European Journal of Operational Research 285 (2020) 405–428

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Invited Review

Recent advances in selection hyper-heuristics

John H. Drake[a,*], Ahmed Kheiri[b], Ender Özcan[c], Edmund K. Burke[a]

[a] School of Informatics, University of Leicester, University Road, Leicester LE1 7RH, UK
[b] Department of Management Science, Lancaster University, Lancaster LA1 4YX, UK
[c] School of Computer Science, University of Nottingham, Wollaton Road, Nottingham NG8 1BB, UK

ARTICLE INFO

Article history:
Received 16 February 2018
Accepted 31 July 2019
Available online 7 August 2019

Keywords:
Decision support systems
Artificial intelligence
Machine learning
Metaheuristics
Heuristics

ABSTRACT

Hyper-heuristics have emerged as a way to raise the level of generality of search techniques for computational search problems. This is in contrast to many approaches, which represent customised methods for a single problem domain or a narrow class of problem instances. The term hyper-heuristic was defined in the early 2000s as a *heuristic to choose heuristics*, but the idea of designing high-level heuristic methodologies can be traced back to the early 1960s. The current state-of-the-art in hyper-heuristic research comprises a set of methods that are broadly concerned with intelligently *selecting* or *generating* a suitable heuristic for a given situation. Hyper-heuristics can be considered as search methods that operate on lower-level heuristics or heuristic components, and can be categorised into two main classes: heuristic selection and heuristic generation. Here we will focus on the first of these two categories, selection hyper-heuristics. This paper gives a brief history of this emerging area, reviews contemporary selection hyper-heuristic literature, and discusses recent selection hyper-heuristic frameworks. In addition, the existing classification of selection hyper-heuristics is extended, in order to reflect the nature of the challenges faced in contemporary research. Unlike the survey on hyper-heuristics published in 2013, this paper focuses only on selection hyper-heuristics and presents critical discussion, current research trends and directions for future research.

1. Introduction

There is significant research interest in offering bespoke heuristic solutions to difficult real-world optimisation problems. Such methods rely on problem-specific knowledge to operate, and often produce computationally efficient solutions in reasonable time. However, specific heuristic methods do not always perform well when applied to other problem domains without significant modification. This is a primary motivation for the development of general-purpose problem-independent heuristic search methodologies, known as hyper-heuristics. Hyper-heuristics have received increased attention in the scientific research community over the past decade or so, with significant progress made in developing high-level methods which are applicable to a range of different problems. A key goal of hyper-heuristic research is not only to compete with state-of-the-art problem-specific approaches, but to offer generalised techniques able to deliver good quality solutions for a variety of computational optimisation problems. Another motivation for the development of hyper-heuristics comes from the study of Fisher and Thompson (1963), who concluded that the performance when mixing and combining different low-level heuristics produced better quality solutions than if they were applied separately. Their study showed that individual heuristics may be particularly effective at certain stages of the search process, but perform poorly at others. Therefore, it is reasonable to expect that several heuristics combined in an appropriate way may produce better solutions.

The term “hyper-heuristic” can be defined as a high-level automated search methodology which explores a search space of low-level heuristics (neighbourhood or move operators, or metaheuristics) or heuristic components, to solve computationally hard problems. There are two main types of hyper-heuristics: hyper-heuristics to *generate* heuristics and hyper-heuristics to *select* heuristics. In this study, we focus on the *selection hyper-heuristics* class, which control a set of low-level heuristics during an iterative search process.

An iterative selection hyper-heuristic applies a chosen low-level heuristic to the current solution at each step of a search, before deciding whether to accept or reject the newly created solution.

* Corresponding author.
E-mail addresses: john.drake@leicester.ac.uk (J.H. Drake), a.kheiri@lancaster.ac.uk (A. Kheiri), ender.ozcan@nottingham.ac.uk (E. Özcan), edmund.burke@leicester.ac.uk (E.K. Burke).

https://doi.org/10.1016/j.ejor.2019.07.073

If the search stagnates, i.e. a locally optimal solution is found, a good selection hyper-heuristic will select an appropriate low-level heuristic to diversify the search to another area of the solution space. Note that traditional hyper-heuristics based on the framework initially proposed by Cowling, Kendall, and Soubeiga (2001) only require limited information to operate, such as the number of low-level heuristics, the direction of the optimisation process (maximising or minimising) and the objective function value of a given solution. This modular design, and the utilisation of the *domain barrier* concept, which prevents a hyper-heuristic from retrieving any problem domain specific information, enables them to offer a more general approach to computational search. The idea is that a selection hyper-heuristic or its components can be reused on other problems without needing major modifications. Low-level heuristics often implement simple neighbourhood moves such as swap or shift, or basic local search operations. However, more complex heuristics such as metaheuristics can also be considered at the lower level.

Over the past few years, a number of review papers and articles on hyper-heuristics have been published (Burke et al., 2003; 2010; 2009). Burke et al. (2013) provided an overview of the scientific literature on hyper-heuristics up until the end of 2012, also discussing the history of hyper-heuristics and the intellectual roots of hyper-heuristic methods. The authors introduced some related areas and discussed directions for future research, encouraging more interaction between related communities, especially those working in the fields of metaheuristics and machine learning. A tutorial article, by Ross (2014), gave useful guidelines for implementing hyper-heuristics in addition to discussing a number of relevant research issues and identifying promising application domains. The article also presented a brief history of the area and discussed selected examples in detail. A more recent publication by Branke, Nguyen, Pickardt, and Zhang (2016) provided a comprehensive review of recent developments in generation hyper-heuristics, with an emphasis on the design of construction heuristics in production scheduling optimisation problems. This paper presented three useful components in the design of hyper-heuristics for the generation of heuristics: (i) the representation of what they call *candidate heuristics*, which define the search space, (ii) the optimisation algorithm used to explore this search space, and (iii) the fitness function used to determine the quality of candidate heuristics. The authors classified hyper-heuristics according to the learning method they adopt (supervised or unsupervised). Another recent publication by Pillay (2016) presented an overview of hyper-heuristics for university examination timetabling, university course timetabling and school timetabling problems. The author emphasised one of the key objectives; namely, to produce reusable technologies to solve difficult real-world educational timetabling problems in a more general manner.

General purpose heuristic search and optimisation methods have been studied in various fields, from Operational Research to Computer Science and Artificial Intelligence. Although this study focuses on selection hyper-heuristic approaches in particular, there are other strands of independent ongoing research using related approaches. A survey on 'Algorithm Selection' was presented by Kotthoff (2014). The key goal of the algorithm selection problem is to select the most suitable algorithm to solve a given problem instance, instead of developing new algorithms. This paper presented an overview of previous categorisations of algorithm selection approaches, providing a unified classification and definition for current work. The author also described the concept of 'Algorithm Portfolios', where the decision of which algorithm to use is decided on a case-by-case basis for each problem instance individually. The author distinguished two main classes of portfolios: *Static Portfolios* which are constructed offline before any problem instances are solved; and *Dynamic Portfolios* which change the composition and/or configuration of the constituent algorithms in an online manner, while solving a given instance problem. Pappa et al. (2014) provided a historical perspective on automated algorithm design, discussing similarities and differences between meta-learning for supervised machine learning and general-purpose hyper-heuristics. This discussion focused on the dimensions of the space of possible problem instances, the search space of algorithms (or heuristics) that a high-level search method is operating over, and the performance measure used to evaluate the performance of a given algorithm to a given problem. There are other well established fields of research where studies related to hyper-heuristics have been carried out, such as adaptive Memetic Algorithms (Ong, Lim, Zhu, & Wong, 2006), Adaptive Operator Selection (Fialho, Da Costa, Schoenauer, & Sebag, 2008; Li, Fialho, Kwong, & Zhang, 2014), Variable Neighbourhood Search (Hansen, Mladenović, & Pérez, 2010), Reactive Search (Battiti & Brunato, 2017), algorithm configuration (López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari, & Stützle, 2016; López-Ibáñez & Stützle, 2014) and hybrid metaheuristics (Raidl, 2015).

In this paper, we will provide a review of the selection hyper-heuristic literature, capturing the recent advances in this rapidly growing area of research, extending the existing categorisation of selection hyper-heuristics and identifying issues to be addressed for future research. The relevant studies covered in this paper include only the full papers that appeared after the survey of Burke et al. (2013).

The remainder of this paper is structured as follows, Section 2 discusses some extensions to the existing classification of selection hyper-heuristics, building on the previous classification provided by Burke et al. (2010). Section 3 describes frameworks introduced to facilitate selection hyper-heuristic research, providing a detailed overview the popular HyFlex framework and the CHeSC competitions it was designed to support. Section 4 outlines the details of the selection hyper-heuristics submitted to the CHeSC 2011 competition, and other methods that have been used in the context of cross-domain optimisation subsequently. In Section 5, a survey of selection hyper-heuristics applied to other problem domains, not included in the HyFlex framework, is provided. Section 6 focuses on selection hyper-heuristics for multi-objective optimisation, distinguishing between methods which select from different low-level operators and those which select from different metaheuristics. An overview of recent selection hyper-heuristics that construct solutions, rather than perturb complete solutions, is given in Section 7. Section 8 considers generative hyper-heuristic methods which either automatically generate or configure selection hyper-heuristics. Finally, Section 9 discusses some of the limitations of contemporary hyper-heuristic research and proposes a number of avenues for future research.

2. Extending the classification of selection hyper-heuristics

This section provides an extended classification of selection hyper-heuristics, based on the original classification of Burke et al. (2010). The proposed extended classification is illustrated in Fig. 1, with each component discussed in detail in the subsequent subsections.

2.1. Nature of feedback received

Selection hyper-heuristics iteratively modify the solutions(s) at hand, controlling a set of perturbative low-level (meta)heuristics until the given termination criterion is satisfied. The low-level heuristics in the context of selection hyper-heuristic methods can be simple operators, metaheuristics or even potentially hyper-heuristics. They were originally classified based on the nature of the feedback received during the search process. Hence, selection

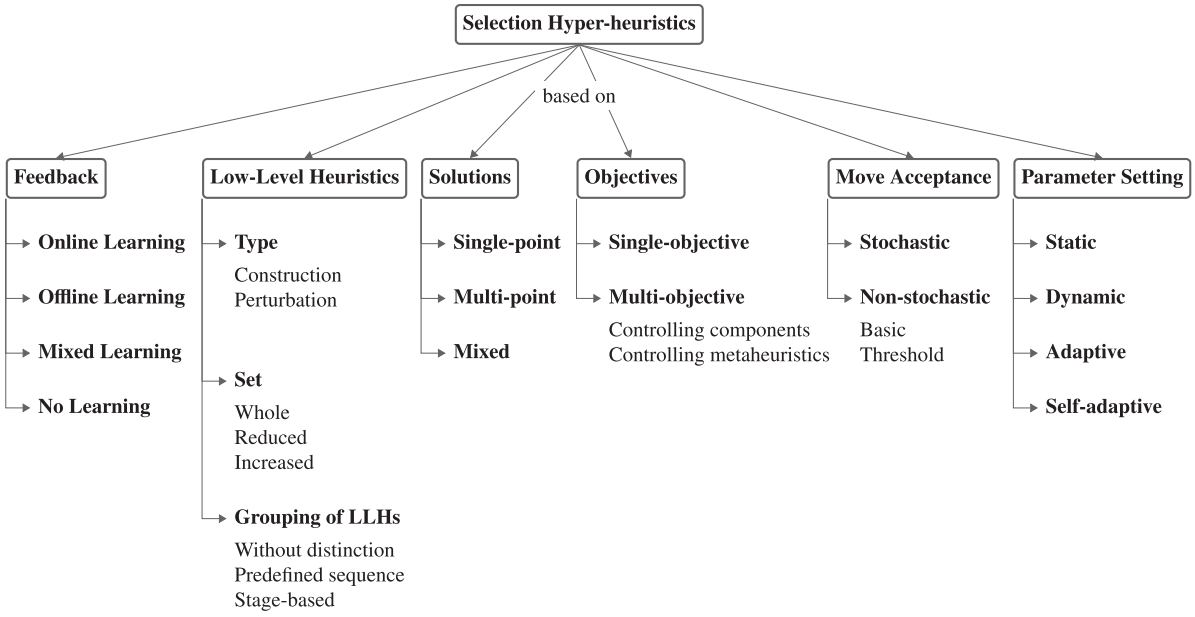

Fig. 1. An extended classification of selection hyper-heuristics.

hyper-heuristics embed either *no learning* mechanism or *online learning* methods to process feedback during the search process, influencing the subsequent decisions made at the hyper-heuristic level (Burke et al., 2010). There are also *offline learning* hyper-heuristics, which are often used as generation hyper-heuristics. Methods of this type are trained on sample problem instances, receiving feedback prior to the search to create new heuristics applicable to unseen problem instances. There are examples of recently proposed selection hyper-heuristics incorporating *mixed learning* methods, combining both offline and online learning (e.g., Asta and Özcan (2015); Uludağ, Kiraz, Etaner-Uyar, and Özcan (2013)).

2.2. Low-level heuristics

2.2.1. Nature of the search structure

Within selection hyper-heuristics Burke et al. (2010) delineated two types of low-level heuristics based on the search structure employed: *construction* and *perturbation* heuristics. Constructive heuristics gradually build a solution from scratch, selecting between a set of pre-defined low-level heuristics to apply at each step, incrementally building a complete solution. Perturbation heuristics operate on complete solutions, performing local search operations using pre-defined neighbourhood structures. Typically this is an iterative process, continuing until some termination criterion is met.

2.2.2. Nature of the low-level heuristic set

Selection hyper-heuristics control a fixed set of low-level heuristics, each of a particular type, such as, mutational, ruin-recreate, local search (hill climbing), crossover or metaheuristics. By design, a selection hyper-heuristic can be allowed to manage the *whole set* of predefined (e.g. unary, binary, *n*-ary) low-level heuristics, a *reduced set* of heuristics excluding a (some) particular type(s) of heuristics (e.g. crossover operators), or an *increased set* of heuristics produced based on the whole set (e.g. via relay hybridisation, creating new heuristics by pairing up existing heuristics). In many previous studies, if the low-level heuristic set consists of metaheuristics, then the whole set is utilised. Recent approaches using an increased set of low-level heuristics often control the size of the heuristic set, attempting to adaptively identify the best performing low-level heuristics with an online learning mechanism or exclude poor performing heuristics using tabu based methods.

2.2.3. Nature of how heuristics are grouped, chosen and applied

The nature of how a heuristic is selected and applied changes from one hyper-heuristic to another. A standard selection hyper-heuristic does not group the low-level heuristics, selecting and applying a single heuristic one at a time *without distinction*. There are other hyper-heuristic methods which group low-level heuristics together and use them separately based on their grouping. Each group of low-level heuristics can be fixed and a *predefined sequence* of heuristic groups can be employed. Özcan, Bilgin, and Korkmaz (2008) identified four different selection hyper-heuristic frameworks utilising a given set of mutational and hill climbing heuristics. It is possible to use the same or multiple selection hyper-heuristics to control heuristic types during the search. There are other hyper-heuristic methods which operate in a *stage-based* manner, deciding on the subset (group) of low-level heuristics to use at each stage, either prior to or during the search. The overall approach can use a fixed number of stages until termination, or perform the search adaptively, switching between stages in an iterated manner.

2.3. Nature of solutions and objective functions

The standard classification of metaheuristics (Birattari, Paquete, Stützle, & Varrentrapp, 2001; Blum & Roli, 2003) also applies to selection hyper-heuristics. *Population* (*multi-point*) based hyper-heuristics use multiple current solutions as they perform a search, while *single-point* based hyper-heuristics use one active current solution. The majority of selection hyper-heuristics are single-point search methods, although there are some population based methods. However, there are also a few studies using a *mixed* approach, combining both single and multi-point based search in phased manner (Hsiao, Chiang, & Fu, 2012; Lehrbaum & Musliu, 2012). A selection hyper-heuristic often consists of a heuristic selection method and a move acceptance mechanism (Özcan et al., 2008), and can be designed to solve *single-objective* or *multiobjective* problems. Multiobjective selection hyper-heuristics tend to focus on one of two approaches, either controlling *components* of a single

multiobjective optimisation algorithm such as the mutation operators, or controlling *multiple multiobjective metaheuristics* within a single search process. It is worth noting that the move acceptance component often becomes a *replacement* strategy, if a population based approach is used.

2.4. Nature of move acceptance

Assuming a single point based search framework, the nature of move acceptance depends on the nature of accept/reject decisions, as well as the parameter setting method used by the move acceptance method (Jackson, Özcan, & John, 2018). The move acceptance mechanism in selection hyper-heuristics can be classified as *stochastic* if a probabilistic framework is considered while making the accept/reject decision (e.g. Simulated Annealing), or *non-stochastic*, otherwise. Non-stochastic move acceptance methods can be further classified into *basic* methods, such as accepting All Moves (AM), accepting Improving or Equal moves (IE), accepting Only Improving moves (OI), and *threshold* acceptance methods (e.g. Great Deluge, Late Acceptance Strategy etc.).

2.5. Nature of parameter setting

Heuristic selection, move acceptance or low-level heuristics often carry parameters that need to be determined or controlled. The algorithmic parameters for the heuristic selection and move acceptance methods need to be handled at the hyper-heuristic level, while the setting of low-level heuristic parameters can be handled by either the hyper-heuristic or low-level heuristic itself. Eiben, Hinterding, and Michalewicz (1999) provided three categories of parameter control for evolutionary algorithms: deterministic, adaptive and self-adaptive. This categorisation based on the nature of parameter setting can be extended to other metaheuristics and hyper-heuristics which embed a move acceptance method.

The parameters can be set *statically* to a fixed value prior to the search process, *dynamically*, allowing the value to change in a predefined manner, or *adaptively* allowing the value to change in a reactive manner during the search process. An algorithm can adjust its behaviour, and hence parameter setting, *self-adaptively* by searching for the best solution and parameter setting simultaneously.

3. Selection hyper-heuristic frameworks

In this section we introduce recent selection hyper-heuristic frameworks, developed to support researchers designing hyper-heuristic methods and to facilitate performance comparison between different hyper-heuristic approaches. The core of this section describes the HyFlex framework, which has become the standard benchmark for comparing cross-domain search methods. A large number of approaches in the literature make use of this framework, and are discussed in detail in Section 4.

3.1. HyFlex v1.0 and CHeSC 2011

HyFlex (**Hy**per-heuristics **Flex**ible framework) is a software tool written in Java, developed for designing and comparing the performance of selection hyper-heuristics (Ochoa et al., 2012). A significant feature of HyFlex is that it implements all of the problem-specific components of optimisation problems, including solution representation, initialisation routines, evaluation functions and low-level heuristics. This allows researchers to focus solely on implementing the high-level strategy to manage the low-level heuristics available. Ross (2014) argued that the enforcement of an explicit separation between the hyper-heuristic and domain-specific aspects, through strict enforcement of the domain barrier, makes HyFlex undesirable for use in large real-world applications, which require far more domain-specific information than it can offer.

In each HyFlex problem domain, a number of benchmark problem instances is supplied, and four different types of low-level heuristics (move operators) are defined. The internal workings of each low-level heuristic are not available to the user. These heuristics can be mutational, ruin-recreate, local search (hill climbing) or crossover. Mutational heuristics modify a solution by randomly perturbing it. Ruin-recreate heuristics make large-scale changes by destroying some parts of a candidate solution, before rebuilding those parts to form a feasible solution, with no guarantee of solution quality. Local search heuristics incorporate an iterative improvement process, guaranteeing that a non-worsening solution will be returned. Crossover heuristics generate a new solution by recombining two existing solutions. Each low-level heuristic in HyFlex is associated with a parameter which can modify its behaviour to a limited extent. Two parameters are used, α and β ($0 <= [\alpha, \beta] <= 1$), representing the *intensity of mutation* and *depth of search* respectively, to control the behaviour of certain low-level heuristics. How these parameters affect the search depends on the low-level heuristic in question. For example, the *depth of search* parameter could specify an iteration or time limit for a particular local search heuristic, while the *intensity of mutation* could indicate how many elements are changed when invoking a mutational heuristic or the percentage of the solution that is destroyed and rebuilt by a ruin-recreate heuristic.

