Hyper-heuristics

Michael G. Epitropakis and Edmund K. Burke

Contents

M. G. Epitropakis (✉)
Data Science Institute, Department of Management Science, Lancaster University Management School, Lancaster University, Lancaster, UK
e-mail: m.epitropakis@lancaster.ac.uk

E. K. Burke
School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK
e-mail: vp-se@qmul.ac.uk

R. Martí et al. (eds.), *Handbook of Heuristics*,
https://doi.org/10.1007/978-3-319-07153-4_32-1

Abstract

This chapter presents a literature review of the main advances in the field of hyper-heuristics, since the publication of a survey paper in 2013. The chapter demonstrates the most recent advances in hyper-heuristic foundations, methodologies, theory, and application areas. In addition, a simple illustrative selection hyper-heuristic framework is developed as a case study. This is based on the well-known Iterated Local Search algorithm and is presented to provide a tutorial style introduction to some of the key basic issues. A brief discussion about the implementation process in addition to the decisions that had to be made during the implementation is presented. The framework implements an action selection model that operates on the perturbation stage of the Iterated Local Search algorithm to adaptively select among various low-level perturbation heuristics. The performance and efficiency of the developed framework is evaluated across six well-known real-world problem domains.

Keywords

Hyper-heuristics · Heuristics · Meta-heuristics · Evolutionary computation · Optimization · Search · Machine learning · Multi-objective optimization · Combinatorial optimization · Black box optimization · Dynamic optimization · Scheduling · Timetabling · Packing · Iterated local search

Introduction

Over the last 50 years or so, heuristic search methodologies have been successfully applied to address various real-world complex optimization problems. Computationally challenging optimization problems arise in various disciplines such as operational research, computer science, finance, bioinformatics, industrial informatics, engineering, and data sciences. Representative examples of optimization problems can be found in scheduling, timetabling, planning, resource and space allocation, cutting and packing, software design, hardware design, and engineering design problems. Such optimization problems can be addressed by exact or heuristic methodologies (or a hybridization). The main difficulty of solving such problems occurs from the often remarkably large and possibly heavily constrained search space. In such problems, or in noisy and dynamic real-world problem scenarios, finding optimal solutions by exact methods might be impossible. Therefore, in practice, heuristic search methodologies are often developed or applied that seek to find solutions of acceptable quality in reasonable time, but without having any guarantee of optimality.

Over the last few decades, a broad spectrum of different heuristics have been proposed and successfully applied to a wide variety of problems. However, heuristics are usually designed to solve specific optimization problems. Such heuristics are bespoke problem-specific methods that often demand significant time to design and tune. They are also often expensive to implement and maintain. Hyper-heuristics

have been motivated by the aim of raising the level of generality at which search algorithms can operate, as well as automating the design and tuning of heuristic algorithms [32–34,36,38,102,106,107]. The main characteristic of a hyper-heuristic method is that it operates on a space of heuristics rather than on a space of solutions. As such, a hyper-heuristic methodology attempts to find or generate the appropriate method or sequence of low-level heuristics instead of directly solving a particular problem. A hyper-heuristic can be characterized as a high-level methodology which automatically generates or combines low-level search components (heuristics) to effectively solve a given problem instance or class of problems.

The following definition has been provided in the literature:

> A hyper-heuristic is a search method or learning mechanism for selecting or generating heuristics to solve computational search problems [34,36].

The use of the term of *hyper-heuristics* goes back to 2001 [40]. However, the idea of automating the design and/or selection of heuristics has existed in the literature for more than 50 years [41, 52]. It is possible to characterize hyper-heuristic methodologies by considering two dimensions [34]: the characteristics of the heuristics' search space and the feedback information from the search process. The heuristic search space can be further divided into heuristic selection and heuristic generation. The first category represents methodologies or models that select among a set of available heuristics, while the latter category covers methodologies that generate new heuristics from existing heuristic components. A further refinement in both categories can be made based on the characteristics of the utilized search heuristic, i.e., constructive and perturbative search paradigms [60].

Hyper-heuristics can be further distinguished by looking at how they handle feedback information from the search process. Two main categories can be currently observed: methodologies that learn from the feedback and the ones that do not learn from it. The first category can be further refined based on the source of learning: *online* or *off-line* learning. When considering online learning hyper-heuristics, the learning phase takes place during the search procedure. Thus, timely feedback information can be exploited by the learning mechanism to study the trends and behavior of the underlying components. In the case of off-line learning (off-line learning hyper-heuristics), all information about the process is known a priori. As such, the learning mechanism is able to gather knowledge and learn from a training set of problem instances and then apply its knowledge on a test set of unseen instances. The aim of such a procedure is to generalize from the training information to unseen problem instances in an expected way. Methodologies that do not learn from the feedback information usually simply discard it.

The aforementioned classification reflects the majority of the past and the current research trends that can be widely found in the literature. Nevertheless, interesting example methodologies can be found that represent an interplay between the categories of heuristics, i.e., combinations of selection and generation heuristic search procedures [112,122].

Over the last few years, various introductory tutorials, review, and survey articles have been published on the area of hyper-heuristics. The first book chapter that

appeared in 2003 [32] introduces the general idea of hyper-heuristics and provides a brief historical overview of the field at that time. In particular, the chapter focuses on one of the main objectives of hyper-heuristics, which is to increase the level of generality at which a search methodology can operate. The overview demonstrates the history as well as the developments and trends in the field at that time, through detailed descriptions of the representative applications and methodologies.

An introductory tutorial of the area was presented in [106] that effectively demonstrated the development process of a hyper-heuristic approach, through useful guidelines and characteristic examples. It additionally pinpoints potential application areas and discusses some issues and possible future research pointers in the area. A second-edition tutorial with enriched material and information was recently published in [107].

A classification of the field and a discussion of developments at that time were published in 2008 [38]. The chapter emphasizes the real-world applications that have been tackled by hyper-heuristic approaches and presents three criteria that characterize a hyper-heuristic methodology: operating at a higher level and managing low-level heuristics, searching the heuristic space rather than the solution space, and limited access to problem domain information, i.e., an increased level of generality. The authors identified the last criterion as a particularly important one for the general applicability and robustness of a hyper-heuristic approach.

An overview chapter was published in 2009 [33] that discusses hyper-heuristic methodologies that are able to generate new effective heuristics that are not known beforehand. The chapter addresses the concept of automatically designing heuristics and describes a methodology to generate new heuristics. Several representative case study examples are also provided. In addition, the presented methodology emphasizes the main algorithmic component of this class which is the genetic programming algorithm. Moreover, it briefly covers the related literature and discusses the characteristics and issues of this class of approaches.

A unified classification and definition of the field were presented in 2010 [34]. The chapter provides an overview of the previous categorizations of the area and determines a unified classification and definition of hyper-heuristics that captures the current and potential future directions of hyper-heuristics. Two main classes of hyper-heuristic approaches are recognized in the categorization, heuristic selection and generation approaches. Several characteristic examples are presented and discussed for both categories.

A recent survey of hyper-heuristics was published in 2013 [36]. The article presents the state of the art of the field up to 2012 by providing examples of the main and most representative methodologies and application areas. It also discusses the origins and intellectual roots of the field as well as provides a critical discussion of the literature. Various potential research trends are briefly discussed, and several future directions are highlighted. Finally, the most recent overview of hyper-heuristics was published in 2014 [102]. The article presents an overview and critical analysis of hyper-heuristics specifically for educational timetabling problems.

Building upon the aforementioned overview articles that have been recently published, this chapter aims to provide a thorough literature review of the main very recent advances in the field of hyper-heuristics, since the publication of [36] in 2013. An additional objective is to provide a case study for the development process of a simple but highly efficient selection hyper-heuristic framework as a tutorial-style introduction to the field. The performance and efficiency of the developed framework is evaluated across six well-known real-world problem domains.

The remainder of the chapter is organized into three main parts. The first part will present a timely overview of the current developments in the field of hyper-heuristics (section "Recent Advances in the Field of Hyper-heuristics"). The second part presents the development of a simple selection hyper-heuristic framework as a case study (section "Case Study: A Selection Hyper-heuristic-Based Iterated Local Search"). The last part evaluates the generality and performance of the developed framework on six different problem domains (section "Computational Results"). The chapter concludes with a brief discussion that summarizes the presented work (section "Conclusions").

Recent Advances in the Field of Hyper-heuristics

This literature review covers and discusses the latest articles that have been published in the field. It aims to cover as many of the current advances and trends as possible, without presenting overlapping material with relatively recent overview and survey papers [32–34, 36, 38, 102, 106, 107]. The main aim of this review is to cover the period since the publication of [36] in 2013.

The latest trends and directions regarding foundational and methodological aspects of hyper-heuristics are firstly presented in the literature review (section "Recent Advances on the Methodology of Hyper-heuristics"). Then it proceeds with discussion on the automatic design of algorithms (section "Automatic Design of Algorithms") and the latest theoretical works in the area (section "Recent Theoretical Results in Hyper-heuristics"). Hyper-heuristic developments in multi-objective and dynamic problem formulations are then reviewed (sections "Multi-objective Optimization" and "Hyper-heuristics in Dynamic and Uncertain Environments"). The intersection between machine learning and hyper-heuristic methodologies is studied in section "Machine Learning Methodologies", while section "Automated Parameter Control and Tuning" presents the recent advancements in hyper-heuristics for automatic parameter control. Finally, the most recent developments in application areas of hyper-heuristics are presented that include scheduling, timetabling, games, and various other areas (sections "Hyper-heuristics for Scheduling Problems", "Recent Educational Timetabling Applications of Hyper-heuristics", "Games and Education", and "Other Recent Developments").

Recent Advances on the Methodology of Hyper-heuristics

Starting from the foundations of hyper-heuristics and by studying the classifications of the area, the authors of [128] reformulate the mathematical definition of a hyper-heuristic. It is specifically exhibited that the new presented formulation naturally leads to a recursive definition of a hyper-heuristic. The authors draw parallels between the new hyper-heuristic formulation and a blackboard architecture from artificial intelligence, in which heuristics are able to annotate a shared workspace with information that can be exploited by other heuristics. In such a formulation, heuristics can share information at any level, which suggests that the domain barrier can be relaxed at any level required, without the loss of generality. A detailed description of the proposed formulation and its developments can be found in [128]. A concrete example application on the 3-SAT problem domain has been presented as a case study, which exhibits the improvements of the proposed idea over a wide set of problem instances. Other extensions of the existing formulations of hyper-heuristics have been proposed in [108]. This study advocates the need to reconsider the existing approaches to be able to produce self-organizing heuristics that will be able to automatically self-adapt to the environment when the underlying problem domain changes.

The need for more general and cross-domain efficient methodologies [36, 103, 107] has led to novel studies that try either to introduce unified representations or to embed common domain knowledge across various problem domains [49, 129].

