to be reusable on new problem instances because an appropriate training set is used to train them during their evolution.

Dimopoulos and Zalzala (2001) evolve priority dispatching rules for the single machine scheduling problem to minimise the total tardiness of jobs. The component set is based on the human designed 'Montagne' dispatching rule, and contains five elements, representing both global and local job information. While the component sets are relatively simple, the system evolves heuristics superior to the Montagne, ADD, and SPT heuristics.

Geiger *et al* (2006) also employ genetic programming to evolve dispatching rules for single machine problems. The component sets are expanded from that presented by Ho and Tay (2005) and Dimopoulos and Zalzala (2001). Human competitive heuristics are produced under a variety of scheduling conditions, often replicating the human-created heuristics for the problems. The system also obtains human-competitive results on a two-machine flow-shop problem, where a unique dispatching rule is evolved for each machine simultaneously.

5.2. Cutting and packing

Burke *et al* (2006a, 2007a, b) employ a genetic programming hyper-heuristic methodology to generate heuristics for one-dimensional bin packing. The heuristics generated by this system are functions consisting of arithmetic operators and properties of the pieces and bins. The heuristics operate within a fixed framework that packs the pieces of an instance one at a time. For each piece in turn, the framework iterates through all of the bins, executing the heuristic function once for each. The heuristic returns a value for each bin. The bin that receives the highest value is the one into which the piece is placed (Burke *et al*, 2007b). These heuristics maintain their performance on new instances much larger than the training set in Burke *et al* (2007a). This work shows that there is a trade-off between the time taken to evolve a heuristic on larger instances, and the heuristic's scalability. Burke *et al* (2010c) extend this work by adding a memory component to the genetic programming system. It maintains a record of the pieces that have been seen so far during the packing process. The results show that the GP evolves heuristics which use this component, and that those heuristics perform better because of it.

Poli *et al* (2007) also employ genetic programming to evolve heuristics for one-dimensional bin packing. The structure within which their heuristics operate is based on matching the piece size histogram to the bin gap histogram, and is motivated by the observation that space is wasted if, when placing a piece into a bin, the remaining space is smaller than the size of the smallest piece still to be packed.

Constructive heuristics for the two-dimensional strip packing problem have been evolved with genetic programming in Burke *et al* (2010e), which are competitive with the best human-created constructive heuristic in the literature. In contrast to their previous work on one-dimensional bin packing, the heuristics operate on the offline problem. The heuristics choose the most appropriate piece to pack next, and where to place it in the solution.

Allen *et al* (2009) evolve heuristics for three-dimensional knapsack packing. Their performance is compared with the human-created best-fit heuristic and a simulated annealing methodology. While the evolved heuristics are worse than best-fit on most instances, they are competitive with the simulated annealing method. Burke *et al* (2011b) extend this work by representing one- and two-dimensional packing problems as three-dimensional problems, and therefore creating a system which can evolve heuristics for packing problems of all dimensions. It remains an open question as to how to automatically generate 3D packing heuristics which maintain their performance on new problems. The literature shows that it is often the case that automatically designed heuristics can only be relied upon to maintain their performance on new instances of the same problem class as they were evolved on. Therefore, a possible reason for the difficulties in the 3D packing domain could be that it is more difficult to define classes of instances, as problem instances are generally very diverse. Benchmark classes of instances exist, but within each class the instances may not share characteristics which allow one heuristic to specialise on that class.

Hyper-heuristics that generate heuristics for combinatorial optimisation problems often generate constructive heuristics. This is by far the most common format, especially for cutting and packing problems. The other alternative is to generate local search heuristics, which start with a complete solution and iteratively improve it. There are many choices to make when designing a local search algorithm, and among the most important is the neighbourhood move operator. The work of Burke *et al* (2012) shows that the space of neighbourhood move operators can be specified by a grammar, and high-quality operators can be evolved using a grammatical evolution technique.

Recent work by Kumar *et al* (2008) presents a genetic programming system that evolves heuristics for the biobjective knapsack problem. This is the first paper in which heuristics for a multiobjective problem have been automatically generated with a hyper-heuristic. To pack a knapsack instance, an evolved heuristic iterates through the list of pieces still to be packed, and is evaluated on each, using the profit and weight of the piece as inputs. When an evaluation returns a value of greater than or equal to one, then the iteration stops and that piece is packed. This is similar to the bin packing methodology of Burke *et al* (2006a), as it uses a threshold to make a decision, before all of the options have been evaluated.

Özcan and Parkes (2011) introduce a matrix representation of policies (heuristics). An offline learning genetic algorithm using this efficient representation created constructive heuristics that outperformed human designed heuristics for online bin packing.

Sim *et al* (2012) present a methodology to generate a selection hyper-heuristic (See section 4.1) for the one-dimensional bin packing problem. The idea is to automatically design a selection mechanism which will choose the correct heuristic for the characteristics of a given problem instance. An evolutionary algorithm is employed to evolve combinations of problem characteristics which are similar to those used previously by Ross *et al* (2002). The advantage of automatic generation in this case is to evolve characteristics that are actually relevant to the performance of the heuristics, and the results corroborate other studies which show that selecting from a set of heuristics produces better results than any one heuristic used in isolation.

5.3. Boolean satisfiability (SAT)

Fukunaga presents 'CLASS' (Composite Learned Algorithms for SAT Search), an automated heuristic discovery system for the SAT problem. Earlier papers by Fukunaga (2002, 2004) represent the initial work, while much more analysis is given in Fukunaga (2008). SAT is a domain where the most successful heuristics from the literature have a similar structure. Indeed, better heuristics have been created simply by adding a 'random walk' element to an existing heuristic. Fukunaga has broken this structure down into component parts, and the CLASS system is a genetic programming methodology used to evolve human competitive heuristics consisting of these components. Among others, there are some components which supply a set of variables, some of which select a variable from such a set, and some which make use of conditions to decide which subset of components to execute. Fukunaga (2008) shows that certain human-created heuristics from the literature, such as GWSAT and WalkSAT, can be represented with this component set. Fukunaga states that, because of the number of possibilities involved, the task of combining the components to create effective new heuristics is difficult for humans, but well suited for an automated system.

Fukunaga (2002, 2004, 2008) does not employ the genetic programming operators of crossover and mutation in their standard form. Instead of standard crossover, individuals are combined with a conditional operator, which keeps the original individuals intact and 'blends' their behaviour. If the condition is met, one individual is executed, else the other is executed.

Bader-El-Den and Poli (2007) observe that this results in heuristics consisting of other nested heuristics. The heuristics are composites of those in early generations, and are therefore relatively slow to execute. Bader-El-Den and Poli present a different heuristic generation methodology for SAT, which makes use of traditional crossover and mutation operators to produce heuristics which are more parsimonious, and faster to execute. A grammar is defined, which can express four existing human-created heuristics, and allows significant flexibility to create completely new heuristics.

ADATE is a methodology that generates code in a subset of the functional programming language ML. Lokketangen and Olsson (2010) show how this technique can be utilised to automatically generate metaheuristic code. They apply the methodology to the boolean optimisation problem (BOOP). They begin with an ML implementation of a tabu-search metaheuristic from the literature, and ADATE modifies the section of the code that decides which variable to flip next. This is an example of a common methodology to begin with an existing algorithm as a template, and allow the automatic code generator to modify the 'heuristic' sections of the algorithm which make the decisions during a search.

5.4. Travelling salesman problem

Keller and Poli (2007b) present a linear genetic programming hyper-heuristic for the travelling salesman problem. The hyper-heuristic evolves programs that represent the repeated application of a number of simple local search operations. The programs are sentences of a language defined by a grammar, and the grammar is progressively made more complex over a series of papers, including conditional components and loops (Keller and Poli, 2007a, 2008a, b, c).

Also for the travelling salesman problem, Oltean and Dumitrescu (2004) evolve constructive heuristics, as opposed to the local search heuristics evolved by Keller and Poli (2007b). They use multi-expression programming to evolve functions that decide which node of the graph to add to the tour. This is another example of the common technique of evolving a scoring function which operates within a fixed iterative framework (such examples have been discussed in Sections 5.1, 5.2, and 5.3). The decision-maker is evolved, but the context within which the decision is made remains fixed. In general, at each decision point, the function is evaluated on all available options to obtain a score for each. The option with the highest score is the one that is actually performed on the partial solution. In this case, Oltean and Dumitrescu (2004) apply the candidate function to all of the cities that are not yet included in the partial tour, and the one which receives the highest score from the function is added to the tour.

One general methodology for automatically generating heuristics is to evolve a fundamental part of an existing metaheuristic search algorithm. This has the potential to produce a much better variation of the original human-designed algorithm. Runka (2009) presents such a system for evolving the edge selection formula of an ant colony optimisation algorithm. While the evolved formulae were tested on only two unseen travelling salesman problem instances, the results were better than the original human-designed formula.