The vast majority of the problem instances within HyFlex are taken from well-known benchmark suites. HyFlex initially provided four optimisation problem domains:

- Boolean satisfiability problem (SAT). This problem requires determining whether an assignment of the variables of a boolean formula exists such that the formula evaluates to true. Given an objective function that calculates the number of clauses that are satisfied, the goal is to minimise the number of unsatisfied clauses. The problem instances for this domain were taken from SATLIB and international SAT competitions (SAT 2007/2009 and Max-SAT 2010) and contain between 200 and 800 variables, and 1000 and 3500 clauses (Hyde, Ochoa, Vázquez-Rodríguez, & Curtois, 2010b).
- One-dimensional bin-packing problem (BP). This problem consists of a set of items, each with a given weight, which must be packed into as few limited capacity bins as possible. The BP instances in HyFlex were randomly generated using distributions taken from well-known sources in the literature (Hyde, Ochoa, Curtois, & Vázquez-Rodríguez, 2010a).
- Personnel scheduling problem (PS). This problem involves determining at which times and on which days a set of employees should work over a specific planning period, with the goal of minimising a weighted sum of several objectives. The majority of problem instances are taken from real-world employee shift scheduling problems (Curtois, Ochoa, Hyde, & Vázquez-Rodríguez, 2010).
- Permutation flow shop problem (PFS). In this problem, there are n jobs to be completed on m consecutive machines, visiting machine 1 then machine 2 and so on. Jobs can be processed by only one machine at a time and machines can only process one job at a time. The goal is to find a permutation of the n jobs that minimises the total time to complete all jobs (i.e. minimising the makespan). All problem instances are from the Taillard set of permutation flow shop benchmark problems (Vázquez-Rodrıguez, Ochoa, Curtois, & Hyde, 2010).

A set of 10 instances for each of these four problem domains were the training benchmark that supported an international competition in 2010/2011, known as the Cross-Domain Heuristic Search Challenge (CHeSC 2011): http://www.asap.cs.nott.ac.uk/chesc2011/.

The competition attracted significant international attention with 20 teams participating. To evaluate each of the competing hyper-heuristics the organisers of the challenge conducted 31 independent runs on three of the ten instances from each of these four problem domains, plus two hidden instances from each problem domain. In addition, another five instances from each of two additional unseen problem domains were also tested, again performing 31 independent runs for each instance. The two additional 'hidden' problem domains were:

- Travelling salesman problem (TSP). Given a list of n cities and the pairwise distances between them, the task is to find the shortest possible tour that visits each city exactly once and returns to the starting city. All instances were taken from the well-known TSPLIB.
- Capacitated vehicle routing problem with time windows (VRP). This problem involves meeting the service demand of a set of customers, using as few vehicles as possible, whilst satisfying a set of constraints, such as adhering to time windows within which a customer must be visited. The VRP instances are from the widely used existing benchmarks provided by Solomon and Gehring-Homberger.

A time limit was imposed for each run to 600 seconds on a typical standard desktop machine. A benchmarking tool was developed by the organisers to report the time another machine should take, equivalent to 600 seconds on the standard machine. The competing hyper-heuristics were ranked using a methodology inspired by the Formula 1 scoring system. The median objective values found during the 31 independent runs of each method were calculated for each problem instance. The eight methods with the best median score for each instance were awarded a score of 10, 8, 6, 5, 4, 3, 2 and 1 points respectively, with the remaining methods awarded 0 points for that instance. These points were totalled across the 30 instances (6 problems, 5 instances) for each algorithm, yielding a potential maximum score of 300 points. The approach that achieved the highest total score was deemed the winner of the challenge. The methods submitted to CHeSC 2011 and the competition results are discussed in detail in Section 4.

An analysis of the set of instances used in the HyFlex benchmark set was performed by Mısır (2017). This study focused on assessing the quality of the benchmark set, in terms of its ability to measure and compare the results of different algorithms. Using matrix factorisation, a number of features characterising different types of problem instances were extracted to form a number of problem instance 'clusters'. Rather than each cluster containing instances from a single problem domain, a level of diversity between clusters was observed, with some clusters consisting of instances from a variety of problem domains. Despite this, one large cluster was identified, containing all of the PFS and TSP instances and some of the BP and PS instances. Due to the size of this cluster, any hyper-heuristics that were able to perform well on this type of instance would have an advantage when being compared using the competition criteria.

Adriaensen, Ochoa, and Nowé (2015) implemented three more problem domains, each with ten problem instances, to extend the HyFlex benchmark set. The extended benchmark set can be downloaded from: https://github.com/Steven-Adriaensen/hyflext and contains the following problem domains:

- 0-1 knapsack problem (KP). Given a set of items, each with associated weight and profit, the goal is to select a subset of items that maximises the total profit gained whilst satisfying capacity constraints, that is, the maximum total weight that can be accommodated by the knapsack. These instances were created using the generator of Martello, Pisinger, and Toth (1999).
- Quadratic assignment problem (QAP). This problem consists of a set of facilities and locations, with a defined distance between each pair of locations, and flow between each pair of facilities. The objective is to find an assignment of facilities to distinct locations that minimises the sum of the distances between each location multiplied by the corresponding flows between each facility. All instances were taken from the well-known QAPLIB and contain between 100 and 256 locations.
- Maximum-cut problem (MAC). Let $G = (V, E)$ be a graph with n vertices and m edges, and each edge has an associated weight. This problem requires determining a cut that maximises the total weight of the edges that have an end point in each set. The ten instances provided in this problem domain are from either the 7th DIMACS Implementation Challenge[1] or generated using Rudy by Giovanni Rinaldi[2].

Table 1
The number of heuristics provided in HyFlex for each supported heuristic type.

	SAT	BP	PS	PFS	TSP	VRP	KP	QAP	MAC
Mutational	6	3	1	5	5	3	5	2	2
Ruin-recreate	1	2	3	2	1	2	2	3	3
Local search	2	2	5	4	3	3	6	2	2
Crossover	2	1	3	4	4	2	3	2	2
Total	11	8	12	15	13	10	17	10	11

A summary of the number of each different type of low-level heuristic for each of the nine problem domains is given in Table 1. Although the four categories of low-level heuristic are simple, the low-level heuristics themselves vary significantly in terms of complexity. The details of the low-level heuristics for the original problem domains can be found in the corresponding technical reports (Curtois et al., 2010; Hyde et al., 2010a; Hyde et al., 2010b; Vázquez-Rodrıguez et al., 2010), while the details for the extended benchmark set are provided by Adriaensen et al. (2015).

3.2. HyFlex v1.1 and CHeSC 2014

Following the study of Asta, Özcan, and Parkes (2013a), an extended version of HyFlex, namely HyFlex v1.1, was developed to accommodate the concept of batch mode hyper-heuristics. This feature allows hyper-heuristics to deal with the HyFlex problem instances collectively as a batch, rather than individually. The idea was motivated by the observation that some of the problem instances are easier than others, and good hyper-heuristics may allocate more time to difficult instances (effort balancing). The batched mode concept also allows hyper-heuristics to learn from earlier instances if they belong to the same problem domain (inter-instance learning). The newer version of HyFlex also gives the hyper-heuristic access to some instance-specific information as a problem instance feature, such as, the size of the instance being solved. Another significant feature of HyFlex v1.1 is that it supports multi-core mode of operation, and allows solution exchange through the use of external memory. HyFlex v1.1 was used in the second Cross-domain Heuristic Search Challenge (CHeSC 2014) which was organised in two tracks. In the first track, the problem instances had to be solved sequentially. However, in the second track, hyper-heuristics were allowed to work with multiple problem instances simultaneously. HyFlex v1.1 supports the same problem domains as the original HyFlex (v1.0). The challenge results and brief extended abstracts describing the competing methods can be found on the challenge website: http://www.hyflex.org/chesc2014/.

[1] http://dimacs.rutgers.edu/Challenges/Seventh/Instances/
[2] https://web.stanford.edu/~yyye/yyye/Gset/

3.3. Other Selection Hyper-heuristic Frameworks

There are many software frameworks implemented in a range of programming languages for rapid development of metaheuristics with reusable components. Parejo, Ruiz-Cortés, Lozano, and Fernandez (2012) reviews some commonly used libraries, mostly for evolutionary computation. We provide an overview of hyper-heuristic frameworks other than HyFlex in this section.

3.3.1. Hyperion

Hyperion (Swan, Özcan, & Kendall, 2011) provides a general recursive object-oriented framework for the development of meta/hyper-heuristics, incorporating the selection hyper-heuristic frameworks described by Özcan et al. (2008). Its main goal is to decompose the domain into collections of policy components, yielding a generative algorithm framework that facilitates the identification of the components that contribute to an algorithm's success in a procedural way. The Hyperion framework has been extended to Hyperion[2] (Brownlee, Swan, Özcan, & Parkes, 2014), allowing the analysis of the *trace* taken through the search space by algorithms and their constituent components, and promoting interoperability through component interfaces.

3.3.2. ParHyFlex

Inspired by HyFlex, ParHyFlex (Van Onsem & Demoen, 2013) was built to support the development of hyper-heuristics in a parallel environment. The framework is implemented using Java, and the Message Passing Interface (MPI) protocol is used to handle the communication between different processes. One of its interesting features is the way the search trajectory of a process is influenced by experience learned in other processes, which could potentially reduce the chance of becoming trapped in similar regions of the search space.

3.3.3. hMod

Urra, Cabrera-Paniagua, and Cubillos (2013) proposed a concrete object-oriented design pattern referred to as the *flowchart pattern*, which allows one to construct an objectual representation of an algorithm flowchart for dynamic heuristic environments that can be modified or reused at runtime. The hMod framework supports the development of selection hyper-heuristics by offering specialised semantics, through different techniques for facilitating the algorithm building capabilities offered by the flowchart pattern. hMod allows the developer to define the components of selection hyper-heuristics through XML definition files. The two main components of an iterative selection hyper-heuristic, heuristic selection and move acceptance, are defined in separate XML files.

3.3.4. HH-DSL

Cora, Uyar, and Etaner-Uyar (2013) introduced a domain specific language (DSL) to facilitate rapid implementation of selection hyper-heuristics by non-experts in HyFlex. The proposed HH-DSL eliminates the need to develop hyper-heuristics in Java directly, providing a high-level language to define hyper-heuristics using HyFlex, allowing researchers to focus solely on hyper-heuristic development rather than Java programming. The source code of HH-DSL is available online at: https://bitbucket.org/hcora/hh-dsl.

3.3.5. EvoHyp

EvoHyp is a Java framework recently introduced by Pillay and Beckedahl (2017), targeting researchers with limited experience in hyper-heuristic development, with a focus on hyper-heuristics based on evolutionary algorithms. This framework provides a toolkit from which evolutionary selection and generation hyper-heuristics, both constructive and perturbative, can be built, including distributed variants. The Java source and documentation for EvoHyp is available online: http://titancs.ukzn.ac.za/EvoHyp.aspx.

3.3.6. Multiobjective hyper-heuristic frameworks

There are a growing number of multiobjective hyper-heuristic studies in the literature which use existing software libraries for multiobjective optimisation, such as, PISA accessible at http://www.tik.ee.ethz.ch/pisa/ (Bleuler, Laumanns, Thiele, & Zitzler, 2003) and jMetal accessible at http://jmetal.sourceforge.net/ (Durillo & Nebro, 2011).

4. Selection hyper-heuristics for cross-domain search

Here we discuss the literature of selection hyper-heuristics that were developed and tested using the HyFlex framework. This section is split into four subsections. The first subsection covers the methods which were participants in the CHeSC 2011 competition. The second discusses methods which were applied to the original six benchmark problem domains following the competition. Methods which were applied to only a subset of the problem domains in HyFlex are reported in the third subsection, and finally, the papers which include the additional three problem domains from the extended benchmark set (Adriaensen et al., 2015) are covered in the last subsection.

4.1. Applied to all six HyFlex problem domains - CHeSC 2011 hyper-heuristics

The results of the twenty participants in the CHeSC 2011 challenge using the Formula 1 scoring system are given in Table 2.

The winning algorithm, AdapHH (a.k.a. GIHH), was proposed by Mısır, Verbeeck, De Causmaecker, and Vanden Berghe (2012b). AdapHH adaptively maintains subsets of low-level heuristics for different phases of the search process. A number of performance metrics are used to determine which heuristics are in the active subset, including the number of new best solutions found by the heuristic, the total solution improvement and deterioration over the search, the total solution improvement and deterioration during the current phase, and the remaining execution time. These measures, each with a given weight, are used to calculate a quality index for each low-level heuristic. A heuristic with a quality index value lower than that of the average of the full set of heuristics is excluded from the active subset for the corresponding phase. The length of time that the heuristic is excluded from the subset is referred to as the tabu duration. However, if a particular heuristic is consecutively excluded for a given number of phases, it is permanently excluded from the low-level heuristic set. The phase length is set to a predetermined constant value. AdapHH selects a heuristic from within the active subset based on a set of associated probability values for each heuristic. These probabilities are dynamically modified during the search, based on the number of best improvements found with respect to execution time taken. During part of the search process, AdapHH employs a relay hybridisation technique to discover pairs of low-level heuristics that are effective when applied consecutively. The user controlled parameter values of each low-level heuristic, intensity of mutation and depth of search, are adaptively maintained by AdapHH using a Reinforcement Learning method. The move acceptance criterion accepts solutions below a certain threshold, defined by the fitness values of previous best solutions. This threshold is dynamically adjusted after a certain number of iterations of non-improvement. The acceptance strategy is referred to as Adaptive Iteration Limited List-based Threshold Accepting. Finally, to prevent the search from stagnating, the solu-

Table 2
The results of the CHeSC 2011 competing approaches.

Rank	Method label (reference if available)	Total Score	SAT	BP	PS	PFS	TSP	VRP
1	AdapHH (Mısır et al., 2012b)	181.00	34.75	45.00	9.00	37.00	40.25	15.00
2	VNS-TW (Hsiao et al., 2012)	134.00	34.25	3.00	39.50	34.00	17.25	6.00
3	ML	131.50	14.50	12.00	31.00	39.00	13.00	22.00
4	PHUNTER (Chan et al., 2012)	93.25	10.50	3.00	11.50	9.00	26.25	33.00
5	EPH	89.75	0.00	10.00	10.50	39.00	36.25	12.00
6	HAHA (Lehrbaum & Musliu, 2012)	75.75	32.75	0.00	25.50	3.50	0.00	14.00
7	NAHH (Mascia & Stützle, 2012)	75.00	14.00	19.00	2.00	22.00	12.00	6.00
8	ISEA (Kubalík, 2012)	71.00	6.00	30.00	14.50	3.50	12.00	5.00
9	KSATS-HH	66.50	24.00	11.00	9.50	0.00	0.00	22.00
10	HAEA	53.50	0.50	3.00	2.00	10.00	11.00	27.00
11	ACO-HH	39.00	0.00	20.00	0.00	9.00	8.00	2.00
12	GenHive (Cichowicz et al., 2012)	36.50	0.00	14.00	6.50	7.00	3.00	6.00
13	DynILS	27.00	0.00	13.00	0.00	0.00	13.00	1.00
14	SA-ILS	24.25	0.75	0.00	19.50	0.00	0.00	4.00
15	XCJ	22.50	5.50	12.00	0.00	0.00	0.00	5.00
16	AVEG-Nep (Di Gaspero & Urli, 2012)	21.00	12.00	0.00	0.00	0.00	0.00	9.00
17	GISS	16.75	0.75	0.00	10.00	0.00	0.00	6.00
18	SelfSearch	7.00	0.00	0.00	4.00	0.00	3.00	0.00
19	MCHH-S	4.75	4.75	0.00	0.00	0.00	0.00	0.00
20	Ant-Q	0.00	0.00	0.00	0.00	0.00	0.00	0.00

tion is reinitialised if a certain number of iterations are executed without an improvement in solution quality. Many of the components of AdapHH were introduced in a later paper by the same authors (Mısır, Verbeeck, De Causmaecker, & Vanden Berghe, 2013b). In CHeSC 2011, AdapHH was the best hyper-heuristic in the SAT, BP and TSP problem domains and ranked second in the PFS problem domain. It delivered poor performance in the PS problem domain compared to the other competitors. Despite the fact that the nature of the combination of many adaptive components seems to be key to the performance of AdapHH, Adriaensen & Nowe (2016) have demonstrated that some of these elements do not necessarily contribute to the success of the approach. This paper is discussed in more detail in Section 4.4.

The second place hyper-heuristic was that of Hsiao et al. (2012). Their method is based on Variable Neighborhood Search (VNS), iterating over a predefined sequence of two phases, first with a population of solutions, before moving on to a second phase using only a single solution. The two phases consist of a 'shaking' stage, to promote diversification of the search, and a local search stage for intensification. The shaking stage applies mutational and ruin-recreate low-level heuristics, with a tabu mechanism employed to prevent frequent application of poor performing low-level heuristics. The local search stage is applied until a local optimum is reached, incorporating an adaptive technique to adjust the strength of the hill climbing heuristics over time. The authors argued that the population based search phase could potentially eliminate poor quality solutions by tournament selection. A second phase is initiated when the search stagnates or half of the allowed computational budget is spent. In this phase, the hyper-heuristic reduces the population size to a single solution. This approach was ranked first in the PS problem domain, second in SAT, third in PFS, fourth in TSP and produced relatively poor performance on BP and VRP problem instances when compared to the other nineteen competing methods.

The ML hyper-heuristic finished third in the CHeSC 2011 competition. This method, proposed by Mathieu Larose and described briefly in Ochoa et al. (2012), is based on a self-adaptive metaheuristic using multi-cooperative agents and a Reinforcement Learning technique. The method comprises of three main stages: diversification (using mutational, ruin-recreate and no-op heuristics), intensification (using local search heuristics) and move acceptance. The move acceptance mechanism accepts moves in the case that the current solution is improved, or if the candidate solution has not been improved for a given number of iterations. ML ranked first in PFS, second in PS, third in VRP and produced average performance on the remaining HyFlex problem domains.

The fourth ranking hyper-heuristic, PHunter (Chan, Xue, Ip, & Cheung, 2012). The method can be described as a type of Iterated Local Search, as it involves a process of diversification and intensification. The authors distinguished two forms of intensification by controlling the depth of search parameter in HyFlex. Furthermore, a tabu list to prevent revisiting poor quality solutions is employed. For a given problem domain PHunter determines a mode, consisting of a portfolio of grouped heuristics and a mechanism for diving. In a preliminary run, the algorithm counts the number of suboptimal solutions found by each group of heuristics and different dive mechanisms. An offline learning mechanism is used to decide the final mode. Interestingly, PHunter placed first in the VRP problem domain and third in the TSP, both of which were hidden domains. This indicates that the method is able to adapt well to new, unseen problem domains. Despite the fact that HyFlex was chiefly designed to evaluate the generality level of selection hyper-heuristics, Chan et al. (2012) discovered new best-known solutions for three well-known personnel scheduling problem instances using HyFlex.

The Evolutionary Programming Hyper-heuristic (EPH) hyper-heuristic (Meignan, 2011) finished fifth in the CHeSC 2011 competition. EPH is based on an evolutionary programming methodology and co-evolution, simultaneously maintaining a population of solutions and a population of low-level heuristic sequences to apply to the solutions. A heuristic sequence consists of a set of one or two perturbation heuristics (mutational, ruin-recreate or crossover) followed by a set of all available local search heuristics. Values for the depth of search and intensity of mutation parameters of each low-level heuristic in the sequence are also evolved as part of each sequence. If the perturbation stage consists of two heuristics, then they must be of different types, and if one is a crossover operator it must be invoked first. Local search heuristics are either applied once, or using a Variable Neighbourhood Descent strategy. The population of low-level heuristic sequences is initialised randomly. At each generation, mutation is applied to each low-level heuristic sequence, doubling the size of the population. Following this, the best individuals are selected using tournament selection. EPH uses four mutation strategies each with an equal probability of being applied: (i) modify the intensity of mutation parameter values for the perturbation heuristics; (ii) add, change or delete a perturbation heuristic at random, whilst limiting the sequence to a maximum of two perturbation heuristics; (iii) modify the depth of

search parameter values for the local search heuristics; (iv) change the order of local search heuristics randomly. The parameters of this hyper-heuristic (such as the population size of the solution and low-level heuristic sequence populations, and the choice of local search strategy) are either fixed or determined during a preliminary phase at the start of each run.