In particular, the authors in [49] do not employ a high-level barrier across domains that are used in much of the hyper-heuristic literature. Instead, they propose an evolutionary memetic computing paradigm that is capable of embedding domain knowledge in memes across different but related problem domains. To improve search efficiency, the memetic paradigm is capable of learning and evolving knowledge memes on two different problem domains that can share a common representation, the capacitated vehicle and arc routing problems. Thorough analyses and experimental results show that the evolutionary search can benefit from different relevant problem domains. The knowledge that a meme has is crucial for the search procedure. It was observed that low discrepancy in the common feature space, between the problem domains, is able to enhance the evolutionary search procedure.

Furthermore, a general-purpose hyper-heuristic approach has been proposed that has the ability to solve combinatorial problems with minimal effort [129]. A unified representation and a set of domain-independent low-level heuristics are essentially proposed under a hyper-heuristic framework. To demonstrate the generality of the approach, the framework has been applied on different problem domains that share common characteristics but vastly differ as applications. An analysis reveals that the unified framework is very competitive in performance with state-of-the-art tailor-made heuristics for each problem domain.

The aforementioned studies suggest that both high and low levels of abstraction in the problem domain can enhance the search efficiency of hyper-heuristics. Such approaches demonstrate the real power of hyper-heuristic techniques. It is possible

to develop approaches that have enough generality and efficiency to enable them to produce competitive performance against tailor-made search algorithms for either a given problem class or across problem domains.

Various recent developments include novel hyper-heuristics that operate across the main categories of hyper-heuristics. Such an approach was introduced in [112]. It combines both heuristic selection and generation approaches. Specifically, it incorporates a dynamic multi-armed bandit with extreme credit values to online select the most appropriate low-level heuristic for the problem on hand. In parallel, instead of using a fixed acceptance criterion, it gradually generates new acceptance criteria for each problem instance through a gene expression programming framework. The effectiveness and generality of the approach are evaluated on exam timetabling instances from the 2007 International Timetabling Competition and on dynamic vehicle routing problem instances from the literature. Empirical results suggest that the introduced method shows competitive and general performance across both domains, compared with several state-of-the-art algorithms.

An automatic design framework was introduced in [110]. It automatically generates high-level heuristics for hyper-heuristic methodologies. The framework utilizes a gene expression programming algorithm to create high-level heuristics during the searching process. It captures information from the current state of the given problem and determines the low-level heuristic selection rule, as well as the solution acceptance criterion. As such, possibly new high-level rules can be produced for each problem instance. Experimental analyses on six problem domains exhibit its generality and competitive performance with state-of-the-art hyper-heuristic methodologies across all problem domains.

Techniques have been developed that are capable of learning and self-adapting their behavior based on the learned knowledge. More specifically, a hyper-heuristic system that embraces the concept of lifelong learning has been considered in [57, 120–122]. The system is based on a self-reliant network of interacting components inspired by artificial immune systems. It has the ability to continuously learn over time how to solve combinatorial optimization problems. It is capable of generating new heuristics continuously as well as sample problems from its environment. The core of the system is adaptable to its environment and has the intelligence to save and learn characteristic problem instances and heuristics from it. These characteristics result in an intelligent platform that exploits existing knowledge and quickly generates promising solutions of a given problem, as well as generalizing and adapting to new unseen problems with different properties. A rigorous experimental analysis on the bin packing domain demonstrates its excellent performance and generalization capabilities across a wide range of different problem instances. The dynamic and adaptive nature of the system makes it computationally efficient and scalable.

Various studies have been proposed that carefully investigate intrinsic characteristics of hyper-heuristics, such as diversity in the heuristic space, or understanding the behavior of the low-level heuristics. In particular, in [55], the impact and effect of a diverse set of available heuristic algorithms on the performance of a selection hyper-heuristic is studied. The study firstly makes an attempt to define

and measure the diversity of heuristic spaces and then proceeds to investigate the impact in performance of a hyper-heuristic that utilizes various strategies to manage the measured diversity. The diversity concept is studied with a hyper-heuristic that employs as low-level heuristic various black box optimization methods that operate on continuous spaces (see also [54]). The evaluation of the proposed method on a wide range of benchmark functions suggests that the heuristic space diversity significantly impacts upon the performance of the method.

The performance of low-level heuristics plays a crucial role in the performance of a hyper-heuristic methodology. In [115], the authors try to understand the performance efficiency of each low-level heuristic when it operates individually or in combination with other heuristics. As such, the concepts of competence and affinity (in terms of heuristics) have been defined and qualitatively estimated on a problem-specific hyper-heuristic approach. The resulting information from this process is exploited and used to improve the performance of problem-specific hyper-heuristics for the hybrid flow-shop scheduling problem. The tailor-made hyper-heuristics exhibit promising performance on the studied instances.

In parallel, novel search approaches have recently been incorporated within hyper-heuristic approaches. A novel diversification method is introduced in [105]. The method perturbs the problem instance under investigation to create justifiable new search directions. As such, various low-level heuristics that act upon this idea are proposed under a hyper-heuristic framework. The low-level heuristics are guided through an expressive grammar high-level strategy that has the ability to easily satisfy the problem constraints and thus easily produce feasible solutions. An experimental evaluation and analyses on the Ising spin glass and the p-median problem demonstrate its efficiency and its competitive performance.

When considering selection hyper-heuristics, each of the available search low-level heuristics (to be selected) is scored by a credit assignment mechanism. The credit assignment mechanism is responsible for assigning a reward, or credit value, to the applied heuristic, based on measurable feedback from the search procedure. The feedback may represent a measurement or qualitative characterization of the applied heuristic effects on the current state of the search process. The role of the credit value is crucial for the performance of the hyper-heuristic search algorithm, since it is the main information that guides the search procedure. Recently, different new types of feedback information have been introduced, based on landscape analysis techniques.

In [124], the evolvability of a search operator has been considered. The evolvability metric uses local characteristics of the fitness landscape to approximate the potential of a search component to generate improved solutions, from the current state of the search. An experimental study on benchmark problems reveals that evolvability feedback has significant potential to improve the performance of a hyper-heuristic, when compared against simple fitness improvement feedback.

Additionally, a novel selection hyper-heuristic methodology has been introduced in [39] to address the Capacitated Arc Routing problem. In the higher level, an online learning algorithm has been considered, the Dynamic Weighted Majority method, to predict and determine the most appropriate search mechanism. The

learning mechanism selects a crossover operator from an available set. Each of the available choices does not rely on its ability to improve the fitness of the generated solution. Instead, each operator can be scored based on four different fitness landscape analysis techniques. The analysis takes into account dispersion and neutrality as well as evolvability measures. Extensive comparisons on a set of benchmark problems verify the efficiency of the proposed methodology against state-of-the-art techniques.

Automatic Design of Algorithms

A comparison of the fields of meta-learning and hyper-heuristics has been recently proposed in [100]. The study draws parallels between methodologies and concepts from the two fields and advocates the common objective of automating the algorithm design process. A brief historical overview of automated algorithm design is presented along with a discussion on the similarities and differences between supervised machine learning and hyper-heuristics. Various issues are discussed, including the different levels of automation and generality, the problem and algorithm space, and various measures of performance.

The automatic design process was used to generate simple, general, modular, and efficient Iterated Local Search-based methodologies in [5, 6]. The generated methodologies are very competitive with the state-of-the-art methodologies in domain-independent meta-heuristic search [6]. Other hyper-heuristics based on Iterated Local Search that have been recently proposed include [97] in addition to the case study presented in the sections below. In parallel, a novel grammar representation has been introduced in [90] to be used for the automatic design of heuristic algorithms for hard combinatorial optimization problems. The proposed representation of the grammar includes a sequence of categorical, integer, and real-valued parameters and incorporates an off-line parameter configuration tool to search for algorithms for a given problem. Extensive experimentation on one-dimensional bin packing and on permutation flow-shop problems demonstrates the efficiency of both the grammar and the parameter configuration tools against an established grammatical evolution algorithm for automatic design. An extension of the methodology was able to build more complex algorithmic schemes from a composition of problem-dependent and problem-independent grammars [85].

A unified selection hyper-heuristic framework that handles one- and two-dimensional regular packing as well as two-dimensional irregular packing and cutting problems has been proposed in [80]. The framework utilizes several different stages to generate fast algorithms with competitive performance against the best problem-specific heuristic for a given problem. The framework generates different hyper-heuristic algorithms, by representing them as variable-length chromosomes of low-level heuristics that are being evolved by a messy genetic algorithm. A data mining-based feature selection procedure is used to determine the most relevant features in the problem-state representation, and an off-line analysis is employed to discover the most representative set of well-performing heuristics. The framework

has been evaluated on a large set of problem instances, and the experimental results show that it is able to generate very competitive hyper-heuristics for the given problem. In addition, the performance of the produced hyper-heuristics is comparable or even better than the application of the single low-level heuristics on their own. Moreover, an automatic design process has been proposed to efficiently solve two-dimensional rectangular blocks for packing [132].

Recently, the commonalities of certain areas of memetic computing and hyper-heuristics have been reviewed in [48]. This has led to the development of a new framework that combines a multi-meme approach with an adaptive operator selection hyper-heuristic to address continuous optimization problems. The framework rewards and adaptively selects the most suitable heuristics/memes during the search procedure. Although it has a simple structure, it exhibits high-performance gains against state-of-the-art search algorithms in a wide range of benchmark problems.

To avoid stagnation to local optima solutions, a multi-objective formulation has been proposed to formulate the single-objective two-dimensional packing problem [116]. A new memetic algorithm along with a parallel hyper-heuristic approach has been introduced to search the new formulation of the problem. The behaviors of the proposed methodologies are analyzed, and experimental results on two-dimensional packing problem instances suggest that the parallel hyper-heuristic is very competitive since it has found new best solutions for some problem instances.

The effectiveness and generality of a well-designed hyper-heuristic approach that outperforms human-designed problem-specific algorithms are demonstrated in [125]. Specifically, an Iterated Local Search-based hyper-heuristic methodology that learns from feedback of the problem at hand and dynamically selects among the available search operators is developed. An off-line learning approach is compared and contrasted with an online learning model on a large set of real-world problem instances from the first and second international timetabling competition. The online learning approach demonstrates superior performance over the off-line (static) model and is competitive with the state-of-the-art algorithms in the field.

Recent Theoretical Results in Hyper-heuristics

Although there is a significant growth of research in the field of hyper-heuristics, most studies are empirical. The theoretical foundations of hyper-heuristics are largely unexplored. Prominent examples of recent theoretical work in the field include mostly runtime analysis. A rigorous runtime analysis of hyper-heuristics has been presented in [74]. The analysis considers selection hyper-heuristics that operate on a predefined set of low-level heuristics. It reveals that the combination of heuristics can lead to exponentially faster search than deterministically chosen heuristics for a given problem. However, the analysis shows that an appropriate mixing distribution should be carefully selected.

A runtime analysis on some very common learning mechanisms has been performed in [8]. In particular, the analysis considers the simple random, random gradient, greedy, and permutation learning mechanisms in the case of selection

hyper-heuristics. The analysis shows that the learning mechanisms behave similarly, which suggests that they do not necessary improve the performance of the hyper-heuristics in the studied cases.