5.5. Timetabling and scheduling

A grammar-based genetic programming system is presented by Bader-El-Den *et al* (2009) for the exam timetabling problem. The grammar contains elements of graph colouring heuristics and slot allocation heuristics, and a sentence in this grammar represents a new heuristic for constructing a timetable. The results returned by the evolved heuristics are comparable with a range of human-created search methodologies from the literature. The system is presented as an online learning methodology, as it is not shown whether these evolved heuristics are successful when reused on new problem instances.

The exam timetabling approach of Pillay (2009) evolves a heuristic to order exams. A sorting algorithm, such as quicksort, has a comparator which decides if one element should be ahead of another in the list, and then it is applied to as many pairs of elements as is necessary to create the ordering based on that comparator. Pillay uses strongly typed genetic programming to evolve the comparator of two elements, which in this case are exams. The comparator has a tree structure consisting of standard graph colouring indicators such as largest weighted degree and saturation degree of the two exams, and logic operators such as ‘less than’ and ‘not equal to’. After sorting, the exam that appears at the head of the queue is scheduled next.

5.6. Summary and discussion

This section presents a summary of the literature on heuristic generation methodologies. Investigations have been undertaken on a wide variety of optimisation problems, which have relied on human-generated heuristics thus far. The literature shows that, typically, evolutionary computation methods are employed to automatically generate heuristics, which are reusable on new problem instances.

A heuristic generation process is often computationally expensive when compared with a methodology that operates directly on the solution space. However, this is only a disadvantage in the short term, when results will not be required for future problems. Consider the application of an evolutionary algorithm directly to the problem space. The output is just the solution to the instance, and the entire evolutionary algorithm must be run again if a solution is required for future problems. If the evolutionary process is employed instead as a hyper-heuristic, to generate a quick reusable heuristic, then only one run of the evolution is required. The evolved heuristic can then obtain a comparable result on the future problems, much more quickly than the application of an evolutionary algorithm. This is one of the main benefits of searching for a solution *method* rather than just searching for a solution.

While the evolution process is computationally expensive, it is often quicker than manual heuristic generation. For example, Geiger *et al* (2006) state that production scheduling heuristics from the literature are the result of years of scheduling research, and the identical evolved heuristic rules are generated within a fraction of this time. This illustrates one of the main motivations for automatically generating heuristics.

However, the potential components of the evolved heuristics must still be defined by humans, and the consensus from current research seems to be that the set of components will be different for each problem domain. Research in this new area of automatic heuristic generation shows that it is not yet able to completely replace human ingenuity, as it can be argued that successful sets of components are inspired by the literature on human-created heuristics. As Fukunaga (2008) states, humans are able to invent good building blocks, and the literature *does* show that hyper-heuristic methodologies have been able to successfully combine these human-defined building blocks in superior ways.

6. Related areas

Heuristic search is widely studied in Operational Research, Computer Science and Artificial Intelligence. A promising direction for developing improved search techniques is to integrate learning components that can adaptively guide the search. Many techniques have independently arisen in recent years that exploit either some form of learning, or search on a configuration space, to improve problem-solving and decision-making. We briefly overview some of these approaches below, categorising them between offline and online approaches.

6.1. Offline approaches

Algorithm configuration: Is concerned with determining the appropriate values for algorithm parameters. It is commonly treated as an optimisation problem (and therefore as a search problem in a configuration space), where the objective function captures performance on a fixed set of benchmark instances (Hutter *et al*, 2007). Depending on the number and type of parameters, the methods used to solve this optimisation problem include exhaustive enumeration, beam search (Minton, 1996), experimental design (Ridge and Kudenko, 2007), the application of racing algorithms (Birattar, 2005), combinations of fractional experimental design and local search (Adenso-Diaz and Laguna, 2006), and iterated local search (Hutter *et al*, 2009).

Meta-learning for algorithm selection: The algorithmic selection problem was formulated by Rice (1976) as: *Which algorithm is likely to perform best for my problem?* Recognising this problem as a learning task, the machine learning community developed meta-learning approaches mainly to learn about classification. The generalisation of meta-learning concepts to constraint satisfaction and optimisation as discussed in Smith-Miles (2008b) is closely related to hyper-heuristic research. Examples of these generalisations can be found in Operational Research (Smith-Miles, 2008a) and Artificial Intelligence (Horvitz *et al*, 2001).

6.2. Online approaches

Parameter control in evolutionary algorithms: A different approach to algorithm configuration, also called parameter control, is the idea of tuning algorithm parameters online at execution time. These approaches were already discussed in Section 2. In Eiben *et al* (1999), a useful and widely accepted classification of mechanisms for parameter control is proposed, together with a detailed literature survey of work to date in this topic within evolutionary algorithms. More recent surveys and overviews of the state-of-the-art can be found in Eiben *et al* (2007) and Lobo *et al* (2007).

Adaptive memetic algorithms: Another approach closely related to heuristic selection based on perturbative heuristics is that of *adaptive memetic algorithms*, a breed of hybrid evolutionary algorithms, in which several memes (or local searchers) are available and adaptively selected (or generated altogether) during the search (Krasnogor and Smith, 2000; Krasnogor and Gustafson, 2004; Ong and Keane, 2004; Jakob, 2006; Ong *et al*, 2006). Ong *et al* (2006) present a useful classification of memes adaptation in memetic algorithms based on the widely accepted terminology proposed in Eiben *et al* (1999). As discussed in Smith (2008), self-adaptation ideas have been applied to memetic algorithms in two ways: (i) for self-adapting the choice of local search operators (Krasnogor and Smith, 2000), and (ii) for self-adapting the definition of local search algorithms (Smith, 2002; Krasnogor and Gustafson, 2004). This distinction is analogous to our main classification of hyper-heuristics into heuristic selection and heuristic generation methodologies (see Section 3).

Adaptive operator selection: A related recent research direction, again within evolutionary algorithms, has been termed *adaptive operator selection*. Its goal is to design online strategies able to autonomously select between different variation operators. As discussed in Maturana *et al* (2009), adaptive operator selection approaches contain two main mechanisms: *credit assignment*, which defines the reward to be assigned to an operator (according to its quality) after it has been applied, and *operator selection* that selects one operator to be applied according to previously computed operator qualities. Several approaches for implementing these two mechanisms have been proposed (Fialho *et al*, 2008; Maturana *et al*, 2009, 2010; Candan *et al*, 2012; Veerapen *et al*, 2012).

Reactive search: is an online methodology that advocates the integration of sub-symbolic machine learning techniques into search heuristics for solving complex optimisation problems (Battiti, 1996; Battiti *et al*, 2009). The machine learning component acts on top of the search heuristic, in order to let the algorithm self-tune its operating parameters during the search operation. The learning component is implemented as a reactive feedback scheme that uses the past history of the search to increase its efficiency and efficacy. These ideas have been mainly applied to the tabu search meta-heuristic.

Variable neighbourhood search: Although generally not including an adaptive mechanism, Variable Neighbourhood search (VNS) (Mladenovic and Hansen, 1997) is related to heuristic selection based on perturbative heuristics in that such a method exploits the search power of multiple neighbourhoods. VNS systematically switches neighbourhoods in a predefined sequence so that the search can explore increasingly distant neighbourhoods of the current solution. Therefore, we can say that VNS is a high-level heuristic that coordinates the behaviour of several neighbourhood structures.

Algorithm Portfolios: First proposed in Huberman *et al* (1997), algorithm portfolios represent an alternative way of automating the design of search techniques. They are designed following the standard practice in economics to obtain different return-risk profiles in the stock market by combining different stocks. An algorithm portfolio would run different algorithms concurrently in a time-sharing manner, by allocating a fraction of the total CPU cycles to each of them. The first algorithm to finish reports the solution and determines the completion time of the portfolio, while the other algorithms are immediately stopped. Dynamic portfolios, that include online learning, have been considered in Gagliolo and Schmidhuber (2006).

7. Discussion and future work

The defining feature of hyper-heuristics is that they operate on a search space of heuristics rather than directly on a search space of problem solutions. This feature provides the potential for increasing the level of generality of search methodologies. There have been several independent realisations of this idea over the years (since the early 1960s) within Operational Research, Computer Science and Artificial Intelligence; however, the term hyper-heuristic is relatively new. We identified two main broad classes of approaches to the challenge of automating the design of heuristics; namely: heuristic selection and heuristic generation. This article has covered both the intellectual roots and the state of-the-art of these methodologies up until the end of 2010.

Heuristic generation methodologies offer more scope for greater levels of generalisation. However, they can be more difficult to implement, when compared with their counterpart (heuristic selection methodologies) since they require the decomposition of existing heuristics, and the design of an appropriate framework. These issues are balanced by the potential benefits of the approach. One of the strengths of heuristic generation methodologies is that they can automatically specialise a heuristic to a given class of problem instances. This process of specialisation and tuning is usually the time-consuming and expensive part of the implementation of a heuristic, and so hyper-heuristics have the potential to save a significant amount of effort. In turn this could reduce the cost barriers that prevent smaller organisations from taking advantage of modern search technologies.