The remaining competing hyper-heuristics employ a variety of interesting concepts. The HAHA algorithm (Lehrbaum & Musliu, 2012), which was sixth, splits the search into a single-point based strategy and a population based strategy and repeatedly switches between the two. A dedicated initial phase is used to assign scores to the local search heuristics based on performance. The algorithm embeds an adaptive move acceptance mechanism, a tabu search technique and a method to reinitialise the search if no improvement is observed for a certain amount of time. The NAHH hyper-heuristic, proposed by Mascia and Stützle (2012), uses a stochastic local search search method and selects one of several algorithm schemata (ranging from well-established metaheuristic techniques such as Iterated Local Search, Variable Neighbourhood Descent, and Simulated Annealing, to well-known heuristic selection methods such as Greedy and Simple Random) that have been tuned to solve each of the HyFlex four public problem domains in an offline manner. A preliminary phase is used to discard dominated heuristics, followed by an iterated racing procedure to decide which of the available schemata methods to apply for a given problem instance. This non-adaptive algorithm finished seventh overall in the challenge, generally performing better in the four original domains it was trained on than the two hidden problem domains, but still outperforming many online methods in the hidden domains. The authors subsequently improved their methodology through careful tuning and a different set of schemata. The improved version would have ranked fourth in the CHeSC 2011 challenge (Mascia & Stützle, 2012). Kubalík (2012) developed the ISEA algorithm, which uses an evolutionary based algorithm to evolve a population of sequences of heuristics through add, delete and change mutation moves. The first and last heuristic in a given sequence of low-level heuristics must be a local search heuristic. The constructed sequence of heuristics is applied to the candidate solution, with the current solution reinitialised if no improvements are observed for a certain period of time. This algorithm finished eighth in the challenge. However, a modified version of the method that employs an adaptive reinitialisation scheme would have taken second place in CHeSC 2011 (Kubalík, 2012). Cichowicz et al. (2012) proposed two hyper-heuristics, a Five Phase hyper-heuristic and a Genetic Hive hyper-heuristic. The former performs five different phases including intensification, stagnation, diversification, mutation and crossover using a number of working solutions. The latter uses a population based approach, inspired by an evolutionary algorithm which imitates the behaviour of bees searching for food. The Genetic Hive method evolves a population of sequences of heuristics, each to apply to a selected solution. The authors performed extensive experiments and compared several variants of the two hyper-heuristics, each with different parameter settings. The results revealed that the Five Phase hyper-heuristic performs better than the Genetic Hive hyper-heuristic on average, however the Genetic Hive method receives a better score when using the Formula 1 scoring system from the competition. The Genetic Hive hyper-heuristic was submitted to CHeSC 2011 and finished twelfth. Di Gaspero and Urli (2012) introduced the AVEG-Nep hyper-heuristic that finished sixteenth out of the twenty competition entries. This approach uses Reinforcement Learning as a heuristic selection method. Several variants of Reinforcement Learning were investigated, with a variant that controls the parameters using a multilayer perceptron neural network shown to be the most promising. This method would have finshed thirteenth in the CHeSC 2011 competition.

The decision of which mechanism to use for performance evaluation when comparing the results obtained by empirical testing can often affect the conclusions that can be drawn from such experiments. Although the CHeSC 2011 competition results were decided using the Formula 1 scoring system described in Section 3.1 above, many alternative mechanisms could be used for performance comparison. The Formula 1 system was designed with the intention of rewarding methods which perform well across a set of problem domains, rather than those that show strong performance in only one or two domains. Di Gaspero and Urli (2012) introduced the idea of comparing the performance of the CHeSC 2011 entrants using normalised cost function values of the median results of each method for each instance. This concept has been used in a variety of subsequent papers to compare performance to the CHeSC 2011 competition, examples of this metric can be seen in the box plot comparisons presented by Drake (2014) for example. Adriaensen et al. (2015) used six different metrics for performance comparison, including the normalised objective function as suggested by Di Gaspero and Urli (2012). When using these different metrics, AdapHH (Mısır et al., 2012b) was still shown to perform best among the CHeSC 2011 competition entrants.

4.2. Applied to all six HyFlex problem domains - non-CHeSC 2011 hyper-heuristics

After the CHeSC 2011 challenge, the hyper-heuristics community recognised the results of the competition as a benchmark to evaluate the quality and generality level of newly developed selection hyper-heuristics. A number of papers have presented methods which claim to outperform AdapHH and all of the other hyper-heuristics submitted to CHeSC 2011. However, not all have provided their full results and source code for independent verification. We contacted the authors of all methods claiming to outperform all of the competition entrants and received responses and links to the following resources: Adriaensen, Brys, and Nowé (2014b)[3], Kheiri and Keedwell (2015)[4] and Kheiri and Özcan (2016)[5].

A relatively simple hyper-heuristic, based on Iterated Local Search (ILS), was presented by Adriaensen et al. (2014b). Their Fair-Share ILS hyper-heuristic applied a mutation or ruin-recreate heuristic, selected proportionally based on previous performance, before performing an improvement phase using local search heuristics. This method was shown to outperform AdapHH based on the Formula 1 scores from the competition, obtaining better results than all twenty CHeSC entrants for SAT, PFS and VRP. Additional analysis showed that each design decision made when developing this method was contributing directly to the overall performance of the algorithm. ILS based hyper-heuristics were also considered by Meignan, Schwarze, and Voß (2016), who incorporated look-ahead mechanisms in order to help guide the search process. The proposed methods were shown to outperform traditional ILS on the CHeSC 2011 benchmarks, with additional comparisons performed using instances from the International Nurse Rostering Competition (INRC2010).

Drake, Özcan, and Burke (2012) tested an improved Choice Function selection hyper-heuristic over the CHeSC 2011 benchmark. The proposed method introduced an adaptive mechanism inspired by Reinforcement Learning to control the parameter values of the classic Choice Function heuristic selection method. This work argued that the traditional Choice Function as described by Cowling et al. (2001) can potentially suffer from excessive diversification when the search is trapped in a local optimum. To overcome

[3] https://github.com/Steven-Adriaensen/FS-ILS
[4] http://ahmedkheiri.netlify.com/publications/SSHH.zip
[5] http://ahmedkheiri.netlify.com/publications/MSHHs.zip

this, the Modified Choice Function method rewards the intensification component and heavily punishes the diversification element each time an improvement in solution quality is made. This relatively simple and easy-to-implement approach statistically significantly outperformed the traditional Choice Function in the SAT, PFS and VRP problem domains. The results of the classic Choice Function and the Modified Choice Function when compared to the CHeSC 2011 competitors show that the former ranks twentieth out of twenty-one approaches, and the latter ranks twelfth overall. The results provide evidence to highlight the importance of parameter tuning.

Asta and Özcan (2015) used tensor analysis to identify the latent relationships between low-level heuristics, combining Simple Random heuristic selection with two simple move acceptance methods: Naive Acceptance with a probability of 0.5, and Improving and Equal. Firstly, the hyper-heuristic is run using Naive Acceptance with all low-level heuristics for a short period on a given instance and the search trajectory is saved as a third-order tensor. Based on the analyses of results from tensor factorisation, low-level heuristics are partitioned into two, where each partition is associated with a move acceptance method. Then, a multi-stage hyper-heuristic is run iteratively for the remaining time. Each move acceptance method is used in turn with the associated low-level heuristics at each stage. The results on the CHeSC 2011 problem domains indicate the success of this simple yet effective method, which ranks second against the hyper-heuristics submitted to the competition.

The HyFlex framework includes several low-level heuristic types, including crossover heuristics. Unlike other low-level heuristics, crossover heuristics require more than one solution as input, so a method to select and manage the input for crossover heuristics needs to be defined. Drake (2014) investigated the use of crossover control schemes within two existing selection hyper-heuristics, analysing the difference in performance when modifying the strategy for managing potential solutions for crossover. Ferreira, Gonçalves, and Pozo (2017) maintained an auxiliary set of solutions explicitly for crossover, made up of previously found best-so-far solutions. The proposed framework used a Multi-armed Bandit selection mechanism with a number of different acceptance criteria, and adaptively changed the depth of search and intensity of mutation parameters using Reinforcement Learning.

Jackson, Özcan, and Drake (2013) introduced a number of fitness proportional heuristic selection methods based on the Formula 1 ranking used in the CHeSC 2011 competition. Rather than using this ranking to compare methods, this selection method ranked the individual low-level heuristics under consideration for selection at each step. Using Late Acceptance Strategy (Burke & Bykov, 2017) as an acceptance criterion, good results were observed in the PS and SAT problem domains. Perhaps counter-intuitively, reversing the scores assigned to a heuristic (i.e. punishing the best performing heuristic and rewarding the worst) was shown to improve performance, promoting diversity within the search in order to prevent stagnation and avoid getting stuck in local optima.

Kheiri and Özcan (2016) presented an iterated multi-stage selection hyper-heuristic framework, enabling the use of several interacting hyper-heuristics at different stages during the optimisation process. The authors argued that an additional upper heuristic level, referred to as the multi-stage level, is required to manage the transition and information exchange between multiple hyper-heuristics. A selection hyper-heuristic consisting of two stages was introduced, referred to as MSHH. The first stage applies a greedy strategy using the given low-level heuristics provided by HyFlex and 'new' heuristics generated via relay hybridisation (pairing up of heuristics) for a number of steps. Then a dominance based approach is used to decide which heuristics perform equally well, determining the low-level heuristic set and associated selection probabilities for the following stage. In the second stage, a low-level heuristic is selected and applied using a roulette wheel strategy based on the selection probability of each heuristic. In both stages, an adaptive threshold move acceptance method is used. The empirical results showed that MSHH performs better than five other multi-stage hyper-heuristics across the HyFlex problem domains.

A train and test approach was used by Yates and Keedwell (2017), with sequences of low-level heuristics and objective function values yielded from applying a simple hyper-heuristic used to train a recurrent neural network. Following training, the neural network is used to generate new sequences of heuristics that are then applied to unseen problem instances. The generated sequences were observed to be capable of producing better results than the sequences used for training. Perhaps unsurprisingly, sequences that are trained and tested on a single problem domain perform better than those trained on multiple problem domains. Although this method was developed and tested using the original four HyFlex problem domains (SAT, BP, PS, PFS), no comparison to any previous methods was provided.

Kheiri and Keedwell (2015) investigated the use of hidden Markov models (HMM) to produce sequences of heuristics. The resulting method, referred to as a sequence-based selection hyper-heuristic (SSHH), replaces the hidden states of the HMM with low-level heuristics and uses a matrix of transition probabilities to determine the movement between these hidden states. Another set of observation probabilities determine whether a given heuristic will be applied alone, or will be coupled with another low-level heuristic to form a sequence of heuristics.

4.2.1. Performance of low-level heuristics and heuristic types in specific problem domains

Here we highlight the comments made by different authors on the performance of individual low-level heuristics and heuristic types for each of the six CHeSC 2011 problem domains.

Boolean satisfiability problem (SAT)

The work of Mısır et al. (2013b) demonstrated that the effectiveness of using a relay hybridisation technique, pairing up heuristics to be applied consecutively, is very limited in this problem domain. Kheiri and Özcan (2016) discovered that most improving moves are a result of applying mutational heuristics. Asta and Özcan (2015) indicated that the ruin and recreate heuristic (LLH6) is useful when deployed as a hill climber (i.e. with Improving and Equal move acceptance). They also confirmed the usefulness of all of the mutational heuristics (LLH0, LLH1, LLH2, LLH3, LLH4, LLH5) in addition to the hill climbers (LLH7, LLH8). Although Adriaensen and Nowé (2016) indicated that including crossover heuristics is beneficial in general, Drake (2014) argued that explicitly removing crossover low-level heuristics from the set of available heuristics has the potential to improve performance in this problem domain.

One-dimensional bin-packing problem (BP)

Mısır et al. (2013b) showed that relay hybridisation is very effective for finding pairs of heuristics that intensify the search process. Both hill climbers (LLH4, LLH6) perform as the most effective second heuristics in an identified pair. Mutational (LLH0, LLH3), ruin-recreate (LLH2) and crossover (LLH7) low-level heuristics help to diversify the search. Their work also observed that LLH3, LLH2, LLH7 and LLH6 were mostly maintained in the available heuristic set over time. On the other hand, mutational heuristics (LLH0, LLH5), and the ruin-recreate heuristic (LLH1) and hill climber (LLH4) were mostly excluded despite their effective performance during certain phases of the search. Some of these findings were confirmed by other studies. Drake (2014) also found that crossover (i.e. LLH7) was able to greatly improve solution quality.

Asta and Özcan (2015) commented on the usefulness of LLH3 and LLH6, and Kheiri and Özcan (2016) noted that LLH5 can actually be detrimental to the performance of the hyper-heuristic.

Personnel scheduling problem (PS)

Mısır et al. (2013b) demonstrated that relay hybridisation identified some effective heuristic pairs, composed of a mutational heuristic (LLH11) and ruin-recreate heuristics (LLH6, LLH7) as the first heuristics, and hill climbers (LLH3, LLH4) as the second heuristics. Due to the slow speed of execution of the low-level heuristics associated with this problem domain, in particular the hill climbers, Mısır et al. (2013b) were not able to reduce the heuristic set by eliminating ineffective heuristics within the available computation time. However, Mascia and Stützle (2012) performed a pre-processing phase to eliminate poor performing heuristics, revealing that the hill climbers LLH2, LLH3 and LLH4 are non-dominated heuristics, and therefore the remaining low-level heuristics could be excluded. Kheiri and Özcan (2016) argued that LLH0 and LLH1, which are provided as hill climbers, do not yield any improvement either individually or in combination with another low-level heuristic. Asta and Özcan (2015) indicated that, in most of their experiments, ruin and recreate heuristics are identified as a source of 'noise' and are excluded from the search space, where a noisy heuristic is one that generates solutions with a significant deterioration in quality with high probability. In the remaining cases, mutational heuristics are excluded as noise. Jackson et al. (2013) observed that reducing the heuristic search space in this problem domain can be detrimental to the overall performance.

Permutation flow shop problem (PFS)

Mısır et al. (2013b) showed that the hill climbers (LLH7, LLH8, LLH9, LLH10) were used effectively as second heuristics in heuristic pairs identified by relay hybridisation, with LLH7 and LLH8 performing particularly well. The ruin-recreate heuristics (LLH5, LLH6) and mutational heuristics (LLH0, LLH1) were used as the first heuristics generally. Drake (2014) commented that the use of crossover low-level heuristics greatly improves the solution quality in this domain. Kheiri and Özcan (2016) note that the use of a combination of a mutational heuristic followed by a hill climbing heuristic (i.e. using the same basic structure of Iterated Local Search), was favoured by their approach on this domain and the TSP.

Travelling salesman problem (TSP)

Mısır et al. (2013b) demonstrated that for the TSP the effect of relay hybridisation is useful during certain points of the search process. Again an Iterated Local Search structure was automatically identified as an effective search strategy by their hyper-heuristic. The mutational heuristics (LLH0, LLH1, LLH3, LLH4) and the only ruin-recreate heuristic (LLH5) were used as the first heuristic in a pair, with hill climbers (LLH7, LLH8) most frequently used as second heuristics. Drake (2014) noted that the performance of crossover heuristics can vary significantly depending on the instance being solved. Kheiri and Keedwell (2015) observed that LLH0, LLH1 and LLH5 do not make any improvement when applied on their own, but the majority of the improvements made in this domain were due to combining these heuristics with LLH8 (hill climber). Moreover, LLH8 does not improve the best-of-run solutions unless combined with these heuristics.

Capacitated vehicle routing problem with time windows (VRP)

Mısır et al. (2013b) highlighted that their method took advantage of relay hybridisation during the early stages of the search in the VRP domain. The hill climbers (LLH4, LLH8) were effective second heuristics. Drake (2014) provided evidence that in this domain that the performance of crossover heuristics also varies significantly depending on the instance being solved, and that the choice of crossover control scheme is crucial in determining performance. Asta and Özcan (2015) employed a pre-processing phase to eliminate poor performing heuristics. Ruin and recreate heuristics were identified as a source of noise when solving VRP instances. Kheiri and Özcan (2016) noted that no generated heuristic pairs contribute towards the improvement of the best solutions found.

4.3. Applied to some HyFlex problems (2-5 domains)

Alanazi and Lehre (2016) provided a theoretical analysis of the performance limits when using Reinforcement Learning as a learning mechanism within selection hyper-heuristics. The authors argue that, given the probability of improving a solution at each step is less than 50%, the performance of using additive Reinforcement Learning is asymptotically similar to simple uniform random heuristic selection, suggesting that additive Reinforcement Learning cannot necessarily capture differences in performance of individual low-level heuristics. The results of their analysis were corroborated by a set of empirical experiments using the BP and PFS domains within HyFlex.

Four of the domains from HyFlex were used as a benchmark by Chuang and Smith (2017) when studying 'chains' of solutions, sampled through random heuristic selection. This method is based on a Simple Random - Only Improving framework. However, a certain number of non-improving moves are permitted. If no improving moves are found after a specified number of steps, i.e. the length of the solution chain, the solution returns to the last accepted solution. Empirical studies using different strategies to manage the length of solution chains sampled were presented using SAT, BP, PFS and TSP. No comparison to the CHeSC 2011 entrants or other work in the literature using the HyFlex framework was given.

4.4. Applied to CHeSC 2011 problems and extended benchmark sets (9 domains)

As mentioned in Section 3.1, Adriaensen et al. (2015) introduced an extension to the original benchmark set, providing implementations in HyFlex for the quadratic assignment problem, the 0-1 knapsack problem and the maximum-cut problem. Since then, a number of papers have used the extended set for cross-domain performance comparison of selection hyper-heuristics.

An 'Accidental Complexity Analysis' of the CHeSC 2011 winning AdapHH hyper-heuristic, was presented by Adriaensen and Nowé (2016). In this work, the authors argued that it is possible to reduce the complexity of AdapHH, by removing a number of mechanisms, without loss of performance. A total of 39 simplifications were proposed, with a number of these providing statistically significant performance improvements compared to the original hyper-heuristic. A 'lean' version of the original method, which combines multiple simplifications (reducing the overall program size from 2324 to 288 lines of code in the process[6]), performed better over all 98 instances from the nine problem domains.

Almutairi, Özcan, Kheiri, and Jackson (2016) compared the performance of various selection hyper-heuristics including AdapHH, SSHH (Kheiri and Keedwell, 2015), two methods from Adriaensen et al. (2015) and three other previously proposed algorithms on the extended HyFlex benchmark. The results showed that AdapHH performs the best across the extended domains based on raw ranking. However, SSHH was the best method based on normalised fitness measure.

[6] Available online at: https://github.com/Steven-Adriaensen/Lean-GIHH

Table 3
A sample of selection hyper-heuristics for single objective optimisation belonging to various categories based on the extended classification.