Multi-objective Optimization

A wide range of real-world problems require high-quality decisions for more than one objective. In general, an effective multi-objective algorithm should possess good search characteristics in terms of both solution diversity and convergence on the Pareto-front set. Several very effective multi-objective search algorithms have been proposed in the literature. They have different search dynamics, characteristics, advantages, and disadvantages. Hyper-heuristic approaches have also been explored within the context of multi-objective search (multi-objective hyper-heuristics).

The choice function approach has been used as a selection hyper-heuristic for a number of years. Interestingly, it still continues to be a promising and effective approach. The choice function along with the great deluge and late acceptance as nondeterministic move acceptance mechanisms has been investigated to address multi-objective optimization problems in [84]. The method utilizes the hyper-volume metric during the move acceptance process to effectively guide the search in promising areas and enhance both diversity and convergence characteristics of the method. The proposed method has been applied to traditional multi-objective benchmark problems in addition to the vehicle crashworthiness design problem [78]. Experimental analyses and results indicate that the proposed hyper-heuristic approach has the ability and potential to efficiently solve continuous multi-objective optimization problems.

Similarly, a hyper-heuristic method that combines the strengths of three well-known multi-objective algorithms has been introduced in [83]. The proposed approach implements an online learning choice function as a selection hyper-heuristic to combine the underlying algorithms. The choice function adaptively ranks the performance of the algorithms and determines which algorithm to apply at the next step. The introduced hyper-heuristic framework exhibits very effective performance on a wide range of multi-objective benchmark functions and on a real-world problem case, the vehicle crashworthiness design problem. The employed online learning mechanism greatly enhances the performance of the hyper-heuristic algorithm which is able to efficiently combine and exploit the advantages of the multiple low-level multi-objective algorithms.

In parallel, a multiple evolutionary algorithm portfolio has been developed in [139]. This approach utilizes simultaneously multiple algorithms for solving multi-objective optimization problems. The performance of each algorithm is evaluated by a score function, and the hyper-heuristic approach selects which to apply in the next step based on its score. The developed approach exhibits promising results. Another example of a recently proposed hyper-heuristic has been presented in [93]. This method includes diversity-based techniques to solve the patrol scheduling problem.

The concept of adaptive operator selection in multi-objective algorithms was introduced in [76]. This utilizes state-of-the-art models for adaptive operator selection to select the most suitable operator at each stage of the algorithm. The methodology extends the well-known MOEA/D algorithm, which decomposes the multi-objective problems to single-objective problems and optimizes them simultaneously. Thorough experimental analyses indicate that the proposed methodology is robust, its performance is significantly better than other state-of-the-art multi-objective algorithms, and the operator selection process works effectively. Likewise, a hyper-heuristic approach has been recently incorporated into the Multi-Objective Particle Swarm Optimization algorithm [37]. This is a particle swarm optimization variant for multi-objective problems. Its main characteristic is that it has to employ a leader selection strategy and an archiving mechanism.

In [91], a genetic programming-based hyper-heuristic is proposed to solve the bi-objective aspect of the water distribution network design problem. The goal here is to find the optimal pipe sizes required by a network to provide best service with minimal costs. The objective of the proposed hyper-heuristic methodology is to generate efficient search operators (mutation operators) for multi-objective algorithms applied to solve the bi-objective formulation of the problem. Interestingly, the generated mutation operators reveal useful information for the designer. For example, an evolved operator incorporates domain-specific knowledge (pipe smoothing), while others are able to reproduce well-known operators from the literature. The methodology exhibits significant potential on various problem instances, while the evolved operators are general and can be used as new algorithmic components for future multi-objective algorithms. A further analysis of the strengths of various search operators along with the features of the problem is provided in [92].

Hyper-heuristics in Dynamic and Uncertain Environments

In the last few years, hyper-heuristic methodologies have been employed to address applications that are inherently dynamic and uncertain. This poses a major challenge for the research community since the majority of the developed methodologies that operate on static environments cannot easily cope with the dynamic and uncertain aspects of some applications. Various recent hyper-heuristic methodologies embrace adaptivity and are able to cope with dynamically changing problem scenarios.

A hybrid hyper-heuristic has been introduced to consider dynamic optimization problems [133]. This combines a selection hyper-heuristic and a memory/search algorithm. The hyper-heuristic is able to select between the available low-level heuristics for the problem instance in hand. To evaluate its performance efficiency, the following two problems have been considered: the dynamic generalized assignment problem and the moving peaks benchmark. Experimental results indicate that the hyper-heuristic framework performs significantly better than the memory/search algorithm in the majority of the tested cases. From the considered selection heuristics, the choice function with the only improving acceptance criterion seems

to have the most promising performance, since it is able to track changes in the environment and it can rapidly react to different types of changes.

An additional recent study on hyper-heuristics in dynamic environments includes [126]. This approach analyzes the performance of a hyper-heuristic methodology that utilizes specialized meta-heuristics across different types of dynamic environment. An interesting real-world problem with uncertainty has recently been efficiently addressed by a hyper-heuristic approach [7]. This is the online path planning for autonomous unmanned aerial vehicles. Online path planning is a challenging problem that includes problem scenarios in uncertain or unknown environments. A hyper-heuristic methodology has been implemented to navigate these unmanned vehicles using a 3-D online path planning model with sensing uncertainty. An experimental study on various terrains with different characteristics demonstrates the efficiency of the methodology to create efficient navigation path plans.

Machine Learning Methodologies

Machine learning methodologies play an important role in the field. Various methodologies have been incorporated mostly into the higher level of hyper-heuristics to learn the behavior of the underlying heuristics and to make effective decisions to guide the search procedure toward the most promising areas of the search space. In parallel, the automatic design concept in hyper-heuristics is widely applicable in the machine learning field to generate novel and more efficient machine learning methodologies for specific classes of problems or tasks. Recent advances include a wide range of methodologies.

A new machine learning approach based on tensor analysis was introduced in [18, 19]. It utilizes tensor analysis techniques and in particular tensor factorization to reveal latent relationships between the low-level heuristics and the hyper-heuristic. Intuitively, it models the history trace of the considered hyper-heuristic method and tries to identify the low-level heuristics that exhibit promising performance. Tensor analysis and factorization are employed to identify promising interactions and features between low-level heuristics as the search method operates, by observing feedback from the search history and performance. The method then exploits the knowledge learned to improve its overall search ability.

Inspired by Inverse Reinforcement Learning theory, the authors in [17, 21] explored apprenticeship learning for generalizing experts' behavior. Apprenticeship learning or imitation learning or learning by demonstration has been widely applied in control and robotic applications. The main goal of apprenticeship learning is to learn by observing experts while they are in action. Within the context of hyper-heuristics, apprenticeship learning has been applied to learn expert behavior in vehicle routing [17] and online bin packing problem domains [21]. The learning procedure has been trained with diverse experts on small problem instances, and it has been tested on larger instances with different characteristics to test its generalization ability. The learning procedure successfully captures experts' different

actions and generalizes them on unseen problem instances. Experimental results and analyses suggest that the novel learning mechanism demonstrates outstanding performance gains in both problem domains.

Monte Carlo Tree search has been used in [111] to search the space of low-level heuristics for various problem domains. In particular, it models sequences of low-level heuristics as trees and searches for optimal sequences to be applied at the corresponding state of the search procedure. Experimental results on six problem domains demonstrate the generality of the method as well as its competitive performance.

An automated design of classification algorithms tailored to deal with both balanced and imbalanced data has been proposed in [25]. This aims to automatically design decision-tree induction algorithms. A thorough analysis is firstly performed to determine the impact of different kinds of fitness functions on the performance of the proposed methodology for both balanced and imbalanced datasets. The best-performing fitness function is then used for the automatic design process. The evolved decision-tree induction algorithm is compared against traditional decision-tree induction algorithms on a wide selection of datasets. The experimental analysis suggests that the proposed method significantly outperforms the other methods.

A grammatical evolution-based hyper-heuristic has recently been introduced for designing automatically split criteria in decision trees for data classification tasks [27]. A hyper-heuristic evolutionary algorithm to automatically build Bayesian Network Classifiers tailored to a specific dataset [109] represents another recent addition to the literature.

Automated Parameter Control and Tuning

Automated parameter control represents one of the major applications of hyper-heuristics. A recent overview of the trends and future directions of the area can be found in [67].

Two interesting parameter control methods based on hyper-heuristics and fuzzy logic have been recently proposed in [118, 119]. They investigate the impact of solving single-objective optimization problems with multi-objective approaches. The utilized methodology is a bi-objective approach that uses diversity as a second objective. A probabilistic selection hyper-heuristic and a fuzzy logic method have been utilized to control the parameters of the bi-objective methodology. Experimental analysis on a wide range of problems reveals the efficiency of both parameter control methods against several static parameter settings and self-adaptive rules. Overall, the resulting method is computationally efficient and is able to outperform the single-objective scheme in most of the tested cases. A similar approach has been applied to address the frequency assignment problem [117]. Additionally, fuzzy system methods have been investigated as selection hyper-heuristics to control parameters [62]. Off-line parameter tuning has been employed to fine-tune and successfully generate efficient heuristics for online bin packing [137].

A reinforcement learning-based parameter controller has recently been published in [64] to dynamically adapt the control parameters of evolutionary algorithms. The method is a generic and parameter-independent controller that can be easily adapted on any evolutionary algorithm. Experimental analysis that incorporates the proposed parameter control method on various state-of-the-art evolutionary algorithms, to address several continuous optimization problems, suggests that the method is capable of good performance. The controller is able to improve the quality of the solutions found by the algorithms with minimal effort and additional computational resources. Similar approaches include entropy-based controllers [9]. Furthermore, a recent study investigates the impact in performance of various feedback mechanisms that have been used for adaptive parameter control in evolutionary algorithms [10]. The study considers indicative feedback measures for both single- and multi-objective optimization problem formulations.

Time series prediction methodologies have recently been used [11] as parameter controllers to accurately predict suitable parameter values during the search process. In particular, various prediction methods have been utilized to predict future parameter values based on historical data from the search procedure. Evaluation of the considered methodologies on evolutionary algorithms suggests that the simple linear regression performs quite well, but without having a significant difference against the studied methods. The impact in performance of predictive parameter control methods is evident only when the performance data complies with the required assumptions of the prediction model, resulting in significant performance differences when compared against state-of-the-art parameter controllers. Otherwise, the prediction method does not exhibit any evident impact on the performance of the considered algorithm.

Hyper-heuristics for Scheduling Problems

As widely identified in previously published overview and survey articles [32–34, 36, 38, 106, 107], scheduling problems have played a major role in the development of hyper-heuristic approaches.

The automatic design of dispatching rules has been the subject of recent research attention because of the time-consuming process of manually designing such rules. An analysis of the representation of dispatching rules has been recently undertaken in [31]. The representation of dispatching rules essentially determines the search space and its complexity. As such, efficient representations highly impact upon the search process. Three different kinds of representations have been discussed in [31]: a linear combination of attributes, artificial neural networks, and a tree-based representation. Empirical analysis on the suitability of each representation in dynamic stochastic job-shop scenarios suggests that the tree representation, evolved with a hyper-heuristic genetic programming method, is the most efficient. Artificial neural network representations operate well mostly on processes with a small computational budget, while linear representations operate competitively only on quite small computational budgets.