An additional criterion for classifying hyper-heuristics is the source of the feedback during the learning process, which can be either one instance (online approaches) or many instances of the underlying problem (offline approaches). Both online and offline approaches are potentially useful and therefore worth investigating. Although having a reusable method will increase the speed of solving new instances of a given problem, using online methods can have other advantages. In particular, searching over a space of heuristics may be more effective than directly searching the underlying problem space, as heuristics may provide an advantageous search space structure (Storer *et al*, 1995; Ochoa *et al*, 2009b; Vázquez-Rodríguez and Petrovic, 2010). Moreover, in newly encountered problems there may not be a set of related instances on which to train offline hyper-heuristic methods.

Hyper-heuristics can be used for solving complex real-world problems. Since the search strategy components of a hyper-heuristic process only problem domain independent information, they can be readily applied in a different problem domain (provided that the problem-specific algorithm components are available to the practitioner). The studies on parallel and distributed processing strategies can benefit from hyper-heuristic research and vice versa. Hyper-heuristic methodologies can handle both single and multi-objective problems. The empirical investigations up to now show that hyper-heuristics are fast techniques that produce solutions with *reasonable* quality in a *reasonable* time. Moreover, their performance is often comparable with bespoke systems.

The further development of hyper-heuristic frameworks such as HyFlex (Ochoa *et al*, 2012b) and Hyperion (Swan *et al*, 2011) may help to promote research and thus improve hyper-heuristic methods. It is still an open problem how to easily apply these software frameworks in practice for new domains.

Thus far, little progress has been made to enhance our theoretical understanding of hyper-heuristic approaches. Initial efforts have been devoted to understanding the structure of heuristic search spaces (Vázquez-Rodríguez *et al*, 2007b; Qu and Burke, 2009; Ochoa *et al*, 2009a; Maden *et al*, 2009), and the implications to the Non-Free-Lunch theorem (Poli and Graff, 2009). Further research in this direction, including run time analysis and other foundational studies, would be relevant to both enhancing our understanding and designing efficient and general hyper-heuristics.

As it was discussed in Section 6 several communities have been working in related research themes and sharing common goals. However, there is still little interaction between them. Much is to be gained from a greater awareness of the achievements in various cross-disciplinary approaches; opportunities would open for extension to both new problem domains and new methodologies through cross-fertilisation of these ideas. Hyper-heuristic research has the potential of bringing together promising ideas in the fields of meta-heuristics and machine learning, with knowledge (in the form of problem-specific heuristics) accumulated over the years in the field of operational research. The overall aim is to solve complex real-life combinatorial optimisation problems in a more general fashion, and produce reusable technologies to facilitate systems which can work with users to home in on high-quality solutions to problems.

References

Adenso-Diaz B and Laguna M (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research* **54**(1): 99–114.

Ahmadi S, Barrone P, Cheng P, Burke EK, Cowling P and McCollum B (2003). Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. In: Kendall G, Burke EK, Petrovic S and Gendreau M (eds). *Multidisciplinary International Scheduling: Theory and Applications*. MISTA, Springer: New York, pp 155–171.

Allen S, Burke EK, Hyde MR and Kendall G (2009). Evolving reusable 3d packing heuristics with genetic programming. In: Rothlauf F (ed). *Genetic and Evolutionary Computation Conference (GECCO '09)*. ACM: New York, NY, pp 931–938.

Antunes CH, Lima P, Oliveira E and Pires DF (2009). A multi-objective simulated annealing approach to reactive power compensation. *Engineering Optimization* **43**(10): 1063–1077.

Asmuni H, Burke EK, Garibaldi JM and McCollum B (2005). Fuzzy multiple heuristic orderings for examination timetabling. *Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling, PATAT'04*. Springer-Verlag: Berlin, Hiedelberg.

Ayob M and Kendall G (2003). A Monte Carlo hyper-heuristic to optimize component placement sequencing for multi head placement machine. In: *Proceedings of the International Conference on Intelligent Technologies (InTech'03)*, Institute for science and Technology Research and Development: Chiang Mai, Thailand, pp 132–141.

Bader-El-Den M and Poli R (2007). Generating sat local-search heuristics using a GP hyper-heuristic framework. In: Monmarche N, Talbi E-G, Collet P, Schoenauer M and Lutton E (eds). *International Conference on Artificial Evolution*. Lecture Notes in Computer Science, Springer-Verlag: Berlin, Heidelberg, pp 37–49.

Bader-El-Den M and Poli R (2008). An incremental approach for improving local search heuristics. In: van Hemert JI and Cotta C (eds). *European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP'08), Lecture Notes in Computer Science*. Springer: Naples, Italy, pp 194–205.

Bader-El-Den M, Poli R and Fatima S (2009). Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. *Memetic Computing* **1**(3): 205–219.

Bai R and Kendall G (2005). An investigation of automated planograms using a simulated annealing based hyper-Heuristics. In: Ibaraki T, Nonobe K and Yagiura M (eds). *Metaheuristics: Progress as Real Problem Solver—(Operations Research/Computer Science Interface Series)*. Vol. 32, Springer: New York, pp 87–108.

Bai R and Kendall G (2008). A model for fresh produce shelf-space allocation and inventory management with freshness-condition-dependent demand. *INFORMS Journal on Computing* **20**(1): 78–85.

Bai R, Burke EK, Kendall G and McCollum B (2007). Memory length in hyper-heuristics: An empirical study. In: Kendall G, Tan KC, Burke EK and Smith S (eds) *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CISched2007)*. IEEE: Honolulu, HI.

Bai R, Burke EK and Kendall G (2008). Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. *Journal of the Operational Research Society* **59**(10): 1387–1397.

Bai R, Blazewicz J, Burke EK, Kendall G and McCollum B (2012). A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR: A Quarterly Journal of Operations Research* **10**(1): 43–66.

Battiti R (1996). Reactive search: Toward self-tuning heuristics. In: Rayward-Smith VJ, Osman IH, Reeves CR and Smith GD (eds). *Modern Heuristic Search Methods*. John Wiley & Sons Ltd.: Chichester, pp 61–83.

Battiti R, Brunato M and Mascia F (2009). *Title: Reactive Search and Intelligent Optimization*. Operations Research/Computer Science Interfaces Series, Vol. 45, Springer: New York.

Berberoğlu A and Uyar Ac (2010). A hyper-heuristic approach for the unit commitment problem. In: Di Chio C, Brabazon A, Di Caro G, Ebner M, Farooq M, Fink A, Grahl J, Greenfield G, Machado P, ONeill M, Tarantino E and Urquhart N (eds). *Applications of Evolutionary Computation*. Lecture Notes in Computer Science Springer: Berlin/Heidelberg, pp 121–130.

Berberoglu A and Uyar AS (2011). Experimental comparison of selection hyper-heuristics for the short-term electrical power generation scheduling problem. In: Di Chio C, *et al* (eds). *Proceedings of Evo-Applications 2011, Part II, LNCS*. Springer: Berlin/Heidelberg, pp 444–453.

Bhanu SMS and Gopalan N (2008). A hyper-heuristic approach for efficient resource scheduling in grid. *International Journal of Computers, Communications & Control* **III**(3): 249–258.

Biazzini M, Banhelyi B, Montresor A and Jelasity M (2009). Distributed hyper-heuristics for real parameter optimization. In: Rothlauf F (ed). *Genetic and Evolutionary Computation Conference (GECCO 2009)*. ACM: New York, NY, pp 1339–1346.

Bilgin B, Özcan E and Korkmaz EE (2006). An experimental study on hyper-heuristics and final exam scheduling. In: *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT'06)*, pp 123–140.

Birattar M (2005). Tuning Metaheuristics: A Machine Learning Perspective. Studies in Computational Intelligence, Vol. 197, Springer: New York.

Blazewicz J, Burke E, Kendall G, Mruczkiewicz W, Oguz C and Swiercz A (2011). A hyper-heuristic approach to sequencing by hybridization of dna sequences. *Annals of Operations Research*, pp 1–15, DOI 10.1007/s10479-011-0927-y.

Branke J (2002). *Evolutionary Optimization in Dynamic Environments*. Kluwer: Dordrecht, MA.

Burke EK and Bykov Y (2008). A late acceptance strategy in hill-climbing for exam timetabling problems. In: *Practice and Theory of Automated Timetabling* (PATAT 2008).

Burke EK, Hart E, Kendall G, Newall J, Ross P and Schulenburg S (2003a). Hyperheuristics: An emerging direction in modern search technology. In: Glover F and Kochenberger G (eds). *Handbook of Metaheuristics*. Kluwer: Dordrecht, MA, pp 457–474.

Burke EK, Kendall G and Soubeiga E (2003b). A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics* **9**(6): 451–470.

Burke EK, Dror M, Petrovic S and Qu R (2005a). The next wave in computing, optimization, and decision technologies. chap Hybrid Graph Heuristics within a Hyper-heuristic Approach to Exam Timetabling Problems, Springer: New York, pp 79–91.

Burke EK, Kendall G, Landa-Silva JD, O'Brien R and Soubeiga E (2005b). An ant algorithm hyperheuristic for the project presentation scheduling problem. In: *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, Vol. 3, pp 2263–2270.