Source	Search points	Feedback	LLH set	Grouping of LLHs	Accept/reject	Parameter setting in move acceptance
(Chan et al., 2012)	Single	Mixed	Whole	Predefined	Basic, threshold	Static, adaptive
(Di Gaspero & Urli, 2012)	Single	Online	Whole	Predefined	Basic	None
(Drake et al., 2012)	Single	Online	Reduced	Without distinction	Basic	None
(Hsiao et al., 2012)	Mixed	Online	Reduced	Predefined	–	–
(Kubalík, 2012)	Population	Mixed	Reduced	Predefined	–	–
(Lehrbaum & Musliu, 2012)	Mixed	Online	Reduced	Predefined	–	–
(Mascia & Stützle, 2012)	Single	Offline	Reduced	Predefined	Stochastic	Static
(Mısır et al., 2012b)	Single	Online	Whole	Without distinction	Threshold	Adaptive
(Jackson et al., 2013)	Single	Online	Whole	Without distinction	Threshold	Static
(Adriaensen et al., 2014b)	Single	Online	Whole	Predefined	Stochastic	Adaptive
(Kheiri et al., 2016)	Single	Online	Reduced	Predefined	Threshold	Adaptive
(Asta & Özcan, 2015)	Single	Offline	Reduced	Stage-based	Basic	Static
(Drake, 2014)	Single	Online	Whole	Without distinction	Basic	None
(Kheiri & Keedwell, 2015)	Single	Online	Reduced	Without distinction	Threshold	Adaptive
(Asta et al., 2016a)	Single	Online	Reduced	Stage-based	Threshold	Adaptive
(Kheiri & Özcan, 2016)	Single	Online	Increased	Stage-based	Threshold	Adaptive
(Meignan et al., 2016)	Single	Online	Reduced	Predefined	Basic	None
(Chuang & Smith, 2017)	Single	No learning	Reduced	Predefined	Basic	None
(Ferreira et al., 2017)	Single	Online	Whole	Without distinction	Threshold	Dynamic
(Yates and Keedwell 2017)	Single	Offline	Whole	Without distinction	Stochastic	Static

Table 4
Application domains of selection hyper-heuristics.

Application Domain	*References*
Design problems	(Allen et al., 2013)
Dynamic environments	(Baykasoğlu & Ozsoydan, 2017; Kiraz et al., 2013; van der Stockt & Engelbrecht, 2014; Topcuoglu et al., 2014; Uludağ et al., 2013)
Knapsack	(Drake et al., 2016; Lassouaoui & Boughaci, 2014; Soria-Alcaraz et al., 2014a)
Puzzles and games	(Li & Kendall, 2017; Wauters et al., 2012)
Real-valued blackbox optimisation	(Caraffini et al., 2019; Damaševičius & Woźniak, 2017; Grobler et al., 2015; Tinoco & Coello, 2013)
Scheduling	(Aron et al., 2015; Asta et al., 2016a; Asta et al., 2016b; Bilgin et al., 2012; Chen et al., 2016a; Chen et al., 2017; Koulinas & Anagnostopoulos, 2013; Koulinas et al., 2014; Lin et al., 2017; Mısır et al., 2015; Mısır et al., 2012a; 2013a; Monemi et al., 2015; Pour et al., 2018; Rahimian et al., 2017; Rajni & Chana, 2013; Tsai et al., 2014; Wu et al., 2016; Zheng et al., 2015)
Search-based software engineering	(Henard et al., 2014; Jia et al., 2015; Zamli et al., 2016)
Shelf allocation	(Bai et al., 2013; Zhao et al., 2016)
Telecommunications	(Tsai et al., 2017; Yang et al., 2014)
Timetabling	(Burke et al., 2014; da Fonseca et al., 2016; Kheiri & Keedwell, 2017; Kheiri et al., 2016; Soria-Alcaraz et al., 2017; Soria-Alcaraz et al., 2014b)
Traveling salesman	(Choong et al., 2017; El Yafrani et al., 2018; Martins et al., 2017; Qu et al., 2015; Smith & Imeson, 2017; Swiercz et al., 2014)
Vehicle routing	(Akar et al., 2014; Chen et al., 2016b; Marshall et al., 2015; Mourdjis et al., 2016; Sabar et al., 2015c; Sim & Hart, 2016; Soria-Alcaraz et al., 2017; Tyasnurita et al., 2017; Yin et al., 2016)

Gümüş, Ozcan, and Atkin (2016) tuned the parameters of a generic steady state Memetic Algorithm using the Taguchi method on two arbitrarily selected instances from each of the original four problems in HyFlex. The tuned approach was applied to forty-five instances from nine domains. Each crossover, mutation and local search heuristic is selected at random during the search. The tuned Memetic Algorithm turns out to be competitive, outperforming two other Memetic Algorithm variants, ranking fourth in the CHeSC 2011 competition compared to the competition entrants, and second in the additional domains compared to the same six hyper-heuristics that were compared by Adriaensen et al. (2015).

A sample of selection hyper-heuristics using HyFlex, belonging to various categories based on the extended classification discussed in Section 2, are listed in Table 3.

5. Selection hyper-heuristics for different problem domains

In addition to the HyFlex problem domains, a large number of other problem domains have been addressed by different researchers as illustrated in Table 4. Here we have tried to group methods solving the same or similar problems together, with a final subsection containing the problems that do not fit into any of these categories.

5.1. Dynamic environments

There is a range of *dynamic environment* optimisation problems in which problem components, such as, objective function, constraints etc., may change over time, and as a result move the associated optima for a given instance. The majority of solution techniques for dynamic environment problems are evolutionary approaches (Jin & Branke, 2005). There are some recent studies investigating hyper-heuristics and their hybrids in this area.

Topcuoglu, Ucar, and Altin (2014) tested a number of selection hyper-heuristics on both the discrete generalised assignment problem, and the continuous moving peaks benchmark, a multidimensional dynamic function generator. The authors used a set of parameterised Gaussian mutation operators as low-level heuristics for the moving peaks benchmark. Choice Function combined with accepting Only Improving moves performed better than the majority of the other selection hyper-heuristics tested and a reference memory based evolutionary algorithm. Kiraz, Etaner-Uyar, and Özcan (2013) compared the performance of selection hyper-heuristics using the moving peaks benchmark. The results in this study also indicated the success of Choice Function based hyper-heuristics in solving a range of dynamic environment problems.

Uludağ et al. (2013) investigated a dual population framework, enabling exploitation of online and offline learning methods using other selection hyper-heuristics. Representative examples capturing the change dynamics are sampled to learn probability vectors for an Estimation of Distribution Algorithm in an offline learning phase. During the online learning phase, the probability vectors are used as low-level heuristics controlled by hyper-heuristics. Overall, a Greedy heuristic selection method combined with All Moves acceptance performed best across a large range of Decomposable Unitation-Based Functions with various change dynamics, including cyclic changes. Additionally, it was observed that the proposed approach outperforms other well-known techniques in almost all scenarios, except some deceptive functions.

van der Stockt and Engelbrecht (2014) implemented a simple hyper-heuristic by mixing a set of low-level population based metaheuristics, including two variants of Particle Swarm Optimisation, a Genetic Algorithm, and Differential Evolution to the moving peaks benchmark. Each selected metaheuristic is applied for a fixed number of steps and then another is chosen randomly. Baykasoğlu and Ozsoydan (2017) proposed a greedy randomised adaptive search procedure (GRASP) for solving the dynamic multi-dimensional knapsack problem. The results across the chosen problem instances with various change dynamics show that GRASP performs reasonably well when compared to a Memetic Algorithm, Differential Evolution, a swarm intelligence approach and a Reinforcement Learning based hyper-heuristic managing all of those algorithms.

5.2. Knapsack problems

Drake, Özcan, and Burke (2016) presented the idea of crossover control at two different conceptual levels, using the multidimensional knapsack problem as a testbed. This paper investigated giving responsibility to either the high-level search methodology, or the low-level heuristic operators below the domain barrier at the problem-level. Although improved performance was observed for this problem domain by controlling crossover below the domain barrier, in the case of the HyFlex framework where the domain barrier is strictly enforced, this feature needs to be incorporated into the implementation of the high-level heuristic search methodology. This issue was considered in detail in further work by Drake (2014), discussed in Section 4.1 above.

Soria-Alcaraz, Ochoa, Carpio, and Puga (2014a) used 'evolvability metrics', a measure of quality potential for low-level heuristic-solution pairs, to inform the adaptive selection of operators for a number of problems including the multiple knapsack problem. Lassouaoui and Boughaci (2014) solved the related winner determination problem, using a Choice Function - Only Improving hyper-heuristic. Given a set of bids for different subsets of items, this problem involves finding a set of winning bids that maximise the revenue of an auctioneer in a combinatorial auction. Operating over a set of five low-level heuristics, including a Stochastic Local Search operator, combining heuristics outperformed Stochastic Local Search alone on a well-known set of benchmark instances from the literature.

5.3. Puzzles and games

Puzzles and games have long been a favoured domain for researchers in artificial intelligence, so it is unsurprising that some have caught the attention of researchers working with selection hyper-heuristics. The *Eternity II* Puzzle is an edge-matching puzzle, which requires a set of patterned tiles to be placed on a grid, such that the edges of adjacent tiles share a common pattern. Wauters, Vancroonenburg, and Vanden Berghe (2012) presented the winning entrant for an international competition to solve this puzzle, where the optimisation goal is to maximise the number of matched edges. Using a set of low-level heuristics that swap and rotate tiles, Simple Random heuristic selection was combined with a number of different acceptance criteria, with secondary objective functions also used to help guide the search.

Li and Kendall (2017) introduced hyper-heuristic players for a number of games, defining high-level strategies using a variety of low-level heuristics, including iterated prisoners dilemma and repeated *Goofspiel*. The hyper-heuristic game players were shown to outperform their constituent low-level heuristics, using dynamic strategies over time. A constructive hyper-heuristic for the competitive TSP was also presented. This problem is discussed in more detail in Section 7.

5.4. Real-valued black-box benchmark function optimisation

Real parameter function optimisation has been an area of interest for swarm and evolutionary computation researchers for decades, and still maintains an active research community today. A wide range of algorithms have been developed to solve problems of this type, such as Evolution Strategies, Particle Swarm Optimisation and Differential Evolution. Recently, selection hyper-heuristics have caught the attention of researchers in this area, with methods utilising multiple low-level heuristics, in the form of either individual operators or entire metaheuristics, appearing increasingly often. Grobler, Engelbrecht, Kendall, and Yadavalli (2015) presented a set of selection hyper-heuristics operating over a set of popular metaheuristics using a shared population, each employing different strategies to adaptively change the search space of low-level heuristics during the search. Effectively balancing intensification and diversification within the heuristic search space was shown to outperform using a static set of low-level heuristics. Damaševičius and Woźniak (2017) proposed a 'state flipping' hyper-heuristic, which oscillates between two nature-inspired metaheuristics for benchmark function optimisation during a run, and applied it to eight classic benchmark functions. Hybridising the two metaheuristics offered better performance than applying each individually. Caraffini, Neri, and Epitropakis (2019) compared four Adaptive Operator Selection strategies selecting memes within a Memetic Algorithm for real parameter single-objective optimisation. Three of these were success-based adaptation strategies, rewarding memes that generate good solutions, increasing the chance of selecting those memes during the search. Results were presented using well-known benchmark sets, with significantly improved performance reported when compared to a number of existing methods from the literature.

Whereas the methods above solve unconstrained optimisation problems, Tinoco and Coello (2013) proposed a hyper-heuristic based on Differential Evolution for constrained function optimisation. Using variants of roulette wheel selection, choosing from one of twelve Differential Evolution variants at each step, their approach was demonstrated to outperform state-of-the-art Constrained Differential Evolution on a set of well-known benchmark functions.

5.5. Scheduling

In their short history, selection hyper-heuristics have been applied widely to problems in scheduling. One of the most frequently studied contexts for scheduling problems is healthcare scheduling. Mısır, Verbeeck, De Causmaecker, and Vanden Berghe (2013a) tested fourteen hyper-heuristics with different characteristics under various settings on home care scheduling, nurse rostering and patient admission scheduling problems. Further work (Mısır, Verbeeck, De Causmaecker, & Vanden Berghe, 2012a)

investigated the influence of the set of low-level heuristics available to a selection hyper-heuristic on solution quality for patient admission scheduling. Bilgin, Demeester, Misir, Vancroonenburg, and Vanden Berghe (2012) considered two of these problems, patient admission scheduling and nurse rostering, with hyper-heuristics using Great Deluge as an acceptance criterion shown to work well on well-known benchmark sets for both problems.

Asta, Özcan, and Curtois (2016b) applied a life-long learning multi-stage hyper-heuristic to personnel scheduling. This approach employs two selection hyper-heuristics, both embedding Simple Random heuristic selection but with different move acceptance methods. The algorithm consists of four phases overall which are cycled through periodically, performing tensor analysis in the first three phases to configure the algorithm to be used in the final phase. Tensor analysis is used to discover which low-level heuristics perform well with which parameter settings and move acceptance method. In the last phase, an iterated multi-stage algorithm randomly chooses between Simple Random - Improving and Equal or Simple Random - Naive Acceptance and runs that hyper-heuristic using the previously learned configuration for a fixed duration. This process is repeated until the time allocated for the final phase ends. Two variants of the approach with different memory settings for learning are tested on nurse rostering benchmarks. The results showed that including a 'forgetting' mechanism performs slightly better than remembering everything from the start of the search process, improving upon the best known results for four instances. Nurse rostering was also considered by Rahimian, Akartunalı, and Levine (2017). Their work presented a hybrid approach, combining a solution construction stage with variable neighbourhood descent operating over a search space of five low-level heuristics, before a final Integer Programming-based ruin-and-recreate phase. The proposed approach was shown to outperform two state-of-the-art approaches from the literature in many cases on well-known existing benchmarks.

The winning entrant to the CHeSC 2011 competition, AdapHH, has been discussed in detail already in Section 4.1 above. Mısır, Smet, and Vanden Berghe (2015) applied this hyper-heuristic (called Generic Intelligent Hyper-heuristic (GIHH) in their paper) to three scheduling problems operating in a unified framework: home care scheduling, routing and rostering of security guards, and maintenance personnel scheduling. This hyper-heuristic was also used by Monemi et al. (2015) for workover rig scheduling, a problem in the oil industry which requires scheduling large pieces of mobile equipment required for oil well maintenance. Here, two versions of GIHH using different learning mechanisms were compared. The solutions found were then used to 'warm start' a branch, price and cut algorithm for further optimisation. Where the optimal solutions are known, the solutions found by GIHH initially are noted to already be very close to optimality. Selection hyper-heuristics for maintenance scheduling in a number of different contexts have also been studied elsewhere. A Choice Function hyper-heuristic was used by Pour, Drake, and Burke (2018) to define the working areas of engineers performing maintenance tasks in the Danish rail network. Tasks are re-allocated from one area to another at each step using one of five low-level heuristics, selected based on Choice Function scores. Chen, Cowling, Polack, Remde, and Mourdjis (2017) used binary exponential back off, a tabu based greedy method, to manage a set of six low-level heuristics to generate maintenance schedules for preventive and corrective maintenance of an urban water drainage system over a rolling time horizon.

A hyper-heuristic Genetic Algorithm, which selects from multiple crossover and mutation operators at each generation, was used by Wu, Consoli, Minku, Ochoa, and Yao (2016) to solve software project scheduling problems. Koulinas, Kotsikas, and Anagnostopoulos (2014) used a Particle Swarm Optimisation based hyper-heuristic operating over eight low-level heuristics to solve well-known benchmarks for resource constrained project scheduling. A tabu search based hyper-heuristic algorithm was presented by Koulinas and Anagnostopoulos (2013) to solve special cases of the resource constrained project scheduling problem in the context of the construction industry. The 'Dominance-based and Roulette Wheel with an Adaptive Threshold Acceptance' (Asta, Karapetyan, Kheiri, Özcan, & Parkes, 2016a) hyper-heuristic was used as a part of a hybrid approach which won the MISTA 2013 challenge on 'multi-mode resource-constrained multi-project scheduling problems. This multi-stage selection hyper-heuristic method combines two different stages. The first stage evaluates all low-level heuristics and maintains an active subset of the best performing heuristics, using a dominance based mechanism which keeps heuristics that are not dominated by any other low-level heuristics in the subset. The heuristic dominance strategy measures the improvement in objective function value and the number of steps required to achieve that improvement. This stage also assigns a score to each low-level heuristic. The subsequent hyper-heuristic stage is invoked, using the active subset of heuristics with their associated scores. The proposed multi-stage hyper-heuristic is used as a local search method for improvement as a part of a population based hybrid approach.

Many of the scheduling problems discussed in this section so far operate over heuristics applied to discrete search spaces. There are also examples in the literature where hyper-heuristic search has been used in continuous solution space. Zheng, Zhang, Ling, and Chen (2015) considered the problem of emergency railway transportation planning, for managing evacuation in the event of disaster relief. A hyper-heuristic selecting between multiple evolutionary operators from state-of-the-art methods at each step was shown to outperform each of the constituent methods applied independently.

Selection hyper-heuristics have been applied to a wide range of scheduling problems in addition to those discussed already. Tsai, Huang, Chiang, Chiang, and Yang (2014) considered the problem of cloud scheduling, using multiple metaheuristics in a co-operative manner. Operating on a shared population, a number of existing metaheuristics (Simulated Annealing, Genetic Algorithm, Ant Colony Optimisation and Particle Swarm Optimisation) are applied to the problem in turn, with a new metaheuristic selected when the search stagnates according to one of two given measures. Combining metaheuristics was shown to outperform using each of the metaheuristics individually, in addition to outperforming other traditional rule based algorithms. Chen, Li, Yang, and Rudolph (2016a) applied an adaptive Reinforcement Learning method, inspired by quantum computing theory, to task scheduling in cluster computing to optimise performance and power consumption. Rajni and Chana (2013) used a nature-inspired population based numerical optimisation algorithm, and Aron, Chana, and Abraham (2015) employed a Particle Swarm Optimisation based hyper-heuristic, for resource provisioning in the context of grid computing. Both of these methods operated on a solution space that included both complete and partial solutions. Lin, Wang, and Li (2017) applied an evolutionary method to evolve sequences of low-level heuristics, with their approach tested on a distributed variant of the permutation flow-shop scheduling problem.

5.6. Search-based Software Engineering

Search-based Software Engineering (SBSE) is a field attracting increasing attention, where search and optimisation techniques are used to solve problems in software engineering. Some examples

of selection hyper-heuristics being applied to SBSE problems have emerged in the literature.

The goal of combinatorial interaction testing is to find a set of test cases that cover all possible combinations of values between a set of t parameters, for some small fixed value t. A Reinforcement Learning - Simulated Annealing selection hyper-heuristic to solve this problem, operating over six low-level heuristics of varying complexity, was presented by Jia, Cohen, Harman, and Petke (2015). Over a set of synthetic and real-world problem instances, with 2- and 3-way interactions, comparable performance was shown to existing methods, with particularly good results observed for real-world instances. Zamli, Alkazemi, and Kendall (2016) used a Tabu Search hyper-heuristic for the same problem, selecting from four low-level metaheuristics for t-way test suite generation, demonstrating good results for problems considering up to 6-way interactions.

Henard, Papadakis, and Le Traon (2014) used a Simple Random - Only Improving selection hyper-heuristic to select from four low-level heuristics to generate test configurations for mutation testing of software product lines. Mutation testing seeks to find a suite of test configurations that correctly identify a set of 'mutant' variants of a particular system. Combining low-level heuristics was observed to outperform random generation of test configurations, reducing the size of the test suite required and increasing the number of mutants identified.

In addition to the formulation of single-objective software engineering problems, SBSE also covers a variety of problems that are multiobjective by nature. Selection hyper-heuristics for multi-objective SBSE problems (El Kateb, Fouquet, Bourcier, & Le Traon, 2014; Guizzo, Bazargani, Paixao, & Drake, 2017a; Guizzo, Fritsche, Vergilio, & Pozo, 2015; Guizzo, Vergilio, Pozo, & Fritsche, 2017b; Kumari & Srinivas, 2016) are covered in Section 6.1.

5.7. Timetabling

Timetabling problems are a type of scheduling problem. They have also frequently been addressed using selection hyper-heuristics. Pillay (2016) provides a survey of all hyper-heuristic approaches applied to various educational timetabling problems. Here we present the recent applications of selection hyper-heuristics for educational timetabling.

Burke, Qu, and Soghier (2014) studied two-stage approaches for examination timetabling, which first construct a feasible solution, then use a hyper-heuristic to sequence perturbative low-level heuristics for improvement. The tested methods apply each heuristic one at a time in turn from an evolved sequence. An approach which adjusts the length and contents of the sequences adaptively performs particularly well on the Toronto benchmarks, but poorly on the ITC2007 dataset.