Dispatching rules for semiconductor manufacturing have been also automatically generated with genetic programming [58]. Grammatical evolution has been successfully applied as a hyper-heuristic methodology to address capacitated vehicle routing problems [87, 88]. Comparisons between adaptive and grammar-based hyper-heuristic approaches have been recently performed for various problem domains [86]. Moreover, a genetic programming-based hyper-heuristic has been considered in [101] to tackle order acceptance and scheduling problems in make-to-order manufacturing systems. This automatically generates dispatching rules that have stochastic behavior in order to improve the search procedure. The evolved rules exhibit high-performance gains compared to rules produced by either tailor-made heuristics or a simple version of the considered genetic programming method.

Several novel multi-objective genetic programming-based hyper-heuristics have recently been proposed that are able to generate scheduling policies in job-shop environments [96]. This paper proposes an automatic design approach to generate dispatching and due-date assignment rules for a dynamic version of the multi-objective job-shop scheduling problem. Having identified the complexity and the labor needed to manually design an effective scheduling policy, four multi-objective algorithms are developed to automate the process. All approaches are capable of handling multiple scheduling decisions in parallel. Thorough experiments and analyses demonstrate that the evolved Pareto sets represent effective scheduling policies that dominate policies from combinations of existing dispatching rules. These policies also have promising performance on unseen scenarios with various shop configurations. Generally all proposed methodologies exhibit not only effective but also very meaningful scheduling policies that help the decision-maker to understand their characteristics.

A two-phase search approach has been introduced in [95] to tackle the Workover Rigs Scheduling problem for maintenance services in onshore oil wells. The problem can be characterized as a parallel heterogeneous machine scheduling problem derived from the maintenance planning of heterogeneous wells. This problem is to find schedules for a small number of work-over rigs that minimize the production loss associated with the large number of wells waiting for service. The first phase of the proposed approach is a selection hyper-heuristic that searches for a promising initial solution to warm-start the second stage, which utilizes an exact branch, price, and cut approach. The considered hyper-heuristic uses learning mechanisms to learn from feedback during search and select the most effective low-level heuristic to apply in the next step. Having applied the abovementioned methodology to a variety of real-world problem instances from Petrobras, the Brazilian National Petroleum Corporation, the authors were able to identify classes of heuristics, which seem to be more efficient for solving the problem. In general, it was observed that the learning mechanisms were very effective and able to guide the search to promising areas of the solution space. It is worth noting that the hyper-heuristic approach generated solutions that were very near to the optimal solutions found by the exact algorithm.

An interesting analysis has been performed in [94] that investigates the performance impact of heuristics while solving a problem with routing and rostering characteristics. The implemented hyper-heuristics act as analysis tools on the

heuristic behavior. The aim is to analyze the behavior of the available heuristics and determine the requirements that a heuristic should consider to solve such problems. The following three different scheduling and routing problems have been considered: home care scheduling, routing and rostering of security guards, and maintenance personnel scheduling. The analysis revealed that some specific features of the problems under investigation significantly affect the performance of the heuristics. These features are the number of activities, the number of resources, and the planning horizon. The observed information of such analysis can be exploited and used in future search approaches to efficiently solve similar problems that share common characteristics. As such, hyper-heuristics can be employed as analysis tools for a particular problem class under investigation and may offer novel information on the characteristics of the problem class.

Nurse rostering is one of the applications where hyper-heuristics find wide applicability, with recent works including research in selection hyper-heuristics [14, 24]. A generic cooperative agent-based search framework has recently been introduced [89] that is able to incorporate the search strengths of various metaheuristics to efficiently solve nurse rostering problems. The framework promotes the asynchronous cooperation of effective agents using pattern matching and reinforcement learning techniques. This enables the exploration of different fairness objectives across several real-world rostering problem instances. A thorough experimental analysis demonstrates the efficiency of such frameworks to generate high-performance rostering solutions in minimal time.

Several hyper-heuristic methodologies draw upon swarm intelligence techniques, such as particle swarm and ant colony optimization. A particle swarm optimization-based hyper-heuristic approach has been introduced to address the resource-constrained project scheduling problem [72]. This operates on the heuristic space by representing its particles as ordered sequences of heuristics to apply for the given problem. Subsequently, a feasible solution is constructed by a serial scheduling construction mechanism, while the generated solution is improved by double justification local search. Several problem instances from the well-known PSPLIB library have been used to test and compare the hyper-heuristic algorithm with other algorithms. Computational results verify its efficiency and competitive performance.

The problem of intercell scheduling with single-processing machines and batch-processing machines has been tackled in [75] with an ant colony optimization hyper-heuristic search method. Generally, intercell transfers in cellular manufacturing systems are essential to substantially reduce production costs. The formulation of this scheduling problem considers the following three subproblems in parallel: an assignment, a sequencing, and a batch subproblem. An ant colony hyper-heuristic has been adapted to search among different assignment heuristic rules for both parts and machines simultaneously. Subsequently, the discovered rules are applied to generate schedules. The efficiency of the proposed method is evaluated on randomly generated problem instances. The experimental results indicate that the method is significantly more efficient than CPLEX and genetic algorithms. Channel

scheduling in multicell networks has also been addressed recently with hyper-heuristic methodologies [42].

A novel hyper-heuristic methodology has been proposed in [134] to solve task scheduling in cloud computing systems. Usually, rule-based scheduling algorithm is used for task scheduling in such systems. The method presented here is able to learn from the performance of several available scheduling algorithms and select the most suitable algorithm to apply. It incorporates a diversity and fitness detection strategies to efficiently balance the diversification and intensification behavior of the search procedure. This approach significantly outperforms several state-of-the-art scheduling algorithms on both simulation and real system scenarios. It produces solutions that have a significantly better makespan compared against other scheduling algorithms.

The problem of resource provisioning-based scheduling tasks in large-scale distributed environments, such as grids, has been addressed by exploring a hyper-heuristic scheduling method [16]. The main goal of this methodology is to efficiently schedule jobs on the available resources in order to simultaneously minimize execution time and increase security and reliability. The proposed hyper-heuristic algorithm is a particle swarm optimization-based algorithm that has been specifically developed to efficiently schedule jobs on available resources while addressing security requirements and constraints. A thorough experimental analysis reveals that the hyper-heuristic algorithm outperforms the state-of-the-art approaches, in terms of minimizing time, cost, and the makespan of the scheduling process.

Hyper-heuristics also have wide applicability in transportation scheduling. Motivated by a real railway-based disaster transportation relief scenario in the area of China, a selection hyper-heuristic search method [140] has been proposed to tackle emergency railway transportation problems. The evolutionary hyper-heuristic utilizes various search strategies as low-level heuristics and rewards each search strategy by assessing its ability to generate successful movements in the search space. The method has been compared with various state-of-the-art evolutionary algorithms on several problem instances, which have been created from disaster rescue operations data in China. Experimental results show high-performance gains across all instances against the considered evolutionary algorithms. The method has also been successfully applied on a real emergency during the 2013 Dingxi earthquake in China, producing a very satisfactory solution.

Another recent application of hyper-heuristic methodologies is the dial-a-ride transportation problem with time windows [135]. This is concerned with scheduling a set of available vehicles to pick up customers from an origin location and deliver them to a destination. It considers a range of constraints. The proposed hyper-heuristic methodologies are efficient and result in competitive performance against state-of-the-art approaches from the literature. Moreover, cooperative hyper-heuristics have been successfully applied to bus driver scheduling problems [77].

Recent Educational Timetabling Applications of Hyper-heuristics

A recent overview and critical analysis of the literature on the hyper-heuristic methodologies applied to educational timetabling problems have been published in [102]. The article focuses on the hyper-heuristic research in three general educational timetabling problems, the university examination, the university course, and the school timetabling problems, as well as proposes several future research directions on the subject.

An estimation of distribution algorithm has recently been introduced [104] as a general hyper-heuristic approach to address exam timetabling problem instances. The objective was to develop a hyper-heuristic with increased generality over different problem instances. The hyper-heuristic operates on sequences of low-level exam-selection heuristics that construct a timetable, based only on non-domain-specific knowledge. The resulting algorithm is capable of guiding the search to very promising areas. It generates efficient sequences of timetabling heuristics. Experimental results on various real-world exam timetabling and graph coloring problem instances suggest that the hyper-heuristic is generic across all problem instances, with competitive performance against other hyper-heuristic approaches. In addition, the developed approach is capable of learning the most promising heuristic for the problem at hand.

A two-stage hybrid hyper-heuristic approach has recently been proposed to solve timetabling problems [35]. The two-stage hybrid approach utilizes a Kempe chain move heuristic and the time-slot swapping heuristic as search operators. Firstly, it generates and automatically analyzes random heuristic sequences. The analysis keeps the repeated heuristics of the best sequence and empties the remaining positions to be processed in the next stage. The second stage randomly assigns sequences if heuristics are in the empty positions in order to search for high-quality sequences of heuristics. The constructed ones are used for the given problem. The hybrid method is evaluated on two challenging sets of problems, the Toronto benchmark and the exam timetabling set of problems from the second International Timetabling Competition. The method exhibited competitive performance against the state-of-the-art methodologies for both problem sets.

New selection hyper-heuristic approaches that are based on simple stochastic search methods have exhibited outstanding performance in the recent high school timetabling competition [69]. Indeed, such approaches have provided exceptional results and new best solutions to very challenging problem instances. A hybrid hyper-heuristic which utilized sequences of low-level heuristics has also been applied recently in examination timetabling problems [15].

Games and Education

Hyper-heuristic methodologies have recently been proposed in the areas of games and education. A recent study focuses on solving the Jawbreaker puzzle [113].

The evolutionary process operates on the low-level heuristic space and produces sequences of low-level heuristics that can be applied to solve the given puzzle. The resulting methodology is able to efficiently solve instances of the puzzle with various sizes and levels of difficulty, including very difficult levels.

An educational game-based software tool has been introduced in [114] to teach the intrinsic operational characteristics of hyper-heuristics to engineering students. The tool is based on the Bubble Breaker puzzle, and it represents a strong synergy with the concept of selection hyper-heuristics. In addition, a hyper-heuristic method is used, under a context-aware ubiquitous learning environment, to maximize the learning efficacy of students [138]. The main goal of the search method is to locate quality learning paths for students by considering real-world context and constraints. The proposed approach is evaluated on real-world scenarios.

Other Recent Developments

The general character and the wide applicability of hyper-heuristics enable them to efficiently address a wide variety of applications. Here, the most recent developments in various scientific fields are presented, from linear algebra to aircraft structural design applications. A genetic programming-based hyper-heuristic is introduced in [71] to evolve the structure of the Sloan algorithm that aims to find a reduced envelope size of matrices. The envelope is the sum of the distances between each element of the matrix and its main diagonal. Its size has a major impact on the performance of large linear systems solvers. The best evolved algorithm (with the addition of a local search method) exhibits substantial performance gains and significantly outperforms state-of-the-art algorithms in the literature on classic problem sets.