Burke EK, Landa-Silva JD and Soubeiga E (2005c). Meta-heuristics: Progress as real problem solvers. chap *Multi-objective Hyper-heuristic Approaches for Space Allocation and Timetabling*, Springer: New York, pp 129–158.

Burke EK, Hyde MR and Kendall G (2006a). Evolving bin packing heuristics with genetic programming. In: *Parallel Problem*

Solving from Nature (PPSN'06), Lecture Notes in Computer Science, Vol. 4193, pp 860–869.

Burke EK, Petrovic S and Qu R (2006b). Case-based heuristic selection for timetabling problems. *Journal of Scheduling* **9**(2): 115–132.

Burke EK, Hyde M, Kendall G and Woodward J (2007a). The scalability of evolved on line bin packing heuristics. In: *IEEE Congress on Evolutionary Computation (CEC'07)*, Singapore, pp 2530–2537.

Burke EK, Hyde M, Kendall G and Woodward J (2007b). Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one. In: *Genetic and Evolutionary Computation Conference (GECCO'07)*, ACM, pp 1559–1565.

Burke EK, McCollum B, Meisels A, Petrovic S and Qu R (2007c). A graph-based hyperheuristic for educational timetabling problems. *European Journal of Operational Research* **176**: 177–192.

Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E and Woodward J (2009). Exploring hyper-heuristic methodologies with genetic programming. In: Mumford C and Jain L (eds). *Computational Intelligence: Collaboration, Fusion and Emergence, Intelligent Systems Reference Library*. Springer: New York, pp 177–201.

Burke E, Kendall G, Mısır M and Özcan E (2010a). Monte Carlo hyper-heuristics for examination timetabling. *Annals of Operations Research* **196**: 73–90.

Burke EK, Curtois T, Hyde M, Kendall G, Ochoa G, Petrovic S, Vazquez-Rodriguez JA and Gendreau M (2010b). Iterated local search *versus* hyper-heuristics: Towards general purpose search algorithms. In: *IEEE Congress on Evolutionary Computation* (CEC 2010), pp 3073–3080.

Burke EK, Hyde M and Kendall G (2010c). Providing a memory mechanism to enhance the evolutionary design of heuristics. In: *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI'10)*, Barcelona, Spain, pp 3883–3890.

Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E and Woodward J (2010d). Handbook of meta-heuristics, international series in operations research & management science. Vol. 146, chap *A Classification of Hyper-heuristic Approaches*, Springer: New York, pp 449–468. Chapter 15.

Burke EK, Hyde MR, Kendall G and Woodward J (2010e). A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics. *IEEE Transactions on Evolutionary Computation* **14**(6): 942–958.

Burke E, Gendreau M, Ochoa G and Walker J (2011a). Adaptive iterated local search for cross-domain optimisation. In: *Genetic and Evolutionary Computation* (GECCO 2011), ACM, pp 1987–1994.

Burke EK, Hyde MR, Kendall G and Woodward J (2011b). Automating the packing heuristic design process with genetic programming. *Evolutionary Computation* **20**(1): 63–89.

Burke EK, Hyde MR and Kendall G (2012). Grammatical evolution of local search heuristics. *IEEE Transactions on Evolutionary Computation* **16**: 406–417.

Candan C, Goeffon A, Lardeux F and Saubion F (2012). An exploration-exploitation compromise-based adaptive operator selection for local search. In: Soule T and Moore JH (eds). *Genetic and Evolutionary Computation (GECCO 2012)*. ACM: New York, NY, pp 1253–1260.

Cano-Belmáan J, Ŕ R and Bautista J (2010). A scatter search based hyperheuristic for sequencing a mixed-model assembly line. *Journal of Heuristics* **16**(6): 749–770.

Cerny V (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications* **45**(1): 41–51.

Chakhlevitch K and Cowling P (2005). Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In: Raidl GR and Gottlieb J (eds). *Proceedings of 5th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP2005)*. Lecture Notes in Computer Science, Springer: New York, pp 25–33.

Chakhlevitch K and Cowling P (2008). Hyperheuristics: Recent developments. In: Cotta C, Sevaux M and Sörensen K (eds). *Adaptive and Multilevel Metaheuristics, Studies in Computational Intelligence*. Vol. 136, Springer: New York, pp 3–29.

Chan C, Xue F, Ip W and Cheung C (2012). A hyper-heuristic inspired by pearl hunting. In: Hamadi Y and Schoenauer M (eds) *International Conference on Learning and Intelligent Optimization (LION 6), Lecture Notes in Computer Science*. Vol. 7219, Springer Berlin Hiedelberg: Paris, France, pp 349–353.

Chen PC, Kendall G and Vanden-Berghe G (2007). An ant based hyper-heuristic for the travelling tournament problem. In: Kendall G, Tan KC, Burke EK and Smith S (eds) *Proceedings of IEEE Symposium of Computational Intelligence in Scheduling (CISched 2007)*. IEEE: Honolulu, HI, pp 19–26.

Cheng P, Barone R, Ahmadi S and Cowling P (2003). Integrating human abilities with the power of automated scheduling systems: Representational epistemological interface design. In: Kortenkamp D and Freed M (eds) *AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments*. American Association for Artificial Intelligence: Palo Alto, CA, pp 23–29.

Cobos C, Mendoza M and Leon E (2011). A hyper-heuristic approach to design and tuning heuristic methods for web document clustering. In: *Evolutionary Computation (CEC)*. IEEE: New Orleans, LA, pp 1350–1358.

Corne D and Ross P (1996). Peckish initialisation strategies for evolutionary timetabling. In: Burke E and Ross P (eds). *Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science*. Vol. 1153, Springer: Berlin/Heidelberg, pp 227–240.

Cowling P and Chakhlevitch K (2003). Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2003)*, IEEE Computer Society Press: Canberra, Australia, pp 1214–1221.

Cowling P, Kendall G and Soubeiga E (2000). A hyperheuristic approach to scheduling a sales summit. In: Burke EK and Erben W (eds). *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling. PATAT 2000*, Lecture Notes in Computer Science, Springer: Konstanz, Germany, pp 176–190.

Cowling P, Kendall G and Soubeiga E (2001). A parameter-free hyperheuristic for scheduling a sales summit. In: Sousa JP and Resende MGC (eds). *Proceedings of the 4th Metaheuristic International Conference*. Springer; Porto, Portugal, pp 127–131.

Cowling P, Kendall G and Han L (2002a). An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem. In: Wang L *et al* (eds). *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*. Orchid Country Club: Singapore, pp 267–271.

Cowling P, Kendall G and Han L (2002b). An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: *Proceedings of the Congress on Evolutionary Computation (CEC2002)*. IEEE Computer Society: Honolulu, Hawaii, USA, pp 1185–1190.

Cowling P, Kendall G and Soubeiga E (2002c). Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In: Cagoni S, Gottlieb J, Hart E, Middendorf M and Goenther R (eds). *Applications of Evolutionary Computing: Proceeding of Evo*

Workshops 2002. Lecture Notes in Computer Science, Vol, 2279, Springer-Verlag: Kinsale, Ireland, pp 1–10.

Crainic T and Toulouse M (2003). Parallel strategies for meta-heuristics. In: Glover F and Kochenberger G (eds). *Handbook in Meta-heuristics*. Kluwer Academic Publishers: Dordrecht, MA, pp 475–513.

Crowston WB, Glover F, Thompson GL and Trawick JD (1963). Probabilistic and parametric learning combinations of local job shop scheduling rules. ONR Research memorandum, GSIA, Carnegie Mellon University, Pittsburgh 1(117).

Cruz C, Gonzalez J and Pelta D (2011). Optimization in dynamic environments: A survey on problems, methods and measures. *Soft Computing—A Fusion of Foundations, Methodologies and Applications* **15**(7): 1427–1448.

Demeester P, Bilgin B, Causmaecker PD and Berghe GV (2012). A hyperheuristic approach to examination timetabling problems: Benchmarks and a new problem from practice. *Journal of Scheduling* **15**(1): 83–103.

Denzinger J, Fuchs M and Fuchs M (1996). *High performance ATP systems by combining several ai methods*. Technical Report, SEKI-Report SR-96-09, University of Kaiserslautern.

DiGaspero L and Schaerf A (2007). Easysyn++: A tool for automatic synthesis of stochastic local search algorithms. In: Stützle T, Birattari M, Hoos HH (eds) *International Workshop on Engineering Stochastic Local Search Algorithms (SLS 2007)*. Lecture Notes in Computer Science, Springer: Brussels, Belgium, pp 177–181.

Dimopoulos C and Zalzala AMS (2001). Investigating the use of genetic programming for a classic one-machine scheduling problem. *Advances in Engineering Software* **32**(6): 489–498.

Dorndorf U and Pesch E (1995). Evolution based learning in a job shop scheduling environment. *Computers and Operations Research* **22**(1): 25–40.

Dowsland KA, Soubeiga E and Burke EK (2007). A simulated annealing hyperheuristic for determining shipper sizes for storage and transportation. *European Journal of Operational Research* **179**(3): 759–774.