Soria-Alcaraz et al. (2014b) tested a variant of Iterated Local Search strengthened by a hyper-heuristic for post-enrollment course timetabling. The hyper-heuristic uses an Adaptive Operator Selection method based on credit assignment to control nine low-level heuristics during the improvement phase. The proposed approach did not, on the whole, perform as well as the state-of-the-art methods for the problem instances tested. However, it obtained the new best solution to one of the ITC2007 problem instances. Soria-Alcaraz, Ochoa, Sotelo-Figeroa, and Burke (2017) improved their approach by using a subset of low-level heuristics identified by performing non-parametric statistical tests and fitness landscape probing techniques. They also applied this approach to vehicle routing. However, it does not outperform the state-of-the-art methods.

da Fonseca, Santos, Toffolo, Brito, and Souza (2016) proposed a three-stage approach referred to as GOAL (Group of Optimization and Algorithms) which won the third international timetabling competition (ITC2011), based on a real-world high school timetabling problem. The first stage of the approach constructs an initial solution which is then fed into a hyper-heuristic controlling six low-level heuristics for improvement. The heuristic selection component of this hyper-heuristic chooses a heuristic with a prefixed probability. Simulated Annealing with reheating is used for move acceptance. In the last stage, an Iterated Local Search algorithm is used, employing two low-level heuristics, each selected at random during the search and accepting improving moves only. Hyper-heuristic Search Strategies and Timetabling (HySST) (Kheiri, Özcan, & Parkes, 2016) is a multi-stage hyper-heuristic approach, which also competed at ITC2011. This method combines two selection hyper-heuristics, one operates on a set of mutational low-level heuristics and the other on hill climbing heuristics. The proposed method switches between diversification and intensification stages if the candidate solution cannot be improved after a certain duration. The method employs an adaptive threshold move acceptance which accepts solutions that are a factor $(1+\epsilon)$ worse, where ϵ is a threshold that changes adaptively during the search. This solver generated the all-time-best solutions for three instances in round 1 of the ITC2011 competition, and came second in rounds 2 and 3. Kheiri and Keedwell (2017) proposed a sequence-based selection hyper-heuristic utilising a hidden Markov model, which keeps scores to represent the probability of selecting a low-level heuristic based upon the previously invoked heuristic. The scores are updated using Reinforcement Learning. The proposed hyper-heuristic outperforms GOAL, obtaining new best results for seven instances, matching the best-known results for four instances from the the ITC2011 benchmark.

A number of selection hyper-heuristics that *construct* solutions to timetabling problems are also present in the literature (Qu, Pham, Bai, & Kendall, 2015; Soghier & Qu, 2013). These methods are discussed in detail in Section 7, which is dedicated to constructive selection hyper-heuristics.

5.8. Traveling salesman

The traveling salesman problem (TSP) is arguably the most well-known combinatorial optimisation problem, with a number of variants which consider different constraints or objectives. A unified framework which is able to represent a number of TSP variants, in addition to a DNA sequencing problem found in bioinformatics and a knapsack problem, was presented by Swiercz et al. (2014). Using a set of representation specific low-level heuristics, rather than problem specific low-level heuristics, good performance was still observed despite the relative simplicity of the low-level heuristic set.

Smith and Imeson (2017) presented a large neighborhood search heuristic for the generalised traveling salesman problem, based on the highly successful method of Pisinger and Ropke (2007) for the vehicle routing problem. This framework is based on an iterative ruin-and-recreate structure, adaptively selecting heuristics to add and remove elements of a solution. Their approach was shown to outperform a number of state-of-the-art methods on several benchmark libraries, including finding new best solutions for some instances. Choong, Wong, and Lim (2017) used a Modified Choice Function to select between low-level heuristics within a swarm based evolutionary algorithm, applying their method to the TSP benchmarks within HyFlex. No direct performance comparison to other CHeSC entrants, or other methods in the literature was given.

The traveling thief problem (TTP) is a recently proposed combinatorial optimisation problem which combines the classical traveling salesman problem with a knapsack problem. A selection hyper-heuristic based on the existing Estimation of Distribution Algorithm hyper-heuristic of Qu et al. (2015) was presented by

Martins et al. (2017). The proposed framework sampled sequences of low-level heuristics to apply to small and medium-sized instances of the TTP. Another hyper-heuristic approach was introduced by El Yafrani et al. (2018), using Genetic Programming to evolve selection hyper-heuristics to address the TTP. This paper is discussed in detail in Section 8.

5.9. Vehicle Routing

Vehicle routing problems (VRPs) are an area in which selection hyper-heuristic methods have been particularly successful, with the Adaptive Very Large Neighbourhood Search (AVLNS) approach of Pisinger and Ropke (2007) achieving state-of-the-art results for multiple VRP variants back in 2007.

Sim and Hart (2016) combined the two paradigms of selection and generation heuristics, using Genetic Programming to generate initialisation and perturbation low-level heuristics for the VRP, before employing the generated low-level heuristics within a selection hyper-heuristic framework. After initialising a population of solutions using the set of constructive low-level heuristics generated with Genetic Programming, a Simple Random selection hyper-heuristic operating within a Memetic Algorithm framework was applied. Using the set of perturbation low-level heuristics generated by Genetic Programming, at each step one crossover heuristic, one mutation heuristic and one hill climbing heuristic are selected at random. The low-level heuristics are then applied in that order with a given probability for each. The proposed method was able to outperform a previous method based on Grammatical Evolution on some instances of the well-known Solomon benchmark VRP instances.

Marshall, Johnston, and Zhang (2015) compared forty-eight different selection method-acceptance criteria combinations, consisting of six selection methods and eight move acceptance criteria, over randomly generated instances of the Capacitated Vehicle Routing Problem (CVRP). In this problem domain, using a set of twelve low-level heuristics, selection methods which considered execution time, penalising low-level heuristics which took longer to execute, were more successful than those that didn't. Exponential Monte Carlo and Improving or Equal acceptance performed particularly poorly, with Simulated Annealing and a Naïve move acceptance method which accepts all improving solutions and non-improving solutions 50% of the time showing good performance.

Yin, Lyu, and Chuang (2016) presented a coevolutionary approach to solve an integrated vehicle routing and scheduling problem for cross-dock buffering in warehouses. Whilst the scheduling component was tackled using a population based Ant Colony Optimisation method, a hyper-heuristic using rule-based selection with Simulated Annealing move acceptance was employed to co-evolve a solution to the routing component. The rules for the behaviour of the selection hyper-heuristic depend on the current state of the solution, with certain low-level heuristics selected if the solution is infeasible.

The periodic vehicle routing problem, for which daily routes are required based on customer behaviour, was studied by Chen, Mourdjis, Polack, Cowling, and Remde (2016b). This work tested a number of different hyper-heuristic frameworks, similar to the F_A and F_C frameworks presented by Özcan et al. (2008), using Simple Random, Random Descent, Reinforcement Learning, Choice Function and Binary Exponential Backoff heuristic selection with Only Improving move acceptance. In line with the observations made by Özcan et al. (2008), the inclusion of a dedicated local search phase improved the performance of the hyper-heuristics tested.

Sabar, Zhang, and Song (2015c) presented a 'math-hyper-heuristic', combining an exact approach based on column generation, with a Multi-armed Bandit selection, Exponential Monte Carlo move acceptance selection hyper-heuristic, for the VRP with time windows (VRPTW). Using the solutions to a set of subproblems solved by column generation, the hyper-heuristic selects from a set of low-level heuristics to combine them into a single solution to the complete problem. Using the exact method to 'warm start' the hyper-heuristic was shown to outperform an existing constructive heuristic from the literature. The combined math-hyper-heuristic was shown to outperform state-of-the-art metaheuristics from the literature on large-scale VRPTW instances.

A supermarket resupply problem was modelled as a dynamic pickup and delivery problem with 'soft' time windows (PDPSTW) by Mourdjis, Chen, Polack, Cowling, and Robinson (2016). Four hyper-heuristics were used to select from a set of local search operators within an Iterated Local Search framework. Using both benchmark data-sets and real-world data, an approach using Variable Neighbourhood Descent with memory was shown to outperform a Choice Function variant, Tabu Search and Random Descent heuristic selection.

Tyasnurita, Özcan, and John (2017) presented an apprenticeship learning hyper-heuristic framework for the open vehicle routing problem (OVRP). In this framework, an 'expert' hyper-heuristic is trained on a subset of instances, whilst an 'apprentice' hyper-heuristic learns by observing the search process and is then applied to a set of unseen instances. Their results showed that it is possible to produce an apprentice hyper-heuristic that performs better than the original expert, using the distance between solutions as additional information to guide the apprentice.

Ahmed, Mumford, and Kheiri (2019) evaluated the performance of a range of selection hyper-heuristics combining different reusable components for the urban transit routing problem. The results over a set of benchmark instances demonstrate the strength of an approach which combines sequence-based heuristic selection with Great Deluge move acceptance. This method is very successful, outperforming the current known state-of-the art results within much shorter execution times.

5.10. Other application domains

In the previous sub-sections, we have collected together selection hyper-heuristic methods applied to a number of well-known problem domains. However, the variety of problem domains tackled across the literature is far greater than this. Here we will discuss other papers that do not fit into the categories outlined above, highlighting the diversity of problems that have been solved by selection hyper-heuristics in recent times.

Shelf space allocation, where the goal is to maximise the utilisation of search space according to some quality measure, is an important problem in retail. Bai, Van Woensel, Kendall, and Burke (2013) investigated a two-dimensional shelf space allocation model, using Simple Random and Reinforcement Learning heuristic selection with Simulated Annealing acceptance. Inspired by this work, Zhao, Zhou, and Wahab (2016) presented an extended model, designed to better reflect the practicalities of the real-world problem. They used a Simple Random - Simulated Annealing hyper-heuristic operating over a set of low-level heuristics restricted to feasible regions of the search space to solve this problem.

Telecommunications is another high impact field. Yang, Peng, Jiang, Wang, and Li (2014) introduced a hyper-heuristic Genetic Algorithm to solve the frequency assignment problem (FAP). This approach evolves sequences of low-level heuristics to be applied to the current solution during the search. Tsai, Chang, Hu, and Chiang (2017) used multiple metaheuristics, randomly selected and applied for a fixed number of iterations, to address the problem of selecting a *cluster head* in wireless sensor networks.

Selection hyper-heuristics have been applied to a number of problems in industrial design, many of which are continuous rather

Table 5
Application domains of multiobjective selection hyper-heuristics.

Application domain	*Reference(s)*
Benchmark function optimisation	(de Carvalho & Sichman, 2017; Castro Jr & Pozo, 2015; Gómez & Coello, 2017; Maashi et al., 2015; Maashi et al., 2014; Segura et al., 2012; Vazquez-Rodriguez & Petrovic, 2013; Walker & Keedwell, 2016)
Engineering design	(Hitomi & Selva, 2016; McClymont et al., 2013)
Graph colouring	(Elhag & Özcan, 2015)
Job shop scheduling	(Grobler & Engelbrecht, 2016)
Packing	(Segredo et al., 2014)
Search-based software engineering	(El Kateb et al., 2014; Gonçalves et al., 2015; Guizzo et al., 2017a; Guizzo et al., 2015; Guizzo et al., 2017b; Kumari & Srinivas, 2016)
Timetabling	(Elhag & Özcan, 2015; Muklason et al., 2017)
Vehicle crashworthiness	(Maashi et al., 2015; Maashi et al., 2014)
Windfarm layout optimisation	(Li et al., 2017)

than discrete optimisation problems. Allen, Coates, and Trevelyan (2013) applied a variety of selection hyper-heuristics to aircraft structural design optimisation.

Splines are piecewise polynomial functions which can be constructed from a set of control points. Using a set of low-level heuristics operating over a space of control points, representing unmanned aerial vehicle routes, Akar, Topcuoglu, and Ermis (2014) compared a number of well-known selection hyper-heuristics and a Genetic Algorithm. Using the OneMax and Gap-Path functions as examples, Lehre and Özcan (2013) analysed the expected runtime of a simple selection hyper-heuristic. This work concluded that in the case of some problem domains, mixing low-level heuristics can be more effective than using a single low-level heuristic. Alanazi and Lehre (2014) also analysed the runtime of selection hyper-heuristics, comparing different learning mechanisms commonly used in the literature. Using the simple LeadingOnes function as an example, similar performance was observed for all four learning mechanisms tested. Sabar, Turky, Song, and Sattar (2017) tuned deep belief networks for image recognition using a Multi-armed Bandit - Monte Carlo selection hyper-heuristic, outperforming the results reported by existing metaheuristic methods in the literature. Kampouridis, Alsheddy, and Tsang (2013) presented a framework which applied low-level heuristics to Genetic Programming trees for financial forecasting. This framework selects from a set of up to fourteen low-level heuristics to modify decision trees, using a roulette wheel based Reinforcement Learning scheme, and demonstrated improved performance over a well-known existing tool.

6. Selection hyper-heuristics for multiobjective optimisation

Most of the hyper-heuristics used for multiobjective optimisation are generative, often based on Genetic Programming, and particularly applied to scheduling problems (Branke et al., 2016). However there are a growing number of studies on multiobjective *selection* hyper-heuristics focusing on two separate approaches: (i) selection hyper-heuristics managing components, such as, low-level heuristics/operators of a particular multiobjective optimisation algorithm, and (ii) selection hyper-heuristics managing and mixing a set of low-level multiobjective metaheuristics under an iterated cooperative search framework. Multiobjective Evolutionary Algorithms (MOEAs) are the most commonly used metaheuristics in the field. Table 5 provides a summary of the application domains where multiobjective hyper-heuristics are utilised.

6.1. Controlling multiple components of a multiobjective algorithm

Segura, Segredo, and León (2012) studied a parallel hyper-heuristic approach based on an island model, converting single-objective large-scale continuous optimisation benchmark functions into multiobjective problems through multiobjectivisation. The authors applied multiple variations of NSGA-II executed in parallel. Those variations included twenty-four configurations of NSGA-II combining three crossover operators, two mutation operators, and four different multiobjectivisations. The selection hyper-heuristic scores each configuration considering the improvement on the best solution achieved so far. It is run on the master island, mapping the *promising* configurations with higher scores to worker islands with higher probabilities during the search. The results show that superlinear speedups can be achieved in some cases, and that multiobjectivisation works well overall.

Elhag and Özcan (2015) presented a general two-objective selection hyper-heuristic approach for grouping problems, requiring the partitioning of a given set of items, while minimising the number of groups and optimising another objective simultaneously. This study investigated the performance of combinations of various heuristic selection and move acceptance methods applied to nineteen graph colouring and five examination timetabling benchmark problem instances. The results indicate the effectiveness of a selection hyper-heuristic consisting of Reinforcement Learning heuristic selection and the move acceptance method of AdapHH (Mısır et al., 2012b) on both domains.

McClymont, Keedwell, Savić, and Randall-Smith (2013) presented a heuristic selection mechanism based on Markov chains and Reinforcement Learning, embedded into the well-known NSGA-II and SPEA2 MOEAs, applied to the optimisation of water distribution network design. Adding a set of four mutational low-level heuristics improved performance over the original MOEAs. In this work, the ratio of dominating solutions produced by each heuristic was used to measure performance.

El Kateb et al. (2014) introduced a framework to select from multiple mutation operators within MOEAs when optimising software deployment in a cloud environment. At each generation, a score is calculated for each mutation operator, applying the operator with the best score for that generation. Score-based selection was shown to outperform Simple Random selection of the mutation operator. Kumari and Srinivas (2016) proposed a multiobjective hyper-heuristic (MHypEA) to solve the multiobjective software module clustering problem. Operating a set of twelve low-level heuristics, consisting of different combinations of selection, crossover and mutation operators, applied to a population of solutions, MHypEA uses a roulette wheel based Reinforcement Learning strategy to select a low-level heuristic at each step. Compared to NSGA-II, MHypEA was shown to achieve better performance in fewer evaluations on the problem instances tested. Guizzo et al. (2017a, 2015) used two heuristic selection methods, Choice Function and Multi-armed Bandit, within an NSGA-II framework to solve the multiobjective integration and test order problem. The hyper-heuristics operated over a set of nine low-level heuristics consisting of combinations of a crossover and a mutation operator. Using seven Java-based systems with two objectives, experiments showed that hyper-heuristic selection of

crossover and mutation operators within NSGA-II outperformed a traditional NSGA-II implementation. An extension was provided by Guizzo et al. (2017b), formulating the problem as a many-objective problem and comparing to a number of state-of-the-art MOEAs. Strickler, Lima, Vergilio, and Pozo (2016) investigated a similar framework to the original paper of Guizzo et al. (2015), performing either Simple Random or Multi-armed Bandit selection of twelve low-level heuristics within NSGA-II, when optimising the products derived from a feature model to test software product lines with multiple objectives. Their hyper-heuristic outperformed a number of well-known MOEAs, including traditional NSGA-II and SPEA2.

Gonçalves, Kuk, Almeida, and Venske (2015) incorporated a variant of Choice Function heuristic selection to control five Differential Evolution operators at the lower level within MOEA/D. The multiobjective hyper-heuristic improves upon the performance of the generic MOEA/D using a single operator, when applied to ten unconstrained benchmark functions with two and three objectives. Walker and Keedwell (2016) used a previous selection hyper-heuristic (Kheiri & Keedwell, 2015) controlling seven low-level heuristics within an MOEA for many-objective optimisation. The analyses using three different comparison operators as alternatives to dominance on a subset of the DTLZ test suite show that the *favour relation* and *hyper-volume* indicators are the best choices.

Hitomi and Selva (2016) considered a multiobjective design problem for an Earth observation satellite system, where a number of instruments must be assigned to an orbit, with the goal of minimising cost whilst maximising the scientific benefit of the assignment. Based within the framework of ϵ-MOEA, using five domain-specific heuristics, an Adaptive Pursuit strategy was used to assign probabilities of selecting different operators. Low-level heuristic performance was measured by the number of solutions generated by it that are added to the ϵ-MOEA archive.

Muklason, Parkes, Özcan, McCollum, and McMullan (2017) applied a three-stage multiobjective approach for examination timetabling, optimising the standard objective along with *fairness* within a cohort of students. In the first stage, a set of feasible initial solutions is generated by employing a squeaky wheel method. In the second and third stages, a search is performed using Reinforcement Learning heuristic selection and Great Deluge move acceptance controlling fourteen low-level heuristics. In the second stage, the standard objective is optimised while in the third stage both of the objectives are optimised simultaneously. The experimental results on three well-known benchmarks indicate the effectiveness of the proposed approach for multiobjective examination timetabling.

Gómez and Coello (2017) presented a many objective approach using a hyper-heuristic which extends an elitist Genetic Algorithm, denoted as MOMBI-II using the R2 performance indicator. The selection hyper-heuristic chooses from seven scalarising functions during the search. The proposed approach performs significantly better than NSGA-III, MOEA/D and MOMBI-II across the ZDT, DTLZ and WFG benchmark functions.

6.2. Controlling multiple metaheuristics

Vazquez-Rodriguez and Petrovic (2013) proposed variants of multiobjective genetic algorithms combining the rank indicators of NSGA-II, SPEA2 and two variants of IBEA for selection. Each indicator is associated with a probability which is set based on mixture experiments. At each iteration, the individuals are subdivided into four subpopulations, one per indicator, in a random manner based on their probabilities. The mating pool is formed using binary tournament applied to the individuals from the same subpopulation. Then, the remaining evolutionary processes of crossover and mutation follow. The results on a set of DTLZ, LZ07F and ZDT benchmark functions show that dynamically updating the probabilities via mixture experiments is the best approach, outperforming all of the other variants as well as NSGA-II, SPEA2 and IBEA.

Segredo, Segura, and León (2014) extended the work of Segura et al. (2012) and investigated parallel hyper-heuristics adaptively applying various configurations of a choice of multiobjective metaheuristics, including NSGA-II and SPEA2. The results on two instances of a 2D packing problem show that multiobjectivisation yields improved performance.