The magic square problem has been efficiently tackled by a wide range of selection hyper-heuristic approaches [68] that utilize perturbative low-level heuristics.

A modified version of the well-known and widely used choice function has been successfully hybridized with a late acceptance strategy hyper-heuristic to tackle the multidimensional 0–1 knapsack problem [43]. In addition, a stochastic hyper-heuristic version of the choice function has been utilized to address the winner determination problem in [30, 73].

Another interesting application of hyper-heuristics is the real-world warehouse storage location assignment problem [136], in which a genetic programming methodology is employed to generate matching functions that guide the selection of subsets of items for a given problem. The methodology is able to locate high-quality solutions on the training problem cases. It is also able to perform well on unseen scenarios in the testing phase. This indicates that the evolved heuristics are reusable and can be directly applied to similar problem scenarios. Furthermore, aircraft structural design optimization problems have recently been efficiently addressed by hyper-heuristic methodologies [12, 13].

The problem of distributing resources among multiple threads in a processor has been studied in [56]. A learning hyper-heuristic that predicts the best-performing

heuristic between a set of available heuristics has successfully been applied to the problem. On average, its performance is comparable or significantly better than the state-of-the-art techniques in the field.

Case Study: A Selection Hyper-heuristic-Based Iterated Local Search

In this section, a simple selection hyper-heuristic framework is demonstrated to provide a tutorial-style introduction to hyper-heuristic development. Specifically, a hyper-heuristic framework is developed that is based on the well-known Iterated Local Search algorithm. The development process and the decisions that had to be made during implementation are briefly discussed (section "Iterated Local Search with Adaptive Heuristic Selection"). The framework implements an action selection model that operates on the Iterated Local Search algorithm to adaptively select among various search low-level perturbation heuristics (section "Action Selection Models"). It is based on the HyFlex platform which can be used to easily implement prototypes of hyper-heuristic search methods without having to be concerned about the implementation of problem domain low-level heuristics (which are available from the HyFlex site) (section "The HyFlex Framework").

The HyFlex Framework

HyFlex is a Java-based software framework that enables the easy development and testing of cross-domain hyper-heuristic search methodologies. The framework implements a common software interface to provide all the essential ingredients for easily developing a general-purpose search algorithm, without having to implement specific problem domain knowledge.

The framework implements six different combinatorial optimization problems, namely, maximum satisfiability (Max-SAT), one-dimensional bin packing, personnel scheduling, permutation flow-shop scheduling, the traveling salesman problem, and the vehicle routing with time windows problem. For each problem domain, there are 10 to 12 available problem instances, including both challenging benchmark and real-world industrial problem instances. The framework additionally exposes the user to a broad set of state-of-the-art low-level heuristic search operators with different search dynamics and characteristics. For all problem domains and all instances, appropriate evaluation functions have been implemented to easily evaluate generated solutions.

HyFlex currently has two main versions, 1.0 and 1.1. The first version includes the main functionality described above, while the second version adopts the concept of batch-mode hyper-heuristics, i.e., allowing the developed hyper-heuristic methodology to operate on a whole set of instances for each run, instead of using a single instance. A description of the APIs and documentation for both versions can

be found in [20,98] and in [1,3,4]. The HyFlex framework has been utilized in two cross-domain search challenges, CHeSC 2011 [1] and CHeSC 2014[3].

In this chapter, the first version of the framework has been adopted to develop the hyper-heuristic presented in the next section.

Iterated Local Search with Adaptive Heuristic Selection

This section discusses the motivation (and provides a detailed description) for designing a simple and general applicable hyper-heuristic search algorithm. However, the main purpose of this section is not to design a hyper-heuristic that has the best performance obtained in the literature so far. Instead, the goal of this section is to demonstrate how a simple search algorithm can be seen as a general hyper-heuristic approach. Moreover, it aims to show how the search algorithm can easily be applied to different problem domains without considering many unusual and made-to-measure design choices. The resulting approach can be easily fine-tuned on the application domains at hand and produce state-of-the-art performance gains [5,6,90].

The developed approach is based on a simple, general, and highly successful local search algorithm, the Iterated Local Search algorithm [29, 60, 82]. Iterated Local Search (ILS) is a single-point local search method that belongs to the wide category of stochastic local search methods [60]. The main idea behind its structure is to "iteratively create a sequence of potential solutions produced by a given heuristic search method, that leads to much better solution quality, than performing repeated random executions of it" [29, 60, 82]. In general, ILS includes four main stages in its structure. Firstly, ILS generates an initial solution for the problem at hand and then iteratively improves the current solution by applying a diversification (perturbation) and an intensification (local search) operation. Diversification is accomplished by applying a perturbation heuristic on the current solution, while intensification is achieved by employing a local search method on the perturbed solution. Having performed these two search operations, ILS determines whether the new solution is accepted to the next iteration, via an acceptance criterion. As such, in order to create an effective ILS variant, one should carefully determine the perturbation heuristic, the local search, and the acceptance criteria. A vast amount of available choices for various problem domains can be found in the literature [29,60,82].

Notice that the strength of perturbation heuristics varies among the available heuristics and greatly influences the behavior of the algorithm. For example, strong perturbations vastly increase the diversification of the search and might lead to a random restart-like behavior, i.e., there is a very low probability to find better solutions. In contrast, perturbations with small strength might lead to very similar solutions that will not help the local search procedure find a better solution. Instead, the local search might revisit its previous solutions. Similarly, acceptance criteria can influence the balance between the intensification and diversification search abilities of the algorithm. For example, the acceptance of improving solutions will

increase the intensification behavior of the algorithm, while accepting worsening moves will increase diversification.

Naturally, several ILS variants adapt their search strategies as well as their properties during the search procedure to react on the respective stage of the search. Representative reactive search principles can be found in [28, 29, 60]. Arguably this is the essence of a hyper-heuristic methodology [32–34, 36, 38, 102, 106, 107]. Motivated by these findings, this approach is adopted to develop a simple and general hyper-heuristic search framework and apply it to various problem domains. In particular, a simple framework of Iterated Local Search-based hyper-heuristic (HHILS) is implemented to exhibit the development process and the design choices that an interested researcher, or practitioner, might be faced with during the development phase of the hyper-heuristic search methodology.

In general, the HHILS framework follows the basic algorithmic structure of an Iterated Local Search methodology and enhances its reactive search ability with an action selection and a credit assignment module. The action selection module is devoted to efficiently predict and select the most suitable action to apply in the next step, while the credit assignment module is responsible for assigning a reward, or a credit value to the applied action, based on feedback from the search procedure. Arguably, the action selection module can be applied on any level or combination of levels within the search procedure, i.e., it can select among different perturbation heuristics, local search methodologies, and acceptance criteria, or any possible combination of them. This design choice is based on the level of adaptivity or reactivity that the researcher/practitioner is willing to incorporate in the search algorithm. In many cases, this decision will have a major impact on the performance of the resulting search methodology, since the adaptive procedure may rapidly change the dynamics of the search procedure, i.e., the diversification and intensification levels of the search. Here, a simple case is adopted in which the action selection module will operate only in the perturbation phase of the algorithm, i.e., it will be able to select from a set of available perturbation low-level heuristics.

The selected and applied action is scored based on the credit assignment module. The main role of the credit assignment module is to assign a reward, or credit value, to the applied action, based on feedback from the search procedure. The feedback can be seen as one or more measurements (or qualitative characterizations) of the applied action effects on the current state of the search process. The most common measurement for a search action is the quality improvement of its state (e.g., the difference in fitness value between the previous and the current state). However, various other qualitative different characterizations may be used, such as ranking successful movements [50], and evolvability [39, 124] of the search procedure. Such feedback is able to change the search characteristics of the algorithm and thus guide the search based upon different objectives. For example, the resulting algorithm will have more diversification characteristics or will avoid stagnating during the search procedure. Arguably, the quality of the feedback has a major impact on the performance and the success of the action selection model. As such, the literature includes various different attempts to investigate the impact of different feedback characterizations, like fitness improvement [46, 50, 53, 99], successful

Algorithm 1: Pseudo code of the HHILS framework of ILS based hyper-heuristics.

1: Initialize the action selection, and the credit assignment module as well as create all data structures required by HHILS.
2: s_{cur} ← GenerateInitialSolution() /* **Generate Initial Solution:** Initialize or construct a solution for the problem instance at hand. */
3: **while** termination criteria do not hold **do**
4: selected$_{llh}$ ← ActionSelection(str) /* **Action Selection:** Select a low-level heuristic with the str-th action selection model. */
5: s_{tmp} ← ApplyAction(s_{cur}, selected$_{llh}$) /* **Perturbation:** Perturb the current solution (s_{cur}) with the selected low-level heuristic (selected$_{llh}$). */
6: s_{tmp} ← ApplyLocalSearch(s_{tmp}) /* **Local Search:** Apply a local search procedure on the temporary solution s_{tmp}. */
7: s_{cur} ← AcceptanceCriterion(s_{cur}, s_{tmp}) /* **Acceptance:** Accept which solution, between s_{cur}, and s_{tmp}, will survive to the next iteration based on an Acceptance criterion. */
8: CreditAssignment(selected$_{llh}$) /* **Credit Assignment:** Score the used action (selected$_{llh}$) based on feedback from the problem at hand. */
9: **end while**
10: **Return:** the best solution found so far s_{cur}.

movements [47], rank based improvements [50], and evolvability [39, 124]. The selected feedback for the HHILS framework is based on the mean improvement of the applied perturbation heuristics and will be explained in detail below.

A family of different HHILS variants can be easily produced by adopting new action selection models and different credit assignment feedback functions. Algorithm 1 presents the general structure of the HHILS framework. Although some design choices have been made based on specific details related to the HyFlex framework (which was used for the implementation of HHILS), the algorithmic structure of the HHILS framework is relatively general and can be easily adapted to any other available framework.

HHILS firstly initializes all the necessary modules and data structures that are required by the algorithm to run properly (line 1 of Algorithm 1). This procedure includes the initialization of the credit assignment module with an empty or zero initial reward as well as the initialization of the action selection module that sets all actions to have equal chances to be selected. Next, HHILS initializes a solution (s_{cur}) for the problem instance at hand (line 2). The initial solution can be either generated randomly or constructed with a constructive heuristic. This depends on the problem domain and the available method to initialize, or construct, a feasible solution. HyFlex provides such functionality for all problem domains which is convenient for easily constructing an initial solution independently of the problem domain.

Having found an initial solution, each iteration of HHILS performs the following five main steps. Let set $\mathcal{S}$ define the κ available perturbation low-level search heuris-

tics $\mathcal{S} = \{llh_1, llh_2, \ldots, llh_\kappa\}$, for the problem domain in hand. In HyFlex, the developer has the option to select between greedy and non-greedy heuristics. Here, all available non-greedy heuristics for the current domain have been considered, which correspond mainly to *mutation*- and *ruin-and-recreate*-type-based heuristics. HHILS employs an action selection method (`ActionSelection(str)`, line 4) to predict and select the most appropriate perturbation low-level search heuristics to apply at the next step (selected$_{llh} \in \mathcal{S}$). The `ActionSelection` method might include various models, such as uniform selection and proportionate selection based on the reward values. The user is able to define which model to use with the str variable $str \in \mathcal{M}$, where $\mathcal{M}$ is the set of the available m action selection models, $\mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_m\}$. Section "Action Selection Models" briefly describes the models used in this chapter.