Drake J, Özcan E and Burke E (2012). An improved choice function heuristic selection for cross domain heuristic search. In: Coello Coello CA, Cutello V, Deb K, Forrest S, Nicosia G and Pavone M (eds). *Parallel Problem Solving from Nature – PPSN XII*. Lecture Notes in Computer Science, Springer: Taormina, Italy, pp 307–316.

Drechsler R and Becker B (1995). Learning heuristics by genetic algorithms. In: Shirakawa I (ed). *ASP Design Automation Conference*. ACM: Makuhari, Massa, Chiba, Japan, pp 349–352.

Drechsler R, Göckel N and Becker B (1996). Learning heuristics for obdd minimization by evolutionary algorithms. In: Voigt H-M, Ebeling W, Rechenberger I and Schwefel H-P (eds). *Parallel Problem Solving from Nature (PPSN'96)*. Lecture Notes in Computer Science, Springer: Berlin, Germany, pp 730–739.

Dueck G (1993). New optimization heuristics the great deluge algorithm and the record to-record travel. *Journal of Computational Physics* **104**(1): 86–92.

Eiben AE, Hinterding R and Michalewicz Z (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* **3**(2): 124.

Eiben AE, Michalewicz Z, Schoenauer M and Smith JE (2007). Parameter setting in evolutionary algorithms. In: Lobo FJ, Lima CF and Michalewicz Z (eds). Chap *Parameter Control in Evolutionary Algorithms*. Springer: New York, pp 19–46.

Elyasaf A, Hauptman A and Sipper M (2012). Evolutionary design of Freecell Solvers. *IEEE Transactions On Computational Intelligence and AI in Games* **4**(4): 270–281.

Ersoy E, Özcan E and Etaner-Uyar AS (2007). Memetic algorithms and hyperhill-climbers. In: Baptiste P, Kendall G, Kordon AM and Sourd F (eds). *Proceedings of the 3rd Multidisciplinary International Scheduling Conference on Scheduling: Theory and Applications*, MISTA: Paris, France, pp 28–31.

Fang H, Ross P and Corne D (1993). A promising genetic algorithm approach to job shop scheduling, rescheduling, and open-shop scheduling problems. In: Cohn AG (ed). *International Conference on Genetic Algorithms*. John Wiley and Sons: Amsterdam, the Netherlands, pp 375–382.

Fang H, Ross P and Corne D (1994). A promising hybrid ga/heuristic approach for openshop scheduling problems. In: Cohn AG (ed). *European Conference on Artificial Intelligence*. John Wiley & Sons: New York.

Feo TA and Resende MGC (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization* **6**(2): 109–133.

Fialho A, Costa L, Schoenauer M and Sebag M (2008). Extreme value based adaptive operator selection. In: Rudolph G, Jansen T, Lucas SM, Poloni C and Beume N (eds). *Industrial Scheduling*. Lecture Notes in Computer Science, Springer: Dortmund, Germany, pp 175–184.

Fisher H and Thompson GL (1963). Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF and Thompson GL (eds). *Industrial Scheduling*. Prentice-Hall: Englewood Cliffs, NJ, pp 225–251.

Freisleben B and Härtfelder M (1993). Optimization of genetic algorithms by genetic algorithms. In: Albrecht RF, Steele NC and Reeves CR (eds). *Artificial Neural Nets and Genetic Algorithms*. Springer Verlag: Berlin, pp 392–399.

Fukunaga AS (2002). Automated discovery of composite sat variable-selection heuristics. In: Dechter R and Sutton RS (eds). *National Conference on Artificial Intelligence*. AAAI Press: Edmonton, Alberta, Canada, pp 641–648.

Fukunaga AS (2004). Evolving local search heuristics for SAT using genetic programming. In: Deb K (ed). *Genetic and Evolutionary Computation Conference (GECCO '04)*. ISGEC, Lecture Notes in Computer Science, Springer: Berlin, pp 483–494.

Fukunaga AS (2008). Automated discovery of local search heuristics for satisfiability testing. *Evolutionary Computation (MIT Press)* **16**(1): 31–1.

Gagliolo M and Schmidhuber J (2006). Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence* **47**(3–4): 295–328.

Garcia-Villoria A, Salhi S, Corominas A and Pastor R (2011). Hyper-heuristic approaches for the response time variability problem. *European Journal of Operational Research* **211**(1): 160–169.

Garrido P and Castro C (2009). Stable solving of cvrps using hyperheuristics. In: Rothlauf F (eds). *Genetic and Evolutionary Computation Conference (GECCO'09)*. ACM: Montreal, Canada, pp 255–262.

Garrido P and Riff MC (2007a). Collaboration between hyperheuristics to solve strippacking problems. In: Ochoa G and Özcan E (eds). *12th International Fuzzy Systems Association World Congress, Proceedings*. Lecture Notes in Computer Science, Springer: New York, pp 698–707.

Garrido P and Riff MC (2007b). An evolutionary hyperheuristic to solve strip-packing problems. In: Yin H, Tino P, Corchado E, Byrne W and Yao X (eds). *Intelligent Data Engineering and Automated Learning—IDEAL 2007 Proceedings*. Lecture Notes in Computer Science, Springer: Berlin, pp 406–415.

Garrido P and Riff MC (2010). Dvrp: A hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics* **16**(6): 795–834.

Gaspero LD and Urli T (2012). Evaluation of a family of reinforcement learning crossdomain optimization heuristics.

In: Hamadi Y and Schoenauer M (eds). *International Conference on Learning and Intelligent Optimization (LION 6)*. Lecture Notes in Computer Science, Springer: Berlin, pp 384–389.

Geiger CD, Uzsoy R and Aytug H (2006). Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. *Journal of Scheduling* **9**(1): 7–34.

Gibbs J, Kendall G and Özcan E (2010). Scheduling English football fixtures over the holiday period using hyper-heuristics. In: Schaefer R, Cotta C, Kolodziej J and Rudolph G (eds). *Parallel Problem Solving from Nature, PPSN'10*. Springer-Verlag: Berlin, pp 496–505.

Goldberg DE, Korb B and Deb K (1990). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems* **3**(5): 493–530.

Gratch J and Chien S (1996). Adaptive problem-solving for large-scale scheduling problems: A case study. *Journal of Artificial Intelligence Research* **4**(1): 365–396.

Gratch J, Chien S and DeJong G (1993). Learning search control knowledge for deep space network scheduling. In: *International Conference on Machine Learning*. Morgan Kaufmann: Amherst, MA, pp 135–142.

Grefenstette J (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics SMC* **16**(1): 122–128.

Grobler J, Engelbrecht A, Kendall G and Yadavalli V (2012). Investigating the use of local search for improving meta-hyper-heuristic performance. In: *Evolutionary Computation (CEC)*. IEEE: Brisbane, Australia, pp 1–8.

Han L and Kendall G (2003). Guided operators for a hyper-heuristic genetic algorithm. In: Gedeon TD and Fung LCC (eds). *The 16th Australian Conference on Artificial Intelligence (AI 03)*. Springer: Perth, Australia, pp 807–820.

Hart E and Ross P (1998). A heuristic combination method for solving job-shop scheduling problems. In: Eiben AE, Bäck T, Schoenauer M and Schwefel H-P (eds). *Parallel Problem Solving from Nature, PPSN V*. Lecture Notes in Computer Science, Springer: Amsterdam, the Netherlands, pp 845–854.

Hart E, Ross P and Nelson JAD (1998). Solving a real-world problem using an evolving heuristically driven schedule builder. *Evolutionary Computing* **6**(1): 61–80.

He J, He F and Dong H (2012). Pure strategy or mixed strategy?—An initial comparison of their asymptotic convergence rate and asymptotic hitting time. In: Hao JK and Middendorf M (eds). *Evolutionary Computation in Combinatorial Optimization—12th European Conference, EvoCOP 2012*. Málaga, Spain, 11–13 April 2012. Proceedings, Lecture Notes in Computer Science, Vol. 7245, Springer, pp 218–229.

Ho NB and Tay JC (2005). Evolving dispatching rules for solving the flexible job-shop problem. In: *IEEE Congress on Evolutionary Computation (CEC'05)*. IEEE: Edinburgh, UK, pp 2848–2855.

Hoos HH and Stützle T (2004). *Stochastic Local Search: Foundations and Applications*. Elsevier/Morgan Kaufmann: San Francisco, CA.

Horvitz E, Ruan Y, Gomes C, Kautz H, Selman B and Chickering D (2001). A bayesian approach to tackling hard computational problems. In: Breese JS and Koller D (eds). *Conference in Uncertainty in Artificial Intelligence, UAI '01*. Morgan Kaufman: Los Altos, CA, pp 235–244.

Huberman BA, Lukose RM and Hogg T (1997). An economics approach to hard computational problems. *Science* **275**(5296): 51–54.

Hutter F, Hoos HH and Stützle T (2007). Automatic algorithm configuration based on local search. In: Cohn A (ed). *Proceedings of the Twenty-second AAAI Conference on Artificial Intelligence*. AAAI Press: Vancouver, British Columbia, Canada, pp 1152–1157.