Maashi, Özcan, and Kendall (2014) employed a Choice Function based hyper-heuristic (CF-HH) selecting from three low-level MOEAs, namely NSGA-II, SPEA2, and MOGA, at each decision point during the search, and then applying the selected MOEA for a fixed duration. The resultant population is accepted as a whole at each decision point and fed into the next stage as the initial population. The proposed methodology outperforms each MOEA when run on its own as well as the AMALGAM (a multialgorithm, genetically adaptive multiobjective) approach (Vrugt & Robinson, 2007) on the WFG test suite. AMALGAM enables the use of multiple multiobjective approaches simultaneously and forms an offspring pool where each constituent algorithm contributes in proportion to its individual past performance. CF-HH was also tested on the real-world vehicle crashworthiness design problem. Although the performance of CF-HH is still superior to all individual MOEAs, AMALGAM delivers a slightly better better performance on the three-objective vehicle crashworthiness problem. Maashi, Kendall, and Özcan (2015) extended the previous study introducing an acceptance method into the hyper-heuristic approach. Choice Function metaheuristic selection is tested in combination with Great Deluge and Late Acceptance methods. The results show the effectiveness of the Choice Function - Great Deluge hyper-heuristic, which outperformed all previously tested approaches for multi-objective optimisation on the WFG and vehicle crashworthiness problems, including its bi-objective variants.

Grobler and Engelbrecht (2016) applied multiple continuous optimisation metaheuristics, including variants of Particle Swarm Optimisation and Differential Evolution, to a shared population of solutions for a multiobjective job shop scheduling problem. Li, Özcan, and John (2017) explored and showed the effectiveness of nine different selection hyper-heuristics controlling NSGA-II, SPEA2 and IBEA, deciding which one to invoke at each decision point for a fixed number of generations for various multiobjective wind farm optimisation problems.

Castro Jr and Pozo (2015) tested a multiobjective Particle Swarm Optimisation method, embedding a variant of a Choice Function hyper-heuristic on a set of DTLZ benchmark functions with dimensions varying from 2 to 20. The hyper-heuristic chooses from two archiving strategies in combination with three leader selection methods during the search process based on the R2 performance indicator. The results indicate the success of the proposed approach, even outperforming the state-of-the-art methodology MOEA/D-DRA for many objective optimisation in selected cases.

de Carvalho and Sichman (2017) presented an agent-based hyper-heuristic mixing NSGA-II, SPEA2 and IBEA based on Copeland voting, considering five performance indicators: hypervolume, spread, generational distance, inverted generational distance, and ratio of non-dominated solutions. The approach splits the whole population using Copeland voting scores into three subpopulations on which each MOEA operates. The number of individuals in a subpopulation is aligned with the Copeland ranking of each MOEA. The results on the WFG suite with two and three objectives show that the proposed agent-based hyper-heuristic is capable of identifying the best MOEA for a given instance, performing competitively.

Table 6
Application domains of constructive selection hyper-heuristics.

Application domain	*Reference(s)*
Bin packing	(Gomez & Terashima-Marín, 2018; López-Camacho et al., 2014; Pillay, 2012; Thomas & Chaudhari, 2014)
Constraint satisfaction	(Crawford et al., 2013; Gutierrez-Rodríguez et al., 2017; Ortiz-Bayliss et al., 2013; 2016; Rosales-Pérez et al., 2017)
Competitive travelling salesman	(Kendall & Li, 2013)
Exam timetabling	(Qu et al., 2015; Soghier & Qu, 2013)
Flow shop scheduling	(Salhi & Rodríguez, 2014)
Neural network construction	(Gascón-Moreno et al., 2013)
Production scheduling	(Li et al., 2015; Li et al., 2016)
Puzzles and games	(Salcedo-Sanz et al., 2014)

7. Constructive selection hyper-heuristics

The vast majority of the selection hyper-heuristics discussed in this paper to this point, operate over sets of perturbative low-level heuristics exploring a space of complete solutions to optimisation problems. There are also a small number of papers in the literature which present selection hyper-heuristics choosing from a set of constructive low-level heuristics to build solutions from empty or partial solutions. Many methods of this type utilise an evolutionary algorithm to evolve sequences of low-level heuristics to apply during solution construction. The application domains that methods of this nature have been applied to recently are summarised in Table 6.

Constructive selection hyper-heuristics have been used to solve educational timetabling problems for over a decade (Burke, McCollum, Meisels, Petrovic, & Qu, 2007). More recently, Soghier and Qu (2013) presented a hybrid approach for exam timetabling, using classical graph colouring heuristics to select an exam to add to the timetable, before using bin packing heuristics to allocate a time slot and room. Combining low-level heuristics in this way was shown to offer improved performance over applying individual heuristics on the International Timetabling Competition (ITC 2007) benchmark instances. Qu et al. (2015) used a simple Estimation of Distribution Algorithm, a Univariate Marginal Distribution Algorithm (UMDA), to generate probability distributions from which to derive sequences of low-level heuristics at different stages of a search. Using five well-known graph colouring low-level heuristics, each low-level heuristic in a sequence is applied consecutively to assign an exam to a time slot. The quality of solutions found was shown to be competitive with existing hyper-heuristic approaches to construct timetables for the Carter benchmark set.

Another area that has previously seen a high-level of research interest for constructive selection hyper-heuristics is constraint satisfaction problems (CSPs). The order in which variables are selected to be instantiated can have a significant impact on the cost of computing the solution of a CSP. However, although many variable ordering heuristics exist, predicting the performance of a heuristic for a particular problem in advance can be difficult. Ortiz-Bayliss, Terashima-Marín, and Conant-Pablos (2013) used learning vector quantization (LVQ), a type of supervised neural network, to learn a series of rules mapping between the features of the region of search space currently being explored and an appropriate heuristic action to take at that point. Based on the constraint density and constraint tightness of the current problem state, one of four low-level heuristics for variable ordering is used. Based on a similar overall framework, Ortiz-Bayliss et al. (2016) used a variable length Genetic Algorithm, encoding more complex rules using a greater number of features and heuristic actions. Each chromosome in the Genetic Algorithm consists of ten values, nine pertaining to landscape features and a tenth indicating which one of seven low-level heuristics for variable ordering should be used when these landscape features are encountered. When solving a CSP, the chromosome in the Genetic Algorithm with landscape features closest to the current solution state is found, and the corresponding low-level heuristic used. Gutierrez-Rodríguez et al. (2017) extended this work by reducing the size of the low-level heuristic set by heuristic filtering. Their experimentation identified two particularly strong heuristics, able to significantly outperform the other heuristics tested, either individually or combined within a hyper-heuristic framework. Rosales-Pérez, Gutiérrez-Rodríguez, Ortiz-Bayliss, Terashima-Marín, and Coello (2017) presented a co-evolutionary approach for heuristic selection, which identifies subsets of heuristics that perform well for certain instances, using supervised multilabel classification. This approach was shown to outperform the previous work of Ortiz-Bayliss et al. (2016), significantly reducing the time taken to solve a set of known CSP benchmarks.

Crawford et al. (2013) used Choice Function variants to adaptively rank eight enumeration strategies during the process of solving CSPs, where the set of enumeration strategies consists of combinations of variable and value selection heuristics. Each Choice Function variant is composed of a number of weighted indicators (such as number of visited nodes, number of backtracks, number of steps etc.), used to assess the performance at intermediate stages of the search process. The weighting for each of these indicators is controlled by Particle Swarm Optimisation. The proposed method is able to find good solutions on average across different problems (N-queens, Magic Square and Latin Square).

López-Camacho, Terashima-Marin, Ross, and Ochoa (2014) presented a unified framework for solving one and two-dimensional, regular and irregular bin packing problems. A set of six low-level heuristics, to decide which object to place next, and where to place it, were used to iteratively construct solutions. A Genetic Algorithm was used at the high-level, evolving a set of rules to govern which heuristic to apply in a given solution state. The framework was able to generalise well across a variety of problem instances without additional parameter tuning, outperforming the constituent low-level heuristics. Thomas and Chaudhari (2014) also considered two-dimensional bin packing, using a Genetic Algorithm to select a subset of items forming a sub-problem, then selecting a placement strategy from three low-level heuristics via a greedy method. Pillay (2012) evolved disposable heuristics for bin packing problems, consisting of sequences of known low-level heuristics. Again the evolved hyper-heuristics were observed to be superior to applying a single low-level heuristic repeatedly. Sequences of low-level heuristics were also evolved using a Genetic Algorithm by Salhi and Rodríguez (2014), this time for constructing solutions for a flow show scheduling problem variant. This method was shown to perform very well, outperforming a large number of existing strategies from the literature. Gomez and Terashima-Marín (2018) evolved rules for low-level heuristic selection when solving multiobjective two-dimensional bin packing problems. Based on a similar framework to the one used by Ortiz-Bayliss et al. (2016) for constructing solutions to CSPs, a set of rules are evolved, mapping different regions of the search space with particular properties to heuristic actions. Depending on the state of the current solution, one of forty operators, consisting of an ordering heuristic in combination with a packing heuristic, is selected and used to pack the

Table 7
Application domains of generation of selection hyper-heuristics.

Application domain	*Reference(s)*
Bin packing	(Asta et al., 2013b)
CHeSC/HyFlex	(Adriaensen et al., 2014a; 2014b; Choong et al., 2018; Sabar et al., 2015a; Sabar & Kendall, 2015)
Exam timetabling	(Sabar et al., 2013; 2015b)
Protein structure prediction	(Fontoura et al., 2017)
Traveling thief problem	(El Yafrani et al., 2018)
Vehicle routing	(Sabar et al., 2013; 2015b; Tyasnurita et al., 2017)

next item. Significantly improved performance over using single heuristics was observed.

A novel variant of the TSP, the competitive TSP (CTSP) was introduced by Kendall and Li (2013), where a number of salesmen attempt to visit a number of cities in a non-cooperative manner. If they are the first to visit a city they receive a payoff, so must consider the tours of other salesman when devising their route. Agents take turns selecting from a set of five low-level heuristics to construct a tour, one city at a time. The hyper-heuristic approach presented was shown to be able to quickly generate good approximate solutions for the problem. *Jawbreaker* is a puzzle game consisting of a grid of colored balls, the objective is to clear the grid, by eliminating connected balls of the same colour. The evolutionary hyper-heuristic framework of Salcedo-Sanz, Matías-Román, Jiménez-Fernández, Portilla-Figueras, and Cuadra (2014) could be considered to be constructive in nature, as it exhibits many of the same characteristics of other hyper-heuristic methods of this type. This work used a set of nineteen low-level heuristics to make moves in Jawbreaker, evolving sequences of low-level heuristics to be applied sequentially to the current game state.

An Ant Colony Optimisation based hyper-heuristic was presented by Li, Li, Meng, and Tian (2015). Their method searched a space of assignment and sequencing low-level heuristics for constructing solutions to a production scheduling problem in cellular manufacturing systems requiring 'intercell' transfers. The performance of the hyper-heuristic method was shown to scale considerably better than CPLEX as the size of problem instance increased. A similar problem was tackled by Li, Zhan, Zheng, Li, and Kaku (2016), who presented a bi-level approach to generate and select combinations of heuristic rules. Genetic Programming was used to evolve candidate rules to form a search space for a Genetic Algorithm based selection hyper-heuristic to operate over. The evolved system exhibited strong performance in terms of both solution quality and execution time. Gascón-Moreno, Salcedo-Sanz, Saavedra-Moreno, Carro-Calvo, and Portilla-Figueras (2013) used an evolutionary-based hyper-heuristic to evolve sequences of low-level rules to construct each layer of Group Method of Data Handling (GMDH) neural networks. Improved performance was shown over a classical GMDH approach applied to real-world prediction problems.

8. Automated design of selection hyper-heuristics

In previous classifications hyper-heuristic methods have typically been broadly separated into two categories, selection hyper-heuristics and generation hyper-heuristics (Burke et al., 2013; 2010). Although we focus on selection hyper-heuristics within this paper, this section is dedicated to heuristic generation methods, surveying recent work that has sought to *generate* selection hyper-heuristics or components. A summary of the application areas for such methods is given in Table 7.

Genetic Programming (GP) has been associated with generation hyper-heuristics for a number of years (Burke et al., 2009) so it is no surprise that work exists using GP to generate selection hyper-heuristics. El Yafrani et al. (2018) used GP to generate selection hyper-heuristics for the traveling thief problem, a combination of the travelling salesman problem and the knapsack problem. Hyper-heuristics were evolved in both an offline manner, using a train-and-test approach, and an online manner, evolving 'disposable' hyper-heuristics for a single instance. The tailored 'disposable' hyper-heuristics were shown to achieve better results than the offline-tuned hyper-heuristics, and a baseline Genetic Algorithm. Strong performance was observed compared to existing state-of-the-art methods on some larger benchmark problem instances. Fontoura, Pozo, and Santana (2017) used Grammatical Evolution to evolve, and co-evolve, heuristic selection and move acceptance criteria forming selection hyper-heuristics for protein structure prediction. The hyper-heuristics were evolved using a search space of heuristic components and six low-level heuristics, and were trained on three instances from a benchmark set of eleven problems. Best results were observed by evolving a heuristic selection method, using a fixed Improving and Equal move acceptance strategy. Poor performance was observed when co-evolving selection method and move acceptance criteria concurrently. However, the search space when doing so is much larger in this case than with a fixed move acceptance strategy. Despite being evolved specifically for this problem domain, the proposed method was outperformed on average by the AdapHH hyper-heuristic of Mısır et al. (2012b) over the eleven problem instances tested.

Adriaensen, Brys, and Nowé (2014a) performed a meta-level search over a set of potential design decisions that could be made when developing a simple selection hyper-heuristic for the HyFlex framework. An Iterated Local Search (ILS) procedure is used to explore the space of selection hyper-heuristics, with greater computational time given to evaluating higher-quality configurations. The best method found, FS-ILS (Adriaensen et al., 2014b), has been discussed in detail in Section 4. Choong, Wong, and Lim (2018) used Reinforcement Learning, with a Q-learning based feedback mechanism, to select components of ILS based selection hyper-heuristics from a set of thirty actions, consisting of five heuristic selection and six move acceptance criteria. The available computational time is split up into n equal 'episodes', with n decided via offline parameter tuning directly on the competition instances from CHeSC 2011. Each episode consists of iteratively applying a selected heuristic selection - move acceptance combination in an ILS framework, with Variable Neighborhood Descent applied after a selected perturbative heuristic is applied. Good performance was observed in three of the six CHeSC problem domains (SAT, FS, TSP). However, poor performance was reported in the remaining three domains. Sabar and Kendall (2015) used Monte Carlo Tree Search (MCTS) to generate heuristic selection strategies, specifically tuning their method for the CHeSC 2011 competition instances using a parameter tuning method from the literature. Sequences of low-level heuristics are generated in an online manner during the search using MCTS and applied to a population of solutions. At each step an Exponential Monte Carlo acceptance criterion is used to decide whether to accept non-improving solutions. The authors made use of the intensity of mutation and depth of search parameters included in HyFlex to generate different low-level heuristics. However, how these parameters are tuned/controlled and how many low-level heuristics are generated is not specified. As this method performs extensive offline

tuning on a significant proportion of competition instances (the training and test sets are not independent), a direct comparison to the CHeSC 2011 competitors cannot be made. However, strong results are reported, particularly in the SAT and PS problem domains. A similar overall framework was used by Sabar, Ayob, Kendall, and Qu (2015a), also using a population of solutions. Gene Expression Programming was employed to co-evolve a selection method and acceptance criterion for each individual problem instance, yielding disposable hyper-heuristics for particular problem instances. Again, a direct comparison cannot be made to the CHeSC 2011 competitors due to the vast difference in computational effort used by this method and those in the competition. Gene Expression Programming was also used by Sabar, Ayob, Kendall, and Qu (2015b), this time to evolve the acceptance criteria of a selection hyper-heuristic based on Multi-armed Bandit heuristic selection. Results were presented for instances of exam timetabling and dynamic vehicle routing problems, with strong performance reported compared to existing methods in the literature. Sabar, Ayob, Kendall, and Qu (2013) presented a Grammatical Evolution hyper-heuristic to generate selection hyper-heuristics from existing high-level components for exam timetabling and capacitated vehicle routing problem benchmarks. Karapetyan, Punnen, and Parkes (2017) introduced Conditional Markov Chain Search, a method based on evolving transition matrices to configure combinations of metaheuristic components, applied to the bipartite boolean quadratic programming problem. Although it is presented as a general method, the evolved transition matrices can represent the probability of selecting different heuristics in a selection hyper-heuristic.

Apprenticeship learning embodying various machine learning algorithms is a well-known technique in control and robotics, used for generalising the demonstrations provided by an expert (Abbeel & Ng, 2004). Initially, Asta, Özcan, Parkes, and Etaner-Uyar (2013b) trained k-means classifiers as generation hyper-heuristics using the data obtained from a Genetic Algorithm expert for automatically creating online bin packing heuristics. The trained method performed better than the expert in a few cases and outperformed the human designed heuristic in all cases. This work led to the study of Tyasnurita et al. (2017) who trained a time delay neural network using two sets of data obtained from the Modified Choice Function hyper-heuristic (Drake, 2014) as an expert, generating two new selection hyper-heuristics for solving the open vehicle routing problem. The first training dataset contains the changes in the objective values between solutions after application of low-level heuristics, while the second dataset includes the distance between those solutions as additional information. The empirical results show that the generated selection hyper-heuristics generalise well and perform better than the expert overall. More importantly, the inclusion of additional information during training yields an improved performance in the 'new' selection hyper-heuristic.

9. Conclusion and remarks

In this section, we will discuss some of the challenges and limitations of contemporary selection hyper-heuristics, with a focus on the HyFlex framework. We will highlight some of the efforts made to overcome these issues, as well as some potential avenues for future research directions.

An oft-cited criticism of selection hyper-heuristics is the lack of flexibility when it comes to the domain barrier. In its purest sense, as is the case with HyFlex, the domain barrier is opaque, with only objective function values being allowed to pass to the high-level search strategy and no inter-instance learning taking place. Here, the challenge is to balance the trade-off between providing a greater level of information exchange and maintaining a clear split between the problem domain and high-level solution methodology, whilst retaining the same level of plug-and-play modularity. Doing so successfully will aid one of the original goals of hyper-heuristic research: to develop powerful, more general, solver control modules without losing domain-independence. Many suggestions for extensions to permit more information to be passed across the domain barrier, without loss of domain independence, have been proposed. Swan, De Causmaecker, Martin, and Özcan (2018) argued that maintaining the domain barrier in the strictest sense is not necessary for the sake of increased generality, and that it is possible to make use of a much richer set of problem independent information than solely objective function value. *XCSP*, an XML-based format for representing constraint programming problems is suggested as a potential means of exchanging cross-domain knowledge in a domain independent manner.

A variety of potential extensions to the HyFlex interface were proposed by Parkes, Özcan, and Karapetyan (2015), designed to enable better support for applying data science techniques to optimisation. In line with the arguments made by Swan et al. (2018), the core goal is to provide support for increased exchange of useful information between the domain and search control layers by removing the barrier, imposing proper interfaces for re-usability and beyond. The suggestions include: • A richer set of *annotations*. Extending the low-level heuristic annotations beyond the existing limited set of mutation, local search, ruin-recreate and crossover operators, to provide more information regarding operator behaviour. • Providing *solution features* in a domain independent manner. These could potentially be used as a surrogate for objective function value when this is expensive to compute, or to assess the characteristics of the region of search space being explored. • Improved *distance metrics*. Currently it is only possible to compare two solutions based on objective value. As there is not always a direct correlation between the locality of solutions in representation space and objective space, a poor quality solution in terms of objective function value may still be close to high-quality solutions in the search space. This is not captured when comparing solely on objective function values. • Exposing *instance features*. Providing useful information about the features of the current instance, such as size, density or number of constraints. This could inform the high-level search method of the relative difficulty of a problem or the nature of the search landscape. • *Multiobjective support* within an extended HyFlex interface. Given the previous modifications, a hyper-heuristic with access to distance metrics for a population of solutions and an objective function would be able to measure the quality of the Pareto front, and adapt the search process as required. As discussed previously, this could be at one of two levels, either controlling multiple high-level multiobjective metaheuristics or controlling multiple low-level operators within a single multiobjective method.