After selecting the selected$_{llh}$ perturbation low-level heuristic, HHILS proceeds with its application on the current solution s_{cur} to perturb its current position and create a new solution, s_{tmp} (line 5). The new position (s_{tmp}) is being immediately refined by the `ApplyLocalSearch(`s_{tmp}`)` procedure, which applies a greedy local search heuristic. In HyFlex, greedy local search heuristics belong in the *local search* category and have the property that they are not able to produce a worse solution than the given one. To avoid making a poor choice for the current problem instance, all of the available local search methods are applied in an iterative way [6]. More specifically, given a list $\mathcal{L}$ of the available λ local search heuristics, $\mathcal{L} = \{l_1, l_2, \ldots, l_\lambda\}$, at each iteration, a local search method is selected in a uniform random way and is applied to the current solution (s_{tmp}). If the application of the selected local search method does not lead to a better position (fitness improvement), it is excluded from the active list, until another local search heuristic finds an improved solution. If all the local search methods in the active list do not lead to an improved solution, then a local optimum has been reached. In this case, the iterative process will finish with the best solution found so far (s_{tmp}). The algorithmic scheme of the iterative local search procedure is illustrated in Algorithm 2.

Algorithm 2: Pseudo code for the HHILS `ApplyLocalSearch(`s_{tmp}`)` method.

1: $\mathcal{L}_{active} \leftarrow \mathcal{L}$
2: **while** $\mathcal{L}_{active}$ is not empty **do**
3: $i \leftarrow$ uniformly random select a local search from $\mathcal{L}_{active}$
4: $s_{ls} \leftarrow l_i(s_{tmp})$: Apply l_i local search on s_{tmp}
5: **if** $f(s_{ls}) < f(s_{tmp})$ **then**
6: $s_{tmp} \leftarrow s_{ls}$
7: **else**
8: $\mathcal{L}_{active} = \mathcal{L}_{active} \backslash \{i\}$
9: **end if**
10: **end while**
11: **Return:** the best solution found so far s_{tmp}.

Having applied the aforementioned transformations on s_{cur}, a new temporary solution s_{tmp} has been created. In the next step, HHILS employs an acceptance criterion `AcceptanceCriterion`(s_{cur}, s_{tmp}) (line 7 of Algorithm 1) procedure that is responsible to decide which of the two solutions will be accepted for the next iteration. In the literature, there are various choices available for the acceptance criterion [60], where the most common ones are to accept only improved solutions or to accept worsening solutions based on a probability distribution. These criteria will affect the diversification and intensification abilities of the current search method, for the given problem. Their impact depends on the landscape structure of the problem at hand. HHILS utilizes a Metropolis-based acceptance condition that accepts a worse solution based on a probability distribution. Intuitively, the greater the fitness improvement, the greater the probability, or the greater the worsening fitness, the smaller the likelihood to accept the worse solution. A representative example of such an acceptance condition is the simulating annealing method [70].

Specifically, the probability density function of acceptance is defined as $p(f(s_{cur}),\ f(s_{tmp}), T, \mu_i) = e^{\frac{f(s_{cur})-f(s_{tmp})}{T\cdot\mu_i}}$, where $f(s_{cur})$ and $f(s_{tmp})$ are the objective value of the s_{cur} and s_{tmp} solutions, respectively, T is a constant temperature value with $T \in \mathbb{R}$, and μ_i is the mean improvement of the improving iterations [6]. The role of μ_i is to normalize the objective value difference by a quantity that does not depend on the problem at hand. Notice that the value of the temperature parameter is problem dependent and has to be fine-tuned for the considered problem in order to obtain optimal performance. Here, the temperature is fixed in all experiments to $T = 2$. A short and interesting study on the temperature behavior can be found in [6].

By this stage, HHILS has finished with the searching operations, and the applied low-level perturbation heuristic can be scored based on its search behavior. The final step of HHILS, before proceeding to the next iteration, is to employ the credit assignment module `CreditAssignment(`selected_{llh}`)` to score the selected perturbation heuristic, selected_{llh} (line 8 of Algorithm 1). As such, HHILS defines the reward of an action as the improvement moves performed by the action, normalized by the total time it spends on them. Such a reward is a representative score for the improvement rate of each action over the time spend for each improvement. Specifically, let $\mathcal{A} = \{a_1, a_2, \ldots, a_\alpha\}$ be the available set of α actions (note that in HHILS $\mathcal{A} = \mathcal{S}$) and t_{α_i} be the time consumed by action α_i on searching the solution space (initially $t_{\alpha_i} = 0, \forall \alpha_i \in \mathcal{A}$). Then at each iteration, the time spent for the applied action, i, is calculated as $t_i = t_i + t_i^{spent}$, where t_i^{spent} is the time of action i spent on the current iteration. The reward value (r_i) of the applied action (i) can be calculated according to $r_i = \frac{1+improvement_i}{t_i}$, where $improvement_i = f(s_{cur}) - f(s_{tmp})$. In general, various different reward approaches can be used at this point, such as the instantaneous latest reward or the average or a ranked-based reward [50]. However, some of them tend to be unstable and noisy estimations of credit due to the stochastic nature of the search process. To alleviate this drawback, the empirical quality of an action is estimated by utilizing the average value of a sliding window of its latest w rewards. More specifically, let

W_i be a set of the latest w improvement rewards (r_i) achieved by the action a_i during the time step t of the hyper-heuristic. The final credit assignment value $r_{a_i}(t)$ is the average reward of the sliding window W_{a_i}, which can be determined by

$$r_{a_i}(t) = \frac{\sum_{j=1}^{|W_{a_i}|} W_{a_i}(j)}{|W_{a_i}|} \tag{1}$$

where $|W_{a_i}|$ indicates the cardinality of the set W_{a_i}.

Finally, the algorithm will iterate until one or more termination criteria hold. Here, the maximum allowed CPU wall clock time is used as termination criterion, $t_{allowed}$, which the algorithm may consume to solve the problem under investigation. It is worth noting that the HHILS framework does not consider whether the algorithm gets trapped in a particular position of the solution space for a long period of time. This is a very common situation in which either the algorithm is in a local optimum or the search operators are not able to find a better solution for a long period of time. In such cases, a restarting procedure is essential to help the searching capabilities of the algorithm to diversify the current solution. However, a restarting procedure requires specific design choices that for the sake of simplicity are overlooked in this tutorial demonstration.

Action Selection Models

As discussed previously, HHILS is a general framework that has the ability to adopt any action selection model. The literature includes various action selection models from different scientific fields such as statistics, filter theory, artificial intelligence, and machine learning. Each model has different advantages and disadvantages that depend on the properties of the underlying reward distribution that it is trying to predict. Representative examples of such models that have been used for this purpose include probability matching [130, 131], adaptive pursuit [130, 131], statistical-based models like the multinomial distribution with history forgetting [46, 47], evolutionary meta-learning methods [44, 45, 79, 123], and reinforcement learning approaches [50, 51, 127].

In this chapter, five different models have been utilized: the uniform selection model, the proportional selection model, the probability matching [130, 131], the adaptive pursuit [130, 131], and the upper confidence-bound multi-armed bandit methodology [50, 127].

The first two models are very simple and act as baseline models for the HHILS framework. The uniform selection model simply selects between the available actions randomly by following a uniform distribution. In this case, the scores assigned to each action do not have any impact on the selection process. This is the baseline methodology that will provide insights into the usefulness and impact of applying an action selection model on HHILS. Notice that several studies have identified the usefulness of random variation in either action or parameter

spaces [63, 65, 66]. The second model performs proportional selection among the available actions based on their assigned scores from the credit assignment module. Intuitively, this model provides proportional chances to each action based on their reward scores during the optimization phase, i.e., the most successful will have more chances to be selected again in the next step. The proportional selection is performed by applying a simple roulette wheel selection [23].

Probability matching (PM) [50, 130, 131] is a simple probabilistic model that updates the selection probability of each action proportionally to its empirical quality with respect to the other actions. Specifically, given a set $\mathcal{A} = \{a_1, a_2, \ldots, a_\alpha\}$ of the available α actions (here $\mathcal{A} = \mathcal{S}$), let $P(t) = \{p_{a_1}(t), p_{a_2}(t), \ldots, p_{a_\alpha}(t)\}$ be the selection probability vector of all actions, where initially $p_{a_i} = 1/\alpha, \forall a_i \in \mathcal{A}$. After the application of the perturbation, the local search, and the acceptance criterion stages, a reward $(r_{a_i}(t))$ is measured from the credit assignment module. Notice that in the general case, the reactive procedure operates on a nonstationary environment where the estimation of the empirical quality of an action might be more accurate and reliable if the more recent rewards influence the empirical quality more than the past ones. Therefore, in such cases, it is a common practice by various action selection models to estimate the empirical quality $q_{a_i}(t)$ of each action (a_i) in accordance with the following relaxation mechanism:

$$q_{a_i}(t+1) = q_{a_i}(t) + \gamma(r_{a_i}(t) - q_{a_i}(t)) \tag{2}$$

where $\gamma \in (0, 1]$ is the adaptation rate (here γ is fixed to 0.1) [50, 130, 131]. Having calculated the empirical quality of each action (a_i), PM adapts its selection probability (p_{a_i}) using the following rule:

$$p_{a_i}(t+1) = p_{\min} + (1 - \alpha \cdot p_{\min}) \frac{q_{a_i}(t+1)}{\sum_{i=1}^{\alpha} q_{a_i}(t+1)} \tag{3}$$

where $p_{\min} \in [0, 1]$ is the minimum probability value of each action. $p_{\min}$ is responsible for ensuring that each action will always have a small probability (here $p_{\min} = 0.05$) [130, 131]. Having estimated the probabilities of each action, PM utilizes a stochastic proportional selection mechanism, based on these probabilities, to select which action will be applied in the next step.

Notice that PM allows the inefficient actions to have at least $p_{\min}$ selection probability. Thus, the best action will be selected with probability $p_{\max} = (1 - (\alpha - 1) \cdot p_{\min})$ However, this hinders the efficiency and performance of PM, since all inefficient actions keep being selected. Therefore, the adaptive pursuit (AP) strategy [130, 131] addressed this drawback by embracing a winner-takes-all strategy. In detail, AP, instead of updating proportionally the action probabilities, increases the probability of the corresponding best action $(a_{i^\star})$, while in parallel, it decreases the probabilities of the other actions according to the following equations:

$$a_{i^\star} = \arg \max_{i=1,2,\ldots,\alpha} \{q_{a_i}(t+1)\} \tag{4}$$

$$p_{a_i}(t+1) = \begin{cases} p_{a_i}(t) + \beta(p_{\max} - p_{a_i}(t)), \text{ if } a_i = a_{i^\star} \\ p_{a_i}(t) + \beta(p_{\min} - p_{a_i}(t)), \text{ otherwise} \end{cases} \tag{5}$$

where $\beta \in [0, 1]$ is a learning rate that manipulates the greediness of the winner-takes-all strategy (here $\beta = 0.8$). Intuitively, AP first finds the best action of the current step and then updates the probabilities by being in favor of the best action.