Hutter F, Hoos H, Leyton-Brown K and Stützle T (2009). Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research (JAIR)* **36**(1): 267–306.

Jakob W (2006). Towards an adaptive multimeme algorithm for parameter optimisation suiting the engineers' needs. In: Runarsson TP, Beyer H-G, Burke E, Merelo-Guervós JJ, Whitley LD, Yao X (eds). *Parallel Problem Solving from Nature (PPSN IX)*. Lecture Notes in Computer Science, Springer: Berlin, pp 132–141.

Jakobovic D, Jelenkovic L and Budin L (2007). Genetic programming heuristics for multiple machine scheduling. In: Ebner M, O'Neill M, Ekárt A, Vanneschi L, Esparcia-Alcázar A (eds). *European Conference on Genetic Programming (EUROGP' 07)*. Lecture Notes in Computer Science, Springer: Valencia, Spain, pp 321–330.

Kaelbling LP, Littman ML and Moore AW (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* **4**(1): 237–285.

Kampouridis M, Alsheddy A and Tsang E (2012). On the investigation of hyper-heuristics on a financial forecasting problem. *Annals of Mathematics and Artificial Intelligence*. Springer: Netherlands, 1012–2443.

Keller RE and Poli R (2007a). Cost-benefit investigation of a genetic-programming hyperheuristic. In: Monmarché N., Talbi E-G, Collet P, Schoenauer M and Lutton E (eds). *International Conference on Artificial Evolution*. Springer-Verlag: Berlin, Heidelberg, pp 13–24.

Keller RE and Poli R (2007b). Linear genetic programming of parsimonious metaheuristics. In: *IEEE Congress on Evolutionary Computation (CEC'07)*. IEEE: Singapore, pp 4508–4515.

Keller RE and Poli R (2008a). Self-adaptive hyperheuristic and greedy search. In: *IEEE World Congress on Computational Intelligence (WCCI'08)*. IEEE: Hong Kong, China, pp 3801–3808.

Keller RE and Poli R (2008b). Subheuristic search and scalability in a hyperheuristic. In: Ryan C and Keijzer M (eds). *Genetic and Evolutionary Computation Conference (GECCO'08)*. ACM: Atlanta, GA, pp 609–610.

Keller RE and Poli R (2008c). Toward subheuristic search. In: *IEEE World Congress on Computational Intelligence (WCCI'08)*. IEEE: Hong Kong, China, pp 3148–3155.

Kendall G and Mohamad M (2004a). Channel assignment in cellular communication using a great deluge hyper-heuristic. In: *Proceedings of the 2004 IEEE International Conference on Network (ICON2004)*. IEEE: Singapore, pp 769–773.

Kendall G and Mohamad M (2004b). Channel assignment optimisation using a hyperheuristic. In: *Proceedings of the 2004 IEEE Conference on Cybernetic and Intelligent Systems (CIS2004)*. IEEE: Singapore, pp 790–795.

Kenyon C (1996). Best-fit bin-packing with random order. In: Tardos É (ed). *Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM\SIAM: Atlanta, Georgia, pp 359–364.

Kiraz B and Topcuoglu HR (2010). Hyper-heuristic approaches for the dynamic generalized assignment problem. In: *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*. IEEE: Cairo, Egypt, pp 1487–1492.

Kiraz B, Uyar Ac and Özcan E (2011). An investigation of selection hyper-heuristics in dynamic environments. In: *Proceedings of the 2011 International Conference on Applications of Evolutionary Computation*—Volume Part I, Springer-Verlag: Berlin, Heidelberg, EvoApplications'11, pp 314–323.

Kirkpatrick S, Gelatt CD and Vecchi MP (1983). Optimization by simulated annealing. *Science* **220**(4598): 671–680.

Köle M, Etaner-Uyar A, Kiraz B and Özcan E (2012). Heuristics for car setup optimisation in torcs. In: De Wilde P, Coghill GM and Kononova AV (eds). *Computational Intelligence (UKCI), 2012 12th UK Workshop on*. IEEE: Edinburgh, UK, pp 1–8.

Koza JR (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press: Boston, MA.

Koza JR and Poli R (2005). Genetic programming. In: Burke E and Kendall G (eds). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Kluwer: Boston, MA, pp 127–164.

Krasnogor N and Gustafson S (2004). A study on the use of 'self-generation' in memetic algorithms. *Natural Computing* **3**(1): 53–76.

Krasnogor N and Smith JE (2000). A memetic algorithm with self-adaptive local search: TSP as a case study. In: Whitley LD, Goldberg DE, Cantú-Paz E, Spector L, Parmee IC and Beyer H-G (eds). *Genetic and Evolutionary Computation Conference (GECCO 2000)*. Morgan Kaufmann: Las Vegas, Nevada, USA.

Kumar R, Joshi AH, Banka KK and Rockett PI (2008). Evolution of hyperheuristics for the biobjective 0/1 knapsack problem by multiobjective genetic programming. In: Ryan C and Keijzer M (eds). *Genetic and Evolutionary Computation Conference (GECCO'08)*. ACM: Atlanta, Georgia, pp 1227–1234.

Kumar R, Kumar-Bal B and Rockett PI (2009). Multiobjective genetic programming approach to evolving heuristics for the bounded diameter minimum spanning tree problem. In: Franz R (ed). *Genetic and Evolutionary Computation Conference (GECCO '09)*. ACM: Montreal, Québec, Canada, pp 309–316.

Laguna M and Martí R (2003). Scatter search Methodology and Implementations in C. In: *Operations Research/Computer Science Interfaces Series*. Kluwer Academic Publishers: Boston.

Li J, Burke EK and Qu R (2011). Integrating neural network and logistic regression to underpin hyper-heuristic search. *Knowledge-Based Systems* **24**(2): 322–330.

Lobo F, Lima C and Michalewicz Z (eds). (2007). Parameter Setting in Evolutionary Algorithms, Studies in Computational Intelligence. Vol. 54, Springer: Berlin.

Lokketangen A and Olsson R (2010). Generating meta-heuristic optimization code using ADATE. *Journal of Heuristics* **16**(6): 911–930.

van Lon RR, Holvoet T, Vanden Berghe G, Wenseleers T and Branke J (2012). Evolutionary synthesis of multi-agent systems for dynamic dial-a-ride problems. In: Soule T and Moore JH (eds). *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*. GECCO Companion '12, ACM: Philadelphia, PA, pp 331–336.

Lopez-Camacho E, Terashima-Marin H, Ross P and Valenzuela-Rendon M (2010). Problem-state representations in a hyper-heuristic approach for the 2d irregular bpp. In: *Ibero-American Conference on AI, IBERAMIA* 2010, Lecture Notes in Computer Science, Vol. 6433, Springer: Berlin, pp 204–213.

Maden I, Uyar AS and Özcan E (2009). Landscape analysis of simple perturbative hyperheuristics. In: Matousek R (ed). *Mendel 2009: 15th International Conference on Soft Computing*. Springer: Brno, Czech Republic, pp 16–22.

Marín JG and Schulenburg S (2007). A hyper-heuristic framework with XCS: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients. In: *IWLCS*, Lecture Notes in Computer Science, Vol. 4399, Springer: Berlin, pp 193–218.

Maturana J, Fialho A, Saubion F, Schoenauer M and Sebag M (2009). Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In: *IEEE Congress on Evolutionary Computation, CEC '09*. IEEE: Trondheim, Norway, pp 365–372.

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

McClymont K and Keedwell EC (2011). Markov chain hyper-heuristic (MCHH): an online selective hyper-heuristic for multi-objective continuous problems. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ACM: New York, NY, USA, GECCO '11, pp 2003–2010.

Meignan D, Koukam A and Creput JC (2010). Coalition-based metaheuristic: A selfadaptive metaheuristic using reinforcement learning and mimetism. *Journal of Heuristics* **16**(6): 859–879.

Minton S (1996). Automatically configuring constraint satisfaction problems: A case study. *Constraints* **1**(1): 7–43.

Mısır M, Wauters T, Verbeeck K and Berghe GV (2009). A new learning hyper-heuristic for the traveling tournament problem. In: Voss S and Caserta M (eds). *Proceedings of the 8th Metaheuristic International Conference (MIC'09)*. Hamburg: Germany.

Mısır M, Verbeeck K, De Causmaecker P and Vanden Berghe G (2010). Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'10)*. IEEE: Barcelona, Spain, pp 2875–2882.

Mısır M, Vancroonenburg W, Verbeeck K and Berghe GV (2011). A selection hyperheuristic for scheduling deliveries of ready-mixed concrete. In: *Proceedings of MIC* 2011, pp 289–298.

Mısır M, Verbeeck K, De Causmaecker P and Vanden Berghe G (2012). An intelligent hyper-heuristic framework for chesc 2011. In: *International Conference on Learning and Intelligent Optimization* (LION 6), Lecture Notes in Computer Science, Vol. 7219, Springer: Berlin, pp 461–466.