Pappa et al. (2014) previously explored the intersection of the fields of machine meta-learning and hyper-heuristic optimisation, predicting an increase in cross-fertilisation between these two areas. However, this is potential is still as yet unfulfilled. Although their paper focused quite specifically on the roles of evolutionary algorithms and learning, the same logic can be applied to single-point search methods, such as the selection hyper-heuristics discussed here. Permitting a more extensive variety of information to be passed across the domain barrier, such as those discussed here, would increase the scope for applying data science techniques and machine learning dramatically.

In addition, interest from other related areas, including generative hyper-heuristics and automated algorithm configuration/tuning, is likely to increase as such methods would benefit greatly from the improved feedback given during the search process. Another aspect that is often overlooked is the information obtained during the search process that is typically discarded immediately. For example, one advantage of the Hyperion[2] framework,

largely overlooked in the literature, is the possibility for analysis of the trace taken through the search space by a hyper-heuristic. This could be of help to algorithm designers. Indeed, many potential opportunities for utilising such *state features* to guide search methods exist, particularly in the context of areas such as *landscape-aware heuristic search*.

One of the limitations of current selection hyper-heuristic work not considered above is the lack of *delta* (incremental) evaluation. In the context of heuristic search, when a new solution is produced by performing a modification to an existing solution, delta evaluation is the concept of computing the objective value based only on the changes made, rather than fully re-evaluating the whole solution from scratch. This contributes somewhat to another challenge, dealing with the variety of time and space requirements of operators across different domains and sometimes between operators in the same domain. For example, under the CHeSC 2011 competition rules, limiting hyper-heuristics to 10 nominal minutes of CPU time, it is possible to perform many more low-level heuristic applications, searching many more states/solutions, in some domains than others. Providing the high-level hyper-heuristic with some indication of the computational effort required to invoke a particular low-level heuristic would go some way to rectifying this. Some existing hyper-heuristics have overcome this issue by empirically sampling low-level heuristics at the start of a search. However, this is not an ideal situation.

In this review paper, we have discussed a small number of papers which perform theoretical analysis of hyper-heuristic methods (e.g. Alanazi and Lehre (2014, 2016); Lehre and Özcan (2013)). The conclusions of these papers were very much in line with previous work in the area, showing that strategies that mix multiple operators are able to outperform *pure* strategies using only a single operator. However, some of the limitations of traditional learning mechanisms used in hyper-heuristics were exposed, particularly when the performance of individual low-level heuristics is similar. Further investigation into the theoretical underpinnings of selection hyper-heuristics in future will lead to improved understanding of both the expectations that can be placed on such methods, and the limitations that they operate within.

Although our focus here has been on selection hyper-heuristics, Section 8 briefly discussed some hyper-heuristics that generate selection hyper-heuristics or components of selection hyper-heuristics. Although the number of papers focusing on this area is relatively small, there is much scope for potential future work in this area. One of the barriers to developing generation hyper-heuristics to build selection hyper-heuristics is the sheer computational effort required to train and test such methods. This is something that is decreasingly problematic with the rise in parallel processing methods, driven by the wider availability of affordable general purpose GPUs. Another is the limited generality level currently achieved by generated selection hyper-heuristics and components. Many of the existing methods discussed previously generate *disposable* hyper-heuristics that are trained and evaluated on a per-instance basis, rather than *reusable* hyper-heuristics which are able to learn at a higher level. Whilst there are some methods able to learn at a per-domain level, trained on a subset of instances and then tested on another set drawn from the same distribution, there has been some difficulty in effectively achieving genuine cross-domain learning. Mısır (2017) raised some interesting and important questions about the nature of what is meant by cross-domain search, particularly with respect to how inter- and intra-domain learning are defined. This work identified clusters of similar problem instances, using matrix factorisation to identify hidden features, containing instances from multiple problem domains. Previous work has often approached cross-domain search under the assumption that problems of the same class are taken from the same distribution. However, there are clearly latent features of problem instances from which more nuanced, and potentially more useful, classifications could be derived.

Many researchers and practitioners across a variety of disciplines have been working towards the goal of building more general solvers, investigating various perspectives from theory to practice, exploring how high the level of generality of search algorithms can be raised. In particular, the number of studies on selection hyper-heuristics for automatically solving single and multiobjective optimisation problems has been growing rapidly in contemporary research. This paper provides a high-level snapshot of the research landscape in selection hyper-heuristics, focusing on recent progress made in the area.

Acknowledgments

This work has been partially funded by the DAASE project, EPSRC programme grant EP/J017515/1 and the OR-MASTER project, EPSRC programme grant EP/M020258/1.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on machine learning, ICML '04* (pp. 1–8). New York, NY, USA: ACM.

Adriaensen, S., Brys, T., & Nowé, A. (2014a). Designing reusable metaheuristic methods: A semi-automated approach. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2014)* (pp. 2969–2976). IEEE.

Adriaensen, S., Brys, T., & Nowé, A. (2014b). Fair-share ILS: A simple state-of-the-art iterated local search hyperheuristic. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation* (pp. 1303–1310). ACM.

Adriaensen, S., & Nowé, A. (2016). Case study: An analysis of accidental complexity in a state-of-the-art hyper-heuristic for hyflex. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2016)* (pp. 1485–1492). IEEE.

Adriaensen, S., Ochoa, G., & Nowé, A. (2015). A benchmark set extension and comparative study for the hyflex framework. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2015)* (pp. 784–791). IEEE.

Ahmed, L., Mumford, C., & Kheiri, A. (2019). Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research, 274*(2), 545–559.

Akar, E., Topcuoglu, H. R., & Ermis, M. (2014). Hyper-heuristics for online UAV path planning under imperfect information. In *Proceedings of the European conference on the applications of evolutionary computation* (pp. 741–752). Springer.

Alanazi, F., & Lehre, P. K. (2014). Runtime analysis of selection hyper-heuristics with classical learning mechanisms. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2014)* (pp. 2515–2523). IEEE.

Alanazi, F., & Lehre, P. K. (2016). Limits to learning in reinforcement learning hyper-heuristics. In *Proceedings of the European conference on evolutionary computation in combinatorial optimization (EVOCOP)*. In *Lecture Notes in Computer Science: 9595* (pp. 170–185).

Allen, J. G., Coates, G., & Trevelyan, J. (2013). A hyper-heuristic approach to aircraft structural design optimization. *Structural and Multidisciplinary Optimization, 48*(4), 807–819.

Almutairi, A., Özcan, E., Kheiri, A., & Jackson, W. G. (2016). Performance of selection hyper-heuristics on the extended hyflex domains. In *Proceedings of the International symposium on computer and information sciences* (pp. 154–162). Springer.

Aron, R., Chana, I., & Abraham, A. (2015). A hyper-heuristic approach for resource provisioning-based scheduling in grid environment. *The Journal of Supercomputing, 71*(4), 1427–1450.

Asta, S., Karapetyan, D., Kheiri, A., Özcan, E., & Parkes, A. J. (2016a). Combining Monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences, 373*, 476–498.

Asta, S., & Özcan, E. (2015). A tensor-based selection hyper-heuristic for cross-domain heuristic search. *Information Sciences, 299*, 412–432.

Asta, S., Özcan, E., & Curtois, T. (2016b). A tensor based hyper-heuristic for nurse rostering. *Knowledge-Based Systems, 98*, 185–199.

Asta, S., Özcan, E., & Parkes, A. J. (2013a). Batched mode hyper-heuristics. In *Proceedings of the international conference on learning and intelligent optimization (lion)*. In *Lecture Notes in Computer Science: 7997* (pp. 404–409).

Asta, S., Özcan, E., Parkes, A. J., & Etaner-Uyar, A. Ş. (2013b). Generalizing hyper-heuristics via apprenticeship learning. In *Proceedings of the European conference on evolutionary computation in combinatorial optimization (EVOCOP 2013)*. In *Lecture Notes in Computer Science: 7832* (pp. 169–178).

Bai, R., Van Woensel, T., Kendall, G., & Burke, E. K. (2013). A new model and a hyper-heuristic approach for two-dimensional shelf space allocation. *4OR, 11*(1), 31–55.

Battiti, R., & Brunato, M. (2017). *The LION way. Machine learning* plus *intelligent optimization*. Italy: LIONlab, University of Trento. http://intelligent-optimization.org/LIONbook/

Baykasoğlu, A., & Ozsoydan, F. B. (2017). Evolutionary and population-based methods versus constructive search strategies in dynamic combinatorial optimization. *Information Sciences, 420*, 159–183.

Bilgin, B., Demeester, P., Misir, M., Vancroonenburg, W., & Vanden Berghe, G. (2012). One hyper-heuristic approach to two timetabling problems in health care. *Journal of Heuristics, 18*(3), 401–434.

Birattari, M., Paquete, L., Stützle, T., & Varrentrapp, K. (2001). *Technical Report, Darmstadt, Germany*.

Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E., et al. (2003). PISA—A platform and programming language independent interface for search algorithms. In C. M. Fonseca, et al. (Eds.), *Proceedings of the conference on evolutionary multi-criterion optimization (EMO 2003)*. In *LNCS: 2632* (pp. 494–508). Berlin: Springer.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys, 35*(3), 268–308.

Branke, J., Nguyen, S., Pickardt, C. W., & Zhang, M. (2016). Automated design of production scheduling heuristics: a review. *IEEE Transactions on Evolutionary Computation, 20*(1), 110–124.

Brownlee, A. E., Swan, J., Özcan, E., & Parkes, A. J. (2014). Hyperion2: a toolkit for {meta-, hyper-} heuristic research. In *Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation* (pp. 1133–1140). ACM.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., & Schulenburg, S. (2003). *Hyper-heuristics: An emerging direction in modern search technology*. In F. Glover, & G. A. Kochenberger (Eds.) (pp. 457–474)). Boston, MA: Springer.

Burke, E. K., & Bykov, Y. (2017). The late acceptance hill-climbing heuristic. *European Journal of Operational Research, 258*(1), 70–78.

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

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In *Handbook of metaheuristics* (pp. 449–468). Springer.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (2009). Exploring hyper-heuristic methodologies with genetic programming. In *Computational intelligence* (pp. 177–201). Springer.

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

Burke, E. K., Qu, R., & Soghier, A. (2014). Adaptive selection of heuristics for improving exam timetables. *Annals of Operations Research, 218*(1), 129–145.

Caraffini, F., Neri, F., & Epitropakis, M. (2019). Hyperspam: A study on hyper-heuristic coordination strategies in the continuous domain. *Information Sciences, 477*, 186–202.

de Carvalho, V. R., & Sichman, J. S. (2017). Applying copeland voting to design an agent-based hyper-heuristic. In *Proceedings of the 16th conference on autonomous agents and multiagent systems* (pp. 972–980). International Foundation for Autonomous Agents and Multiagent Systems.

Castro Jr, O. R., & Pozo, A. (2015). Using hyper-heuristic to select leader and archiving methods for many-objective problems. In *Proceedings of the international conference on evolutionary multi-criterion optimization* (pp. 109–123). Springer.

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

Chen, S., Li, Z., Yang, B., & Rudolph, G. (2016a). Quantum-inspired hyper-heuristics for energy-aware scheduling on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems, 27*(6), 1796–1810.

Chen, Y., Cowling, P., Polack, F., Remde, S., & Mourdjis, P. (2017). Dynamic optimisation of preventative and corrective maintenance schedules for a large scale urban drainage system. *European Journal of Operational Research, 257*(2), 494–510.

Chen, Y., Mourdjis, P., Polack, F., Cowling, P., & Remde, S. (2016b). Evaluating hyperheuristics and local search operators for periodic routing problems. In *Proceedings of the European conference on evolutionary computation in combinatorial optimization*. In *Lecture Notes in Computer Science: 9595* (pp. 104–120). Springer.

Choong, S. S., Wong, L.-P., & Lim, C. P. (2017). An artificial bee colony algorithm with a modified choice function for the traveling salesman problem. In *Proceedings of the IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 357–362). IEEE.

Choong, S. S., Wong, L.-P., & Lim, C. P. (2018). Automatic design of hyper-heuristic based on reinforcement learning. *Information Sciences, 436-437*, 89–107.

Chuang, C.-Y., & Smith, S. F. (2017). A study of agnostic hyper-heuristics based on sampling solution chains. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 271–278). IEEE.

Cichowicz, T., Drozdowski, M., Frankiewicz, M., Pawlak, G., Rytwiński, F., & Wasilewski, J. (2012). Five phase and genetic hive hyper-heuristics for the cross–domain search. In Y. Hamadi, & M. Schoenauer (Eds.), *Learning and intelligent optimization*. In *Lecture Notes in Computer Science* (pp. 354–359). Berlin Heidelberg: Springer.

Cora, H. K., Uyar, H. T., & Etaner-Uyar, A. Ş. (2013). Hh-dsl: a domain specific language for selection hyper-heuristics. In *Proceedings of the 15th annual conference companion on genetic and evolutionary computation* (pp. 1317–1324). ACM.

Cowling, P., Kendall, G., & Soubeiga, E. (2001). A hyperheuristic approach to scheduling a sales summit. In E. Burke, & W. Erben (Eds.), *Practice and theory of automated timetabling III*. In *Lecture Notes in Computer Science: 2079* (pp. 176–190). Berlin Heidelberg: Springer.

Crawford, B., Soto, R., Monfroy, E., Palma, W., Castro, C., & Paredes, F. (2013). Parameter tuning of a choice-function based hyperheuristic using particle swarm optimization. *Expert Systems with Applications, 40*(5), 1690–1695.

Curtois, T., Ochoa, G., Hyde, M., & Vázquez-Rodríguez, J. A. (2010). *Technical Report*.

Damaševičius, R., & Woźniak, M. (2017). State flipping based hyper-heuristic for hybridization of nature inspired algorithms. In *Proceedings of the international conference on artificial intelligence and soft computing* (pp. 337–346). Springer.

Di Gaspero, L., & Urli, T. (2012). Evaluation of a family of reinforcement learning cross-domain optimization heuristics. In Y. Hamadi, & M. Schoenauer (Eds.), *Learning and intelligent optimization*. In *Lecture Notes in Computer Science* (pp. 384–389). Berlin Heidelberg: Springer.

Drake, J. H. (2014). *Crossover control in selection hyper-heuristics : case studies using MKP and HyFlex*. University of Nottingham, UK (Ph.D. thesis). (Chapter 7)

Drake, J. H., Özcan, E., & Burke, E. K. (2012). An improved choice function heuristic selection for cross domain heuristic search. In *Proceedings of the international conference on parallel problem solving from nature (PPSN 2012)* (pp. 307–316). Springer.

Drake, J. H., Özcan, E., & Burke, E. K. (2016). A case study of controlling crossover in a selection hyper-heuristic framework using the multidimensional knapsack problem. *Evolutionary Computation, 24*(1), 113–141.

Durillo, J. J., & Nebro, A. J. (2011). jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software, 42*, 760–771.

Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation, 3*(2), 124–141.

El Kateb, D., Fouquet, F., Bourcier, J., & Le Traon, Y. (2014). Optimizing multi-objective evolutionary algorithms to enable quality-aware software provisioning. In *Proceedings of the 14th international conference on Quality software (QSIC), 2014* (pp. 85–94). IEEE.

El Yafrani, M., Martins, M., Wagner, M., Ahiod, B., Delgado, M., & Lüders, R. (2018). A hyperheuristic approach based on low-level heuristics for the travelling thief problem. *Genetic Programming and Evolvable Machines, 19*(1-2), 121–150.

Elhag, A., & Özcan, E. (2015). A grouping hyper-heuristic framework: Application on graph colouring. *Expert Systems with Applications, 42*(13), 5491–5507.

Ferreira, A. S., Gonçalves, R. A., & Pozo, A. (2017). A multi-armed bandit selection strategy for hyper-heuristics. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 525–532). IEEE.

Fialho, Á., Da Costa, L., Schoenauer, M., & Sebag, M. (2008). Extreme value based adaptive operator selection. In G. Rudolph, T. Jansen, N. Beume, S. Lucas, & C. Poloni (Eds.), *Parallel problem solving from nature – PPSN x* (pp. 175–184). Berlin, Heidelberg: Springer.

Fisher, H., & Thompson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. In J. F. Muth, & G. L. Thompson (Eds.), *Industrial scheduling* (pp. 225–251). New Jersey: Prentice-Hall, Inc.

da Fonseca, G. H. G., Santos, H. G., Toffolo, T. Â. M., Brito, S. S., & Souza, M. J. F. (2016). Goal solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research, 239*(1), 77–97.

Fontoura, V. D., Pozo, A. T., & Santana, R. (2017). Automated design of hyper-heuristics components to solve the PSP problem with HP model. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 1848–1855). IEEE.

Gascón-Moreno, J., Salcedo-Sanz, S., Saavedra-Moreno, B., Carro-Calvo, L., & Portilla–Figueras, A. (2013). An evolutionary-based hyper-heuristic approach for optimal construction of group method of data handling networks. *Information Sciences, 247*, 94–108.

Gomez, J. C., & Terashima-Marín, H. (2018). Evolutionary hyper-heuristics for tackling bi-objective 2D bin packing problems. *Genetic Programming and Evolvable Machines, 19*(1-2), 151–181.

Gómez, R. H., & Coello, C. A. C. (2017). A hyper-heuristic of scalarizing functions. In *Proceedings of the genetic and evolutionary computation conference* (pp. 577–584). ACM.

Gonçalves, R. A., Kuk, J. N., Almeida, C. P., & Venske, S. M. (2015). MOEA/D-HH: A hyper-heuristic for multi-objective problems. In *Proceedings of the International conference on evolutionary multi-criterion optimization* (pp. 94–108). Springer.

Grobler, J., & Engelbrecht, A. P. (2016). Hyper-heuristics for the flexible job shop scheduling problem with additional constraints. In *Proceedings of the International conference in swarm intelligence* (pp. 3–10). Springer.

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

Guizzo, G., Bazargani, M., Paixao, M., & Drake, J. H. (2017a). A hyper-heuristic for multi-objective integration and test ordering in google guava. In *Proceedings of the International symposium on search based software engineering* (pp. 168–174). Springer.

Guizzo, G., Fritsche, G. M., Vergilio, S. R., & Pozo, A. T. R. (2015). A hyper-heuristic for the multi-objective integration and test order problem. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation* (pp. 1343–1350). ACM.

Guizzo, G., Vergilio, S. R., Pozo, A. T., & Fritsche, G. M. (2017b). A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem. *Applied Soft Computing, 56*, 331–344.

Gümüş, D. B., Ozcan, E., & Atkin, J. (2016). An investigation of tuning a memetic algorithm for cross-domain search. In *IEEE congress on evolutionary computation (CEC 2016)* (pp. 135–142). IEEE.

Gutierrez-Rodríguez, A. E., Ortiz-Bayliss, J. C., Rosales-Pérez, A., Amaya–Contreras, I. M., Conant-Pablos, S. E., Terashima-Marín, H., & Coello, C. A. C. (2017). Applying automatic heuristic-filtering to improve hyper-heuristic performance. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 2638–2644). IEEE.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighbourhood search: Methods and applications. *Annals of Operations Research, 175*(1), 367–407.

Henard, C., Papadakis, M., & Le Traon, Y. (2014). Mutation-based generation of software product line test configurations. In *Proceedings of the International symposium on search based software engineering* (pp. 92–106). Springer.

Hitomi, N., & Selva, D. (2016). A hyperheuristic approach to leveraging domain knowledge in multi-objective evolutionary algorithms. In *Proceedings of the ASME 2016 international design engineering technical conferences and computers and information in engineering conference*. American Society of Mechanical Engineers. V02BT03A030–V02BT03A030

Hsiao, P.-C., Chiang, T.-C., & Fu, L.-C. (2012). A vns-based hyper-heuristic with adaptive computational budget of local search. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2012)* (pp. 1–8). IEEE.

Hyde, M., Ochoa, G., Curtois, T., & Vázquez-Rodríguez, J. (2010a). *Technical Report*.