The last action selection model used in this chapter formulates the action selection model as a multi-armed bandit (MAB) problem and utilizes the upper confidence-bound (UCB) algorithm to solve it. The MAB problem considers κ one-arm slot machines and a player that has to decide which arm to pull, how many times to pull it, and in which order to choose between the available arms, having in mind that each machine provides a random reward. Intuitively, the MAB problem faces the well-known trade-off of *exploration versus exploitation*, where each time the player has to decide either to "exploit" its current knowledge, i.e., to pull the arm that gives the highest expected payoff, or to "explore" the available choices, i.e., to acquire more information about the expected payoff of the other arms.

More specifically, let us have κ arms (one-arm slot machines), where each arm (i-th) has a fixed unknown reward probability $p_i \in [0, 1], \forall i \in \{1, 2, \ldots, \kappa\}$. Assume that the arms are independent and that the associated rewards with each arm are independently and identically distributed. At each time step (t), the player selects an arm (j) and obtains a reward $r_t = 1$ with probability p_j or $r = 0$ otherwise. At any stage of the procedure, the performance of the MAB algorithm is estimated either by calculating the cumulative reward at the current stage or by employing a regret measure, i.e., the difference in performance between the current and the best arm. As such, the main objective of the selection algorithm at any stage is to either maximize the cumulative reward of the procedure or minimize its regret. In such a context, an "optimal" algorithm will have the best possible performance if at each time step it is able to select the best (unknown) arm, i.e., the arm with the maximum probability of obtaining a reward.

The upper confidence-bound (UCB) algorithm is implemented to solve the MAB problem [22, 50, 51]. It is worth mentioning that UCB achieves an optimal regret rate on the aforementioned formulation. Specifically, let $q_{a_i}(t)$ be an estimation of the empirical quality of action a_i on time step t, and let c_i be a confidence interval associated with the empirical quality of action a_i that depends on the amount of times $n_i(t)$ action i has been tried until the current step t. Then, at each time step, UCB deterministically selects the next action ($selected_{llh}$) with the optimal upper bound of the confidence interval c_i. The selection process can be calculated according to

$$selected_{llh} = \arg \max_{i=1,2,\ldots,\alpha} \left(q_{a_i}(t) + \mathbf{C} \cdot \sqrt{\frac{2 \log \sum_k n_k(t)}{n_i(t)}} \right) \quad (6)$$

$$n_i(t+1) = n_i(t) + 1 \quad (7)$$

$$q_{a_i}(t+1) = \frac{(n_i(t)-1) \cdot q_{a_i}(t) + r_{a_i}(t)}{n_i(t)} \quad (8)$$

where $\mathbf{C}$ is a user-defined scaling factor that balance the trade-off between exploitation and exploration ability of the model (here $\mathbf{C} = 0.8$). Intuitively, the first term of Equation 6 favors the action with the best empirical quality (exploitation), while the second term benefits the trial of the other actions (exploration). UCB thus selects mostly the action that potentially provides the best reward, while giving the opportunity for infrequently tried actions to be applied regularly.

An analysis of the behavior and adaptability of the applied models should provide useful insights into the behavior of each model and its impact on the respective algorithm that incorporates it. However, this issue is beyond the scope of this tutorial chapter.

Computational Results

This section demonstrates a thorough analysis of the performance of the developed hyper-heuristics by evaluating them on a wide range of problem domains with different characteristics. The experimental protocol used in the experiments is firstly defined, and then two qualitative different analyses are presented. The first study provides comprehensive experimental results and statistical analyses of the developed hyper-heuristics on each of the considered problem domains separately, while the second study follows the experimental protocol used in the CHeSC 2011 [1] competition and compares the developed hyper-heuristics with the state-of-the-art approaches of the competition.

Experimental Protocol

Five different hyper-heuristic approaches have been developed that are based on the HHILS framework described previously in section "Iterated Local Search with Adaptive Heuristic Selection". The difference between the hyper-heuristics mostly lies in the action selection mechanism used to select among the available low-level perturbation heuristics. As such, comparisons between the following five hyper-heuristics are conducted:

- HHILS: The simplest HHILS variant that selects the available perturbation low-level heuristics randomly following a uniform distribution.

- HHILS-SA: An HHILS variant that proportionally selects which perturbation low-level heuristic to apply, based on their reward value.
- HHILS-PM: An HHILS variant that utilizes the probability matching selection method to select the perturbation low-level heuristics.
- HHILS-AP: An HHILS variant that utilizes the adaptive pursuit selection method to select the perturbation low-level heuristics.
- HHILS-MAB: An HHILS variant that utilizes the upper confidence-bound multi-armed bandit methodology to select the perturbation low-level heuristics.

All HHILS variants have been implemented in the Java programming language, based on the functionality provided by the HyFlex framework. Although their implementation details are specific for the HyFlex framework, their algorithmic design is general and can be easily adapted and developed in any other framework. To motivate the usage and further development of the HHILS variants, their source code is available at https://github.com/mikeagn/hhils.

To maintain a fair and reliable comparison, the same parameter configuration is used for the common parameters of all algorithms. In addition, for the different action selection models that have been incorporated in the HHILS variants, the default parameter settings are used as proposed in the literature. Specifically, all HHILS variants employ a temperature value of $T = 2.0$, while the action selection models employ the default values used in the literature as described in the previous section. Notice that it has not been performed any fine-tuning process to obtain high-quality parameter configurations. However, such parameter configuration values can be found by performing either manual or automatic tuning and sensitivity analyses. This tuning process might lead to superior performance gains; nevertheless, this is out of the scope of the current tutorial demonstration. Prominent examples of successful off-line automatic tuning tools among others are the `irace` [81], the `SPOT` [26], and the `SMAC` [61] tools.

Six different problem domains have been implemented within the HyFlex framework, namely, the Max-SAT (SAT), the bin packing (BP), the personnel scheduling (PS), the Flow Shop (FS), the traveling salesman problem (TSP), and the vehicle routing with time windows problem (VRP). For each problem domain, the HyFlex framework provides 10–12 problem instances. A problem instance corresponds to either a real-world realization or a well-known and challenging benchmark case of the considered problem domain. More information about the available instances can be found online in [2]. To evaluate the performance of an algorithm on a given problem instance, the best objective value achieved by the algorithm within a prespecified available time budget is used. Problem domains have been modeled as minimization problems. As such, the lower objective value indicates a better performance. The first part of the experimental analyses provides extensive experimental results for all available instances per problem domain.

For fair comparisons, a benchmarking program is provided on the CHeSC 2011 website [1]. This determines the allowed time limit ($t_{allowed}$) of each run under the specific hardware architecture $t_{allowed}$ which corresponds to 10 min of CPU time on the machine used during the competition. All experiments in this study have been

conducted on an AMD Opteron CPU of 2.2 GHz machine with 32 GB of RAM running GNU/Linux operating system. For this machine, the allowed time limit is 624 s ($t_{allowed} = 624$ s).

Finally, the second part of the analyses strictly follows the rules of the CHeSC 2011 competition in order to provide an easy and fair comparison between all hyper-heuristic search methodologies. A thorough description of the experimental setup used in section "Comparison to HyFlex State-of-the-Art Hyper-heuristics" can be found in [98].

Experimental Results

In this section, the HHILS variants are evaluated on the six available problem domains provided by the HyFlex framework. Specifically, the performances of the five developed HHILS variants are compared on all available instances per problem domain. Firstly the performance gains obtained by each algorithm are described separately, for each problem domain. Then, their overall performance is summarized, and the observed behavior is verified by statistical analysis.

Tables 1, 2, 3, 4, 5, and 6 show statistics on the performance of each algorithm per problem domain, in terms of the best objective value obtained over 31 independent runs. For each algorithm and each problem instance (I), the following statistics are provided: the best (min), the median (m), the mean objective value (μ), and the standard deviation (σ) from the mean objective value obtained by the algorithm at hand. The cases where either the best, the median, or the mean objective value is the smallest (i.e., best performance) across all algorithms for a problem instance in hand are highlighted with **boldface**. In general, it can be observed that, although each algorithm behaves differently across the various problem domains, some algorithms exhibit robust performance across the majority of the domains, such as the HHILS-AP and the HHILS-SA.

More specifically, it can be easily observed that HHILS-AP shows superior performance against the other algorithms in three different domains, BP, FS, and TSP. As presented in Tables 1, 2, and 5, for at least 60% of the problem instances, HHILS-AP shows superior mean and median performance, while it is able to find the best objective value against the other algorithms in 6, 8, and 7 instances for the BP, FS, and TSP, respectively. Regarding the VRP problem domain, although there are three instances where the observed median and mean value are the smallest, in the majority of the remaining cases, it is not able to show competitive performance against the first-ranking algorithms in the domain (HHILS-SA, HHILS-PM).

The next most promising hyper-heuristic is HHILS-SA, which is able to achieve best performance, in terms of both mean and median objective values, in three problem domains FS, PS, and VRP. HHILS-SA also demonstrates the best performance in several problem instances across the majority of problem domains, i.e., 11, 6, 4, 4, and 2 in SAT, VRP, FS, PS, and TSP, respectively. The remaining HHILS variants perform similarly for the majority of the considered domains. HHILS and HHILS-MAB do not exhibit observable performance differences for the majority

Table 1 Statistics on the best objective value achieved by the developed HHILS variants on all problem instances of the bin packing domain

	HHILS				HHILS-AP				HHILS-MAB				HHILS-PM				HHILS-SA			
I.	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ
i0	0.007	0.017	0.017	0.005	0.007	**0.007**	**0.007**	0.000	**0.006**	0.007	0.008	0.001	0.007	0.011	0.010	0.002	0.007	0.007	0.008	0.002
i1	0.012	0.017	0.017	0.004	**0.007**	**0.008**	**0.008**	0.000	0.007	0.008	0.008	0.001	0.008	0.008	0.009	0.002	0.007	0.008	0.008	0.001
i2	0.022	0.024	0.024	0.001	**0.021**	**0.022**	**0.022**	0.000	0.022	0.024	0.024	0.002	0.022	0.023	0.023	0.001	0.023	0.023	0.023	0.000
i3	0.023	0.026	0.025	0.001	**0.021**	**0.022**	**0.022**	0.001	0.024	0.026	0.026	0.002	0.023	0.024	0.024	0.000	0.023	0.024	0.024	0.000
i4	0.007	0.007	0.007	0.001	0.005	0.005	0.005	0.000	**0.000**	**0.005**	**0.005**	0.001	0.005	0.007	0.006	0.001	0.005	0.006	0.006	0.001
i5	0.004	0.009	0.008	0.002	0.004	0.004	0.004	0.000	**0.003**	**0.003**	**0.003**	0.000	0.004	0.004	0.005	0.002	0.004	0.004	0.004	0.001
i6	0.084	0.089	0.088	0.003	**0.011**	0.037	0.045	0.032	0.025	0.032	0.033	0.003	0.022	0.032	0.032	0.008	0.011	**0.016**	**0.018**	0.005
i7	0.107	0.112	0.112	0.003	**0.028**	0.088	0.068	0.029	0.061	0.076	0.074	0.007	0.043	0.069	0.070	0.012	0.033	**0.051**	**0.051**	0.013
i8	0.049	0.052	0.052	0.001	**0.039**	**0.047**	**0.046**	0.005	0.057	0.066	0.065	0.003	0.045	0.049	0.049	0.002	0.046	0.049	0.049	0.001
i9	0.009	0.014	0.013	0.002	0.008	0.010	0.010	0.001	0.016	0.019	0.019	0.001	**0.007**	0.009	0.010	0.001	0.008	**0.009**	**0.009**	0.001
i10	0.110	0.111	0.111	0.001	0.109	0.109	0.109	0.000	**0.108**	**0.108**	**0.108**	0.000	0.110	0.110	0.110	0.000	0.109	0.110	0.110	0.000
i11	0.029	0.031	0.031	0.002	0.017	0.020	0.020	0.002	0.033	0.037	0.037	0.002	**0.012**	**0.017**	**0.016**	0.002	0.018	0.023	0.023	0.002