Mladenovic N and Hansen P (1997). Variable neighborhood search. *Computers and Operations Research* **24**(11): 1097–1100.

Nareyek A (2001). An empirical analysis of weight-adaptation strategies for neighborhoods of heuristics. In: Sousa J (ed). *Metaheuristic International Conference MIC'2001*. Porto, Portugal, pp 211–215.

Nareyek A (2003). Choosing search heuristics by non-stationary reinforcement learning. In: Resende MGC and de Sousa JP (eds). *Metaheuristics: Computer Decision-Making*. chap 9, Kluwer: Boston, MA, pp 523–544.

Nguyen S, Zhang M and Johnston M (2011). A genetic programming based hyper-heuristic approach for combinatorial optimisation. In: *Genetic and Evolutionary Computation Conference (GECCO'11)*, ACM: New York, pp 1299–1306.

Norenkov I and Goodman E (1997). Solving scheduling problems via evolutionary methods for rule sequence optimization. *2nd World Conference on Soft Computing*, WSC2.

Ochoa G and Hyde M (2011). The cross-domain heuristic search challenge (CHeSC 2011). http://www.asap.cs.nott.ac.uk/chesc2011/.

Ochoa G, Qu R and Burke EK (2009a). Analyzing the landscape of a graph based hyperheuristic for timetabling problems. In: *Genetic and Evolutionary Computation Conference (GECCO 2009)*, ACM: New York, pp 341–348.

Ochoa G, Váquez-Rodríguez JA, Petrovic S and Burke EK (2009b). Dispatching rules for production scheduling: A hyper-heuristic landscape analysis. In: *IEEE Congress on Evolutionary Computation* (CEC 2009), IEEE: New York, pp 1873–1880.

Ochoa G, Hyde M, Curtois T, Vazquez-Rodriguez JA, Walker J, Gendreau M, Kendall G, Parkes AJ, Petrovic S and Burke EK (2012a). Hyflex: A benchmark framework for cross-domain heuristic search. In: *Proceedings of the 12th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'12, Lecture Notes in Computer Science, Vol. 7245, Springer-Verlag: Berlin, pp 136–147.

Ochoa G, Walker J, Hyde M and Curtois T (2012b). Adaptive evolutionary algorithms and extensions to the hyflex hyper-heuristic framework. In: *Parallel Problem Solving from Nature—PPSN XII*, Lecture Notes in Computer Science, Vol. 7492, Springer: Berlin, pp 418–427.

Oltean M (2005). Evolving evolutionary algorithms using linear genetic programming. *Evolutionary Computation* **13**(3): 387–410.

Oltean M and Dumitrescu D (2004). Evolving TSP heuristics using multi expression programming. In: *International Conference on Computational Science* (ICCS'04), Lecture Notes in Computer Science, Vol. 3037, Springer: Berlin, pp 670–673.

Oltean M and Grosan C (2003). Evolving evolutionary algorithms using multi expression programming. In: Banzhaf W, Christaller T, Dittrich P, Kim JT and Ziegler J (eds). *European Conference on Artificial Life (ECAL'03)*. Lecture Notes in Artificial Intelligence, Springer: Dortmund, Germany, pp 651–658.

Ong YS and Keane AJ (2004). Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation* **8**(2): 99–110.

Ong YS, Lim MH, Zhu N and Wong KW (2006). Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **36**(1): 141–152.

Ortiz-Bayliss J, Özcan E, Parkes A and Terashima-Marin H (2010). Mapping the performance of heuristics for constraint satisfaction. In: *IEEE Congress on Evolutionary Computation (CEC 2010)*. IEEE: Barcelona, Spain, pp 1–8.

Ortiz-Bayliss J, Terashima-Marin H, Conant-Pablos S, Özcan E and Parkes A (2012). Improving the performance of vector hyper-heuristics through local search. In: Soule T and Moore JH (eds). *Genetic and Evolutionary Computation Conference, GECCO 2012*. ACM: Philadelphia, PA, pp 1269–1276.

Ouelhadj D and Petrovic S (2010). A cooperative hyper-heuristic search framework. *Journal of Heuristics* **16**(6): 835–857.

Özcan E and Kheiri A (2011). A hyper-heuristic based on random gradient, greedy and dominance. In: *Proceedings of the 26th International Symposium on Computer and Information Sciences ISCIS2011*, Springer: Berlin.

Özcan E and Parkes AJ (2011). Policy matrix evolution for generation of heuristics. In: Krasnogor N and Lanzi PL (eds). *Genetic and Evolutionary Computation Conference (GECCO 2011)*. ACM: Dublin, Ireland, pp 2011–2018.

Özcan E, Bilgin B and Korkmaz EE (2006). Hill climbers and mutational heuristics in hyperheuristics. In: *Parallel Problem Solving from Nature* (PPSN 2006), Lecture Notes in Computer Science, Vol. 4193, Springer: Berlin, pp 202–211.

Özcan E, Bilgin B and Korkmaz EE (2008). A comprehensive survey of hyperheuristics. *Intelligent Data Analysis* **12**(1): 1–21.

Özcan E, Bykov Y, Birben M and Burke E (2009). Examination timetabling using late acceptance hyper-heuristics. In: *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2009)*. IEEE: Trondheim, Norway, pp 997–1004.

Özcan E, Mısır M, Ochoa G and Burke E (2010). A reinforcement learning – Great-deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing (IJAMC)* **1**(1): 39–59.

Pappa GL and Freitas A (2009). *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*. Springer: Berlin.

Pillay N (2008). An analysis of representations for hyper-heuristics for the uncapacitated examination timetabling problem in a genetic programming system. In: *Annual Conference of the South African Institute of Computer Scientists and Information Technologists, ACM, ACM International Conference Proceeding Series*, Vol. 338, pp 188–192.

Pillay N (2009). Evolving hyper-heuristics for the uncapacitated examination timetabling problem. In: Blazewicz J, Drozdowski M, Kendall G and McCollum B (eds). *Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA'09)*. Dublin, Ireland, pp 447–457.

Pillay N and Banzhaf W (2007). A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. In: *13th Portuguese Conference on Aritficial Intelligence (EPIA 2007)*, Lecture Notes in Computer Science, Vol. 4874, Springer: Berlin, pp 223–234.

Ping-Che H, Tsung-Che C and FLi-Chen (2012). A vns-based hyper-heuristic with adaptive computational budget of local search. In: *IEEE Congress on Evolutionary Computation*. IEEE: Brisbane, Australia, pp 1–8.

Pisinger D and Ropke S (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research* **34**(8): 2403–2435.

Poli R and Graff M (2009). There is a free lunch for hyper-heuristics, genetic programming and computer scientists. In: *Genetic Programming, 12th European Conference*, EuroGP 2009, Lecture Notes in Computer Science, Vol. 5481, Springer: Berlin, pp 195–207.

Poli R, Woodward JR and Burke EK (2007). A histogram-matching approach to the evolution of bin-packing strategies. In: *IEEE Congress on Evolutionary Computation (CEC'07)*. IEEE: Singapore, pp 3500–3507.

Qu R and Burke EK (2009). Hybridizations within a graph based hyper-heuristic framework for university timetabling problems. *Journal of the Operational Research Society* **60**(9): 1273–1285.

Rattadilok P, Gaw A and Kwan RSK (2005). Distributed choice function hyper-heuristics for timetabling and scheduling. In: Burke EK and Trick M (eds). *The Practice and Theory of Automated Timetabling V: Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science Series*. Vol. 3616, Springer: Berlin, pp 51–70.

Rechenberg I (1973). *Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution*. Fromman-Holzboog: Stuttgart.

Remde S, Cowling P, Dahal K and Colledge N (2007). Exact/heuristic hybrids using rvns and hyperheuristics for workforce scheduling. In: *Evolutionary Computation in Combinatorial Optimization* (EvoCOP 2007), Vol. 4446, Springer-Verlag, pp 188–197.

Remde S, Cowling P, Dahal K and Colledge N (2009). Binary exponential back-off for tabu tenure in hyperheuristics. In: Cotta C and Cowling P (eds). *Evolutionary Computation in Combinatorial Optimization*. Vol. 5482, Springer: Berlin, pp 109–112.

Remde S, Cowling P, Dahal K, Colledge N and Selensky E (2012). An empirical study of hyperheuristics for managing very large sets of low level heuristics. *Journal of the Operational Research Society* **63**(3): 392–345.

Ren Z, Jiang H, Xuan J and Luo Z (2010). Ant based hyper heuristics with space reduction: a case study of the p-median problem. In: *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part I*, Springer-Verlag: Berlin, PPSN'10, pp 546–555.

Rice J (1976). The algorithm selection problem. *Advances in Computing* **15**(1): 65–118.

Ridge E and Kudenko D (2007). Analyzing heuristic performance with response surface models: Prediction, optimization and robustness. In: *Genetic and Evolutionary Computation Conference*, GECCO 2007, Vol. 1, ACM Press: New York, pp 150–157.

Ross P (2005). Hyper-heuristics. In: Burke EK and Kendall G (eds). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. chap 17, Springer: Berlin, pp 529–556.