Hyde, M., Ochoa, G., Vázquez-Rodríguez, J. A., & Curtois, T. (2010b). A HyFlex module for the MAX-SAT problem. *Technical Report*.

Jackson, W., Özcan, E., & John, R. I. (2018). Move acceptance in local search metaheuristics for cross-domain search. *Expert Systems with Applications, 109*, 131–151.

Jackson, W. G., Özcan, E., & Drake, J. H. (2013). Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. In *Proceedings of the 2013 13th UK workshop on computational intelligence (UKCI)* (pp. 228–235). IEEE.

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

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

Kampouridis, M., Alsheddy, A., & Tsang, E. (2013). On the investigation of hyper-heuristics on a financial forecasting problem. *Annals of Mathematics and Artificial Intelligence, 68*(4), 225–246.

Karapetyan, D., Punnen, A. P., & Parkes, A. J. (2017). Markov chain methods for the bipartite boolean quadratic programming problem. *European Journal of Operational Research, 260*(2), 494–506.

Kendall, G., & Li, J. (2013). Competitive travelling salesmen problem: A hyper-heuristic approach. *Journal of the Operational Research Society, 64*(2), 208–216.

Kheiri, A., & Keedwell, E. (2015). A sequence-based selection hyper-heuristic utilising a hidden Markov model. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation* (pp. 417–424). ACM.

Kheiri, A., & Keedwell, E. (2017). A hidden Markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems. *Evolutionary Computation, 25*(3), 473–501.

Kheiri, A., & Özcan, E. (2016). An iterated multi-stage selection hyper-heuristic. *European Journal of Operational Research, 250*(1), 77–90.

Kheiri, A., Özcan, E., & Parkes, A. J. (2016). A stochastic local search algorithm with adaptive acceptance for high-school timetabling. *Annals of Operations Research, 239*(1), 135–151.

Kiraz, B., Etaner-Uyar, A., & Özcan, E. (2013). Selection hyper-heuristics in dynamic environments. *Journal of the Operational Research Society, 64*(12), 1753–1769.

Kotthoff, L. (2014). Algorithm selection for combinatorial search problems: A survey. *AI Magazine, 35*(3), 48–60.

Koulinas, G., & Anagnostopoulos, K. (2013). A new tabu search-based hyper-heuristic algorithm for solving construction leveling problems with limited resource availabilities. *Automation in Construction, 31*, 169–175.

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences, 277*, 680–693.

Kubalík, J. (2012). Hyper-heuristic based on iterated local search driven by evolutionary algorithm. In J.-K. Hao, & M. Middendorf (Eds.), *Evolutionary computation in combinatorial optimization*. In *Lecture Notes in Computer Science: 7245* (pp. 148–159). Berlin Heidelberg: Springer.

Kumari, A. C., & Srinivas, K. (2016). Hyper-heuristic approach for multi-objective software module clustering. *Journal of Systems and Software, 117*, 384–401.

Lassouaoui, M., & Boughaci, D. (2014). A choice function hyper-heuristic for the winner determination problem. In *Proceedings of the nature inspired cooperative strategies for optimization (NICSO 2013)* (pp. 303–314). Springer.

Lehrbaum, A., & Musliu, N. (2012). A new hyperheuristic algorithm for cross-domain search problems. In Y. Hamadi, & M. Schoenauer (Eds.), *Proceedings of the learning and intelligent optimization*. In *Lecture Notes in Computer Science* (pp. 437–442). Berlin Heidelberg: Springer.

Lehre, P. K., & Özcan, E. (2013). A runtime analysis of simple hyper-heuristics: To mix or not to mix operators. In *Proceedings of the twelfth workshop on foundations of genetic algorithms XII* (pp. 97–104). ACM.

Li, D., Li, M., Meng, X., & Tian, Y. (2015). A hyperheuristic approach for intercell scheduling with single processing machines and batch processing machines. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45*(2), 315–325.

Li, D., Zhan, R., Zheng, D., Li, M., & Kaku, I. (2016). A hybrid evolutionary hyper-heuristic approach for intercell scheduling considering transportation capacity. *IEEE Transactions on Automation Science and Engineering, 13*(2), 1072–1089.

Li, J., & Kendall, G. (2017). A hyperheuristic methodology to generate adaptive strategies for games. *IEEE Transactions on Computational Intelligence and AI in Games, 9*(1), 1–10.

Li, K., Fialho, Kwong, S., & Zhang, Q. (2014). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 18*(1), 114–130.

Li, W., Özcan, E., & John, R. (2017). Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation. *Renewable Energy, 105*, 473–482.

Lin, J., Wang, Z.-J., & Li, X. (2017). A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm and Evolutionary Computation, 36*, 124–135.

López-Camacho, E., Terashima-Marin, H., Ross, P., & Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications, 41*(15), 6876–6889.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives, 3*, 43–58.

López-Ibáñez, M., & Stützle, T. (2014). Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research, 235*(3), 569–582.

Maashi, M., Kendall, G., & Özcan, E. (2015). Choice function based hyper-heuristics for multi-objective optimization. *Applied Soft Computing, 28*, 312–326.

Maashi, M., Özcan, E., & Kendall, G. (2014). A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications, 41*(9), 4475–4493.

Marshall, R. J., Johnston, M., & Zhang, M. (2015). Hyper-heuristic operator selection and acceptance criteria. In *Proceedings of the European conference on evolutionary computation in combinatorial optimization (EVOCOP)*. In *Lecture Notes in Computer Science: 9026* (pp. 99–113). Springer.

Martello, S., Pisinger, D., & Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science, 45*(3), 414–424.

Martins, M. S., El Yafrani, M., Delgado, M. R., Wagner, M., Ahiod, B., & Lüders, R. (2017). HSEDA: A heuristic selection approach based on estimation of distribution algorithm for the travelling thief problem. In *Proceedings of the genetic and evolutionary computation conference, (GECCO 2017)* (pp. 361–368). ACM.

Mascia, F., & Stützle, T. (2012). A non-adaptive stochastic local search algorithm for the CHeSC 2011 competition. In Y. Hamadi, & M. Schoenauer (Eds.), *Learning and intelligent optimization*. In *Lecture Notes in Computer Science* (pp. 101–114). Berlin Heidelberg: Springer.

McClymont, K., Keedwell, E., Savić, D., & Randall-Smith, M. (2013). A general multi-objective hyper-heuristic for water distribution network design with discolouration risk. *Journal of Hydroinformatics, 15*(3), 700–716.

Meignan, D. (2011). An evolutionary programming hyper-heuristic with co-evolution for CHeSC'11. In *Proceedings of the 53rd annual conference of the UK operational research society (OR53)*.

Meignan, D., Schwarze, S., & Voß, S. (2016). Improving local-search metaheuristics through look-ahead policies. *Annals of Mathematics and Artificial Intelligence, 76*(1-2), 59–82.

Mısır, M. (2017). Matrix factorization based benchmark set analysis: A case study on hyflex. In *Proceedings of the Asia-pacific conference on simulated evolution and learning (SEAL 2017)* (pp. 184–195). Springer.

Mısır, M., Smet, P., & Vanden Berghe, G. (2015). An analysis of generalised heuristics for vehicle routing and personnel rostering problems. *Journal of the Operational Research Society, 66*(5), 858–870.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2012a). The effect of the set of low-level heuristics on the performance of selection hyper-heuristics. In *Proceedings of the international conference on parallel problem solving from nature* (pp. 408–417). Springer.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2012b). An intelligent hyper-heuristic framework for CHESC 2011. In *Proceedings of the learning and intelligent optimization* (pp. 461–466). Springer.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2013a). An investigation on the generality level of selection hyper-heuristics under different empirical conditions. *Applied Soft Computing, 13*(7), 3335–3353.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2013b). A new hyper-heuristic as a general problem solver: an implementation in hyflex. *Journal of Scheduling, 16*(3), 291–311.

Monemi, R. N., Danach, K., Khalil, W., Gelareh, S., Lima, F. C., & Aloise, D. J. (2015). Solution methods for scheduling of heterogeneous parallel machines applied to the workover rig problem. *Expert Systems with Applications, 42*(9), 4493–4505.

Mourdjis, P., Chen, Y., Polack, F., Cowling, P., & Robinson, M. (2016). Variable neighbourhood descent with memory: A hybrid metaheuristic for supermarket resupply. In *Proceedings of the international workshop on hybrid metaheuristics* (pp. 32–46). Springer.

Muklason, A., Parkes, A. J., Özcan, E., McCollum, B., & McMullan, P. (2017). Fairness in examination timetabling: Student preferences and extended formulations. *Applied Soft Computing, 55*, 302–318.

Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J. A., Walker, J., Gendreau, M., … Burke, E. K. (2012). HyFlex: a benchmark framework for cross-domain heuristic search. In J.-K. Hao, & M. Middendorf (Eds.), *Evolutionary computation in combinatorial optimization*. In *Lecture Notes in Computer Science: 7245* (pp. 136–147). Berlin Heidelberg: Springer.

Ong, Y.-S., Lim, M.-H., Zhu, N., & Wong, K.-W. (2006). Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36*(1), 141–152. doi:10.1109/TSMCB.2005.856143.

Ortiz-Bayliss, J. C., Terashima-Marín, H., & Conant-Pablos, S. E. (2013). Learning vector quantization for variable ordering in constraint satisfaction problems. *Pattern Recognition Letters, 34*(4), 423–432.

Ortiz-Bayliss, J. C., Terashima-Marín, H., & Conant-Pablos, S. E. (2016). Combine and conquer: An evolutionary hyper-heuristic approach for solving constraint satisfaction problems. *Artificial Intelligence Review, 46*(3), 327–349.

Özcan, E., Bilgin, B., & Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis, 12*(1), 3–23.

Pappa, G. L., Ochoa, G., Hyde, M. R., Freitas, A. A., Woodward, J., & Swan, J. (2014). Contrasting meta-learning and hyper-heuristic research: The role of evolutionary algorithms. *Genetic Programming and Evolvable Machines, 15*(1), 3–35.

Parejo, J. A., Ruiz-Cortés, A., Lozano, S., & Fernandez, P. (2012). Metaheuristic optimization frameworks: a survey and benchmarking. *Soft Computing, 16*(3), 527–561.

Parkes, A. J., Özcan, E., & Karapetyan, D. (2015). A software interface for supporting the application of data science to optimisation. In *Proceedings of the international conference on learning and intelligent optimization* (pp. 306–311). Springer.

Pillay, N. (2012). A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem. *South African Computer Journal, 48*(1), 31–40.

Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. *Annals of Operations Research, 239*(1), 3–38.

Pillay, N., & Beckedahl, D. (2017). EvoHyp - a java toolkit for evolutionary algorithm hyper-heuristics. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 2706–2713). IEEE.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research, 34*(8), 2403–2435.

Pour, S. M., Drake, J. H., & Burke, E. K. (2018). A choice function hyper-heuristic framework for the allocation of maintenance tasks in danish railways. *Computers & Operations Research, 93*, 15–26.

Qu, R., Pham, N., Bai, R., & Kendall, G. (2015). Hybridising heuristics within an estimation distribution algorithm for examination timetabling. *Applied Intelligence, 42*(4), 679–693.

Rahimian, E., Akartunalı, K., & Levine, J. (2017). A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems. *European Journal of Operational Research, 258*(2), 411–423.

Raidl, G. R. (2015). Decomposition based hybrid metaheuristics. *European Journal of Operational Research, 244*(1), 66–76.

Rajni, & Chana, I. (2013). Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. *Future Generation Computer Systems, 29*(3), 751–762.

Rosales-Pérez, A., Gutiérrez-Rodríguez, A. E., Ortiz-Bayliss, J. C., Terashima-Marín, H., & Coello, C. A. C. (2017). Evolutionary multilabel hyper-heuristic design. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 2622–2629). IEEE.

Ross, P. (2014). Hyper-heuristics. In E. K. Burke, & G. Kendall (Eds.), *Search methodologies: Introductory tutorials in optimization and decision support techniques* (pp. 611–638)). Springer US.

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

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2015a). Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation, 19*(3), 309–325.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2015b). A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics, 45*(2), 217–228.

Sabar, N. R., & Kendall, G. (2015). Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems. *Information Sciences, 314*, 225–239.

Sabar, N. R., Turky, A., Song, A., & Sattar, A. (2017). Optimising deep belief networks by hyper-heuristic approach. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 2738–2745). IEEE.

Sabar, N. R., Zhang, X. J., & Song, A. (2015c). A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2015)* (pp. 830–837). IEEE.

Salcedo-Sanz, S., Matías-Román, J., Jiménez-Fernández, S., Portilla-Figueras, A., & Cuadra, L. (2014). An evolutionary-based hyper-heuristic approach for the jawbreaker puzzle. *Applied Intelligence, 40*(3), 404–414.

Salhi, A., & Rodríguez, J. A. V. (2014). Tailoring hyper-heuristics to specific instances of a scheduling problem using affinity and competence functions. *Memetic Computing, 6*(2), 77–84.

Segredo, E., Segura, C., & León, C. (2014). Memetic algorithms and hyperheuristics applied to a multiobjectivised two-dimensional packing problem. *Journal of Global Optimization, 58*(4), 769–794.

Segura, C., Segredo, E., & León, C. (2012). Analysing the adaptation level of parallel hyperheuristics applied to multiobjectivised benchmark problems. In *Proceedings of the 2012 20th Euromicro international conference on parallel, distributed and network-based processing* (pp. 138–145). IEEE.

Sim, K., & Hart, E. (2016). A combined generative and selective hyper-heuristic for the vehicle routing problem. In *Proceedings of the 2016 on genetic and evolutionary computation conference* (pp. 1093–1100). ACM.

Smith, S. L., & Imeson, F. (2017). GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research, 87*, 1–19.

Soghier, A., & Qu, R. (2013). Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. *Applied Intelligence, 39*(2), 438–450.

Soria-Alcaraz, J. A., Ochoa, G., Carpio, M., & Puga, H. (2014a). Evolvability metrics in adaptive operator selection. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation* (pp. 1327–1334). ACM.

Soria-Alcaraz, J. A., Ochoa, G., Sotelo-Figeroa, M. A., & Burke, E. K. (2017). A methodology for determining an effective subset of heuristics in selection hyper-heuristics. *European Journal of Operational Research, 260*(3), 972–983.

Soria-Alcaraz, J. A., Ochoa, G., Swan, J., Carpio, M., Puga, H., & Burke, E. K. (2014b). Effective learning hyper-heuristics for the course timetabling problem. *European Journal of Operational Research, 238*(1), 77–86.

van der Stockt, S., & Engelbrecht, A. P. (2014). Analysis of hyper-heuristic performance in different dynamic environments. In *Proceedings of the IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE), 2014* (pp. 1–8). IEEE.

Strickler, A., Lima, J. A. P., Vergilio, S. R., & Pozo, A. T. (2016). Deriving products for variability test of feature models with a hyper-heuristic approach. *Applied Soft Computing, 49*, 1232–1242.

Swan, J., De Causmaecker, P., Martin, S., & Özcan, E. (2018). A re-characterization of hyper-heuristics. In *Recent developments in metaheuristics* (pp. 75–89). Springer.

Swan, J., Özcan, E., & Kendall, G. (2011). Hyperion- a recursive hyper-heuristic framework. In C. A. C. Coello (Ed.), *Proceedings of the fifth international conference on learning and intelligent optimization: (LION 5)*. In *Lecture Notes in Computer Science: 6683* (pp. 616–630). Berlin Heidelberg: Springer.

Swiercz, A., Burke, E. K., Cichenski, M., Pawlak, G., Petrovic, S., Zurkowski, T., & Blazewicz, J. (2014). Unified encoding for hyper-heuristics with application to bioinformatics. *Central European Journal of Operations Research, 22*(3), 567–589.

Thomas, J., & Chaudhari, N. S. (2014). Design of efficient packing system using genetic algorithm based on hyper heuristic approach. *Advances in Engineering Software, 73*, 45–52.

Tinoco, J. C. V., & Coello, C. A. C. (2013). hypDE: A hyper-heuristic based on differential evolution for solving constrained optimization problems. In *Evolve-a bridge between probability, set oriented numerics, and evolutionary computation II* (pp. 267–282). Springer.

Topcuoglu, H. R., Ucar, A., & Altin, L. (2014). A hyper-heuristic based framework for dynamic optimization problems. *Applied Soft Computing, 19*, 236–251.

Tsai, C.-W., Chang, W.-L., Hu, K.-C., & Chiang, M.-C. (2017). An improved hyper-heuristic clustering algorithm for wireless sensor networks. *Mobile Networks and Applications, 22*, 943–958.

Tsai, C.-W., Huang, W.-C., Chiang, M.-H., Chiang, M.-C., & Yang, C.-S. (2014). A hyper-heuristic scheduling algorithm for cloud. *IEEE Transactions on Cloud Computing, 2*(2), 236–250.

Tyasnurita, R., Özcan, E., & John, R. (2017). Learning heuristic selection using a time delay neural network for open vehicle routing. In *Proceedings of the IEEE congress on evolutionary computation (CEC 2017)* (pp. 1474–1481). IEEE.

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

Urra, E., Cabrera-Paniagua, D., & Cubillos, C. (2013). Towards an object-oriented pattern proposal for heuristic structures of diverse abstraction levels. *Proceedings of the In XXI Jornadas Chilenas de Computación 2013, Temuco, Chile*.

Van Onsem, W., & Demoen, B. (2013). ParHyFlex: A framework for parallel hyper-heuristics. In *Proceedings of the 25th Benelux conference on articial intelligence, Benelux conference on artificial intelligence, Delft, 7–8 november 2013: 28* (pp. 231–238).

Vazquez-Rodriguez, J., & Petrovic, S. (2013). A mixture experiments multi-objective hyper-heuristic. *Journal of the Operational Research Society, 64*(11), 1664–1675.

Vázquez-Rodrıguez, J. A., Ochoa, G., Curtois, T., & Hyde, M. (2010). A hyflex module for the permutation flow shop problem. *Technical Report*.

Vrugt, J. A., & Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search, *104*(3), 708–711.

Walker, D. J., & Keedwell, E. (2016). Towards many-objective optimisation with hyper-heuristics: Identifying good heuristics with indicators. In *Proceedings of the international conference on parallel problem solving from nature* (pp. 493–502). Springer.

Wauters, T., Vancroonenburg, W., & Vanden Berghe, G. (2012). A guide-and-observe hyper-heuristic approach to the eternity II puzzle. *Journal of Mathematical Modelling and Algorithms, 11*(3), 217–233.

Wu, X., Consoli, P., Minku, L., Ochoa, G., & Yao, X. (2016). An evolutionary hyper-heuristic for the software project scheduling problem. In *International conference on parallel problem solving from nature (PPSN 2016)*. In *Lecture Notes in Computer Science: 9921* (pp. 37–47). Springer.

Yang, C., Peng, S., Jiang, B., Wang, L., & Li, R. (2014). Hyper-heuristic genetic algorithm for solving frequency assignment problem in TD-SCDMA. In *Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation* (pp. 1231–1238). ACM.

Yates, W., & Keedwell, E. (2017). Offline learning for selection hyper-heuristics with Elman networks. In *Proceedings of the international conference on artificial evolution (EA-2017)*.

Yin, P.-Y., Lyu, S.-R., & Chuang, Y.-L. (2016). Cooperative coevolutionary approach for integrated vehicle routing and scheduling using cross-dock buffering. *Engineering Applications of Artificial Intelligence, 52*, 40–53.

Zamli, K. Z., Alkazemi, B. Y., & Kendall, G. (2016). A tabu search hyper-heuristic strategy for t-way test suite generation. *Applied Soft Computing, 44*, 57–74.

Zhao, J., Zhou, Y.-W., & Wahab, M. (2016). Joint optimization models for shelf display and inventory control considering the impact of spatial relationship on demand. *European Journal of Operational Research, 255*(3), 797–808.

Zheng, Y.-J., Zhang, M.-X., Ling, H.-F., & Chen, S.-Y. (2015). Emergency railway transportation planning using a hyper-heuristic approach. *IEEE Transactions on Intelligent Transportation Systems, 16*(1), 321–329.