Ross P and Marín JG (2005). Constructive hyper-heuristics in class timetabling. In: *IEEE Congress on Evolutionary Computation*, IEEE: New York, pp 1493–1500.

Ross P, Hart E and Corne D (1997). Some observations about GA-based exam timetabling. In: *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 1997*, Lecture Notes in Computer Science, Vol. 1408, Springer: Berlin, pp 115–129.

Ross P, Schulenburg S, Marín JG and Hart E (2002). Hyper-heuristics: Learning to combine simple heuristics in bin-packing problem. In: *Genetic and Evolutionary Computation Conference*, GECCO 2002, Morgan-Kauffman: Los Altos, CA, pp 942–948.

Ross P, Marín JG, Schulenburg S and Hart E (2003). Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In: *Genetic and Evolutionary Computation Conference*, GECCO 2003, Springer: Berlin, pp 1295–1306.

Ross P, Marín JG and Hart E (2004). Hyper-heuristics applied to class and exam timetabling problems. In: *IEEE Congress on Evolutionary Computation*, IEEE Press: New York, pp 1691–1698.

Runka A (2009). Evolving an edge selection formula for ant colony optimization. In: *Genetic and Evolutionary Computation Conference* (GECCO '09), ACM: New York, pp 1075–1081.

Sabar NR, Ayob M, Qu R and Kendall G (2011). A graph coloring constructive hyperheuristic for examination timetabling problems. *Applied Intelligence* **37**(1): 1–11.

Schmiedle F, Drechsler N, Große D and Drechsler R (2002). Heuristic learning based on genetic programming. *Genetic Programming and Evolvable Machines* **4**(4): 363–388.

Schwefel HP (1977). Numerische Optimierung von Computer-Modellen mittels der Evolutionstrategie, ISR, vol 26. Birkhaeuser.

Shaw P (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *Proc. of International Conference on Principles and Practice of Constraint Programming* (CP'98), Lecture Notes in Computer Science, Vol. 1520, Springer: Berlin, pp 417–431.

Sim K, Hart E and Paechter B (2012). A hyper-heuristic classifier for one dimensional bin packing problems: Improving classification accuracy by attribute evolution. In: *Parallel Problem Solving from Nature* (PPSN XII), Lecture Notes in Computer Science, Vol. 7492, Springer: Berlin, pp 217–233.

Smith J (2002). Co-evolving memetic algorithms: Initial investigations. In: *Parallel Problem Solving from Nature, PPSN VII*, Lecture Notes in Computer Science, Vol. 2439, Springer: Berlin, pp 537–546.

Smith J (2008). Adaptive and multilevel metaheuristics. chap *Self-Adaptation in Evolutionary Algorithms for Combinatorial Optimisation*, Springer: Berlin, pp 31–57.

Smith-Miles K (2008a). Towards insightful algorithm selection for optimisation using meta-learning concepts. In: Simpson PK (ed). *IEEE World Congress on Computational Intelligence, WCCI 2008*. IEEE: Hong Kong, China, pp 4118–4124.

Smith-Miles KA (2008b). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys* **41**(6): 1–25.

Stephenson M, OReilly U, Martin M and Amarasinghe S (2003). Meta optimization: Improving compiler heuristics with machine learning. In: Michael L (ed). *Conference on Programming Language Design and Implementation (SIGPLAN03)*. ACM Press: New York, NY, pp 77–90.

Storer RH, Wu SD and Vaccari R (1992). New search spaces for sequencing problems with application to job shop scheduling. *Management Science* **38**(10): 1495–1509.

Storer RH, Wu SD and Vaccari R (1995). Problem and heuristic space search strategies for job shop scheduling. *ORSA Journal of Computing* **7**(4): 453–467.

Sutton RS and Barto AG (1998). *Reinforcement Learning: An Introduction*. MIT Press: Boston, MA.

Swan J, Özcan E and Kendall G (2011). Hyperion—A recursive hyper-heuristic framework. In: Coello CAC (ed). *LION*. Lecture Notes in Computer Science, Vol. 6683, Springer: Berlin, pp 616–630.

Tavares J, Machado P, Cardoso A, Pereira FB and Costa E (2004). On the evolution of evolutionary algorithms. In: *European Conference on Genetic Programming (EUROGP' 04)*, Coimbra, Portugal, Lecture Notes in Computer Science, Vol. 3003, Springer: Berlin, pp 389–398.

Tay JC and Ho NB (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering* **54**(3): 453–473.

Terashima-Marín H, Ross P and Valenzuela-Rendón M (1999). Evolution of constraint satisfaction strategies in examination timetabling. In: Banzhaf W, *et al* (eds). *Genetic and Evolutionary Computation Conference, GECCO'99*. Morgan Kaufmann: Orlando, Florida, pp 635–642.

Terashima-Marín H, Zarate CJF, Ross P and Valenzuela-Rendón M (2006). A ga-based method to produce generalized hyper-heuristics for the 2d-regular cutting stock problem. In: *Genetic and Evolutionary Computation*, GECCO 2006, ACM Press: New York, pp 591–598.

Terashima-Marín H, Zarate CJF, Ross P and Valenzuela-Rendón M (2007). Comparing two models to generate hyper-heuristics for the 2d-regular bin-packing problem. In: *Genetic and Evolutionary Computation Conference* (GECCO 2007), ACM: New York, pp 2182–2189.

Terashima-Marín H, Ortiz-Bayliss JC, Ross P and Valenzuela-Rendón M (2008). Hyperheuristics for the dynamic variable ordering in constraint satisfaction problems. In: *Genetic and Evolutionary Computation Conference* (GECCO 2008), ACM: New York, pp 571–578.

Terashima-Marín H, Ross P, Farias-Zarate CJ, Lopez-Camacho E and Valenzuela-Rendon M (2009). Generalized hyper-heuristics for solving regular and irregular bin packing problems. *Annals of Operations Research* **179**(1): 369–392.

Thabtah F and Cowling P (2008). Mining the data from a hyperheuristic approach using associative classification. *Expert Systems with Applications: An International Journal* **34**(2): 1093–1101.

Thompson J and Dowsland K (1996). General cooling schedules for a simulated annealing based timetabling system. In: EK Burke PR (ed). *Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science, Vol. 1153, Springer: Berlin, pp 345–363.

Tsai CW, Song HJ and Chiang MC (2012). A hyper-heuristic clustering algorithm. In: *International Conference on Systems, Man, and Cybernetics*, IEEE: New York, pp 2839–2844.

Uludag G, Kiraz B, Etaner Uyar A and Özcan E (2012). Heuristic selection in a multiphase hybrid approach for dynamic environments. In: De Wilde P, Coghill GM and Kononova AV (eds). *Computational Intelligence (UKCI)*. 12th UK Workshop on, IEEE: Edinburgh, UK, pp 1–8.

Vancroonenburg W, Wauters T and Vanden Berghe G (2010). A two phase hyper-heuristic approach for solving the eternity ii puzzle. In: *Proceedings of the International Conference on Metaheuristics and Nature Inspired Computing*, Djerba Island, Tunisia.

Vázquez-Rodríguez J and Petrovic S (2010). A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Journal of Heuristics* **16**(6): 771–793.

Vázquez-Rodríguez J, Petrovic S and Salhi A (2007a). A combined meta-heuristic with hyper-heuristic approach to the scheduling of the hybrid flow shop with sequence dependent setup times and uniform machines. In: Baptiste P, Kendall G, Munier-Kordon A and Sourd F (eds). *Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2007)*. Paris, France, pp 506–513.

Vázquez-Rodríguez J, Petrovic S and Salhi A (2007b). An investigation of hyper-heuristic search spaces. In: Tan KC and Xu JX (eds). *IEEE Congress on Evolutionary Computation (CEC 2007)*. IEEE: Singapore, pp 3776–3783.

Veerapen N, Maturana J and Saubion F (2012). A dynamic island model for adaptive operator selection. In: *Genetic and Evolutionary Computation* (GECCO 2012), ACM: New York, pp 1277–1284.

Vrugt J and Robinson B (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences* **104**(3): 708–711.

Wah BW and Ieumwananonthachai A (1999). Teacher: A genetics-based system for learning and for generalizing heuristics. In: Yao X (ed). *Evolutionary Computation*. World Scientific Publishing Co. Pte. Ltd: Singapore, pp 124–170.

Wah BW, Ieumwananonthachai A, Chu LC and Aizawa A (1995). Genetics-based learning of new heuristics: Rational scheduling of experiments and generalization. *IEEE Transactions on Knowledge and Data Engineering* **7**(5): 763–785.

Walker JD, Ochoa G, Gendreau M and Burke EK (2012). Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: *International Conference on Learning and Intelligent Optimization (LION 6)*, Lecture Notes in Computer Science, Vol. 7219, Springer: Berlin, pp 265–276.

Wilson S (1995). Classifier systems based on accuracy. *Evolutionary Computation* **3**(2): 149–175.

Received March 2011;
accepted May 2013 after two revisions