Table 2 Statistics on the best objective value achieved by the developed HHILS variants on all problem instances of the flow-shop domain

	HHILS				HHILS-AP				HHILS-MAB				HHILS-PM				HHILS-SA			
I.	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ
i0	6328.0	6358.0	6356.1	12.8	**6308.0**	**6333.0**	**6331.0**	7.4	6327.0	6346.0	6346.5	9.4	6319.0	6343.0	6340.6	7.5	6324.0	6345.0	6344.2	6.1
i1	6282.0	6304.0	6302.4	8.4	6263.0	6288.0	**6285.6**	9.4	**6260.0**	6297.0	6295.7	9.0	6270.0	6294.0	6293.6	9.6	6270.0	**6287.0**	6288.0	6.4
i2	6370.0	6386.0	6384.9	7.3	6352.0	**6366.0**	**6363.8**	5.9	6365.0	6382.0	6382.5	7.8	6352.0	6372.0	6370.0	7.2	**6349.0**	6368.0	6367.3	7.5
i3	6361.0	6369.0	6369.2	3.3	**6323.0**	**6352.0**	**6352.4**	10.9	6350.0	6369.0	6367.2	6.0	6337.0	6367.0	6365.1	6.3	6330.0	6359.0	6357.7	11.0
i4	6424.0	6451.0	6447.1	10.7	**6402.0**	**6417.0**	**6416.3**	6.9	6409.0	6444.0	6442.0	11.9	6408.0	6426.0	6426.7	10.7	6407.0	6425.0	6424.2	7.2
i5	10501.0	10525.0	10525.6	9.9	**10495.0**	**10501.0**	**10500.9**	3.6	10517.0	10531.0	10529.5	5.9	10501.0	10516.0	10516.5	8.0	10497.0	10509.0	10508.2	6.0
i6	10948.0	10959.0	10960.4	4.4	**10923.0**	**10944.0**	**10943.6**	10.2	10938.0	10960.0	10959.7	6.6	10934.0	10957.0	10955.4	6.8	**10923.0**	10956.0	10950.7	10.2
i7	26290.0	26402.0	26394.8	36.3	**26249.0**	**26293.0**	**26293.0**	20.7	26339.0	26407.0	26408.3	26.4	26291.0	26343.0	26343.5	23.9	26266.0	26324.0	26322.8	27.7
i8	26859.0	26886.0	26895.1	28.1	26765.0	26831.0	26823.6	24.7	26852.0	26896.0	26895.5	21.2	26806.0	26866.0	26865.3	28.8	**26761.0**	**26815.0**	**26812.2**	20.4
i9	26663.0	26731.0	26725.3	27.9	**26580.0**	**26645.0**	**26641.6**	28.3	26667.0	26712.0	26717.9	25.1	26642.0	26704.0	26703.0	23.8	26616.0	26682.0	26679.5	23.2
i10	11420.0	11473.0	11471.4	20.5	**11398.0**	**11432.0**	**11430.2**	14.7	11442.0	11464.0	11466.9	11.8	11431.0	11458.0	11457.5	12.0	11404.0	11445.0	11444.4	12.4
i11	26629.0	26690.0	26694.2	27.3	26567.0	26640.0	26632.6	31.9	26611.0	26700.0	26694.2	27.2	26621.0	26678.0	26675.3	26.4	**26548.0**	**26638.0**	**26631.0**	28.1

Table 3 Statistics on the best objective value achieved by the developed HHILS variants on all problem instances of the personnel scheduling domain

	HHILS				HHILS-AP				HHILS-MAB				HHILS-PM				HHILS-SA			
I.	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ
i0	**3299.0**	**3317.0**	3319.6	12.7	**3299.0**	3318.0	**3318.5**	11.7	3312.0	3335.0	3335.9	14.9	3301.0	3325.0	3325.3	13.7	3301.0	3322.0	3321.5	11.8
i1	**1948.0**	2172.0	2155.8	101.3	2022.0	2175.0	2184.2	85.9	1950.0	2210.0	2208.5	88.3	1967.0	2137.0	2136.9	89.3	1955.0	**2103.0**	**2118.0**	98.1
i2	325.0	365.0	364.5	18.3	**305.0**	360.0	359.7	19.9	335.0	380.0	377.3	21.1	335.0	360.0	360.5	13.5	325.0	**355.0**	**354.8**	16.1
i3	13.0	18.0	18.3	2.1	13.0	19.0	19.0	2.4	17.0	21.0	21.4	2.6	**12.0**	**17.0**	**18.1**	3.5	**12.0**	18.0	**18.1**	3.0
i4	**14.0**	20.0	20.0	2.4	15.0	**19.0**	**18.9**	2.2	20.0	24.0	24.4	3.1	**14.0**	20.0	20.2	3.1	15.0	20.0	20.1	2.4
i5	16.0	21.0	21.3	3.0	**12.0**	**18.0**	**18.5**	2.6	17.0	25.0	24.8	3.2	14.0	20.0	20.6	2.8	16.0	21.0	22.0	3.1
i6	1108.0	1132.0	1148.4	40.2	1112.0	1132.0	1149.8	37.9	1108.0	1219.0	1213.6	66.2	**1103.0**	**1124.0**	**1132.2**	29.8	1107.0	1126.0	1141.1	38.6
i7	2170.0	2219.0	2230.1	38.1	2183.0	2262.0	2251.6	48.4	2195.0	2270.0	2262.6	51.7	2178.0	2225.0	2231.6	47.0	**2166.0**	**2211.0**	**2220.0**	34.0
i8	3159.0	3263.0	3263.3	65.5	3157.0	3256.0	3276.3	81.8	3158.0	3292.0	3308.6	76.8	3156.0	3238.0	3227.4	51.7	**3138.0**	**3181.0**	**3207.0**	59.2
i9	9420.0	9902.0	10006.1	344.6	9429.0	9910.0	10079.2	516.9	**9407.0**	**9678.0**	**9671.2**	177.8	9672.0	9904.0	10090.1	444.0	9422.0	9958.0	9999.1	394.9
i10	**1405.0**	1630.0	1613.7	100.3	1445.0	1580.0	1603.7	84.3	1450.0	1623.0	1639.3	97.1	1462.0	1610.0	1622.2	94.4	1435.0	**1565.0**	**1574.3**	75.0
i11	305.0	**335.0**	335.8	12.6	320.0	**335.0**	337.3	12.0	320.0	350.0	348.3	15.5	315.0	**335.0**	334.9	10.5	**300.0**	**335.0**	**334.4**	15.0

Table 4 Statistics on the best objective value achieved by the developed HHILS variants on all problem instances of the SAT domain

	HHILS				HHILS-AP				HHILS-MAB				HHILS-PM				HHILS-SA			
I.	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ
i0	2.00	**2.00**	2.00	0.00	2.00	**2.00**	2.00	0.00	2.00	**2.00**	2.00	0.00	2.00	**2.00**	2.00	0.00	**1.00**	**2.00**	**1.97**	0.18
i1	19.00	21.00	20.97	1.70	18.00	20.00	20.29	1.47	**17.00**	**19.00**	**18.97**	0.84	18.00	21.00	21.23	1.80	18.00	20.00	19.97	1.14
i2	**14.00**	16.00	16.52	1.61	**14.00**	16.00	16.10	1.04	**14.00**	16.00	15.68	1.11	**14.00**	16.00	16.03	1.28	**14.00**	**15.00**	**15.48**	1.03
i3	**0.00**	2.00	1.90	0.79	**0.00**	2.00	1.58	0.56	**0.00**	**1.00**	1.55	0.81	**0.00**	2.00	1.61	0.88	**0.00**	**1.00**	**1.52**	0.77
i4	1.00	2.00	1.71	0.78	1.00	**1.00**	1.32	0.48	**0.00**	**1.00**	1.55	0.81	**0.00**	**1.00**	1.42	0.56	**0.00**	**1.00**	**1.03**	0.55
i5	**1.00**	3.00	2.84	0.64	**1.00**	3.00	2.23	0.88	**1.00**	3.00	2.29	0.86	**1.00**	3.00	2.65	1.60	**1.00**	**2.00**	**1.97**	0.87
i6	**5.00**	**5.00**	**5.00**	0.00	**5.00**	**5.00**	5.19	0.48	**5.00**	6.00	5.81	0.87	**5.00**	**5.00**	5.03	0.18	**5.00**	**5.00**	5.45	0.62
i7	**5.00**	**5.00**	**5.06**	0.25	**5.00**	6.00	5.52	0.51	**5.00**	**5.00**	5.55	0.62	**5.00**	**5.00**	5.32	0.48	**5.00**	6.00	5.61	0.56
i8	**5.00**	**7.00**	**6.16**	1.19	**5.00**	8.00	7.13	1.31	**5.00**	8.00	7.45	1.21	**5.00**	8.00	7.29	1.22	**5.00**	8.00	7.26	1.37
i9	**209.00**	211.00	210.13	1.06	**209.00**	211.00	210.10	1.01	**209.00**	**209.00**	**209.26**	0.68	**209.00**	**209.00**	209.77	0.99	**209.00**	**209.00**	209.90	1.01
i10	**1.00**	2.00	2.16	1.19	**1.00**	**1.00**	1.42	0.76	**1.00**	**1.00**	1.32	0.83	**1.00**	**1.00**	1.16	0.37	**1.00**	**1.00**	**1.03**	0.18
i11	**7.00**	**7.00**	**7.39**	0.50	**7.00**	**7.00**	7.42	0.56	**7.00**	8.00	7.65	0.66	**7.00**	8.00	7.68	0.70	**7.00**	8.00	7.94	0.73

Table 5 Statistics on the best objective value achieved by the developed HHILS variants on all problem instances of the TSP domain

	HHILS				HHILS-AP				HHILS-MAB				HHILS-PM				HHILS-SA			
I.	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ
i0	48273.9	48392.2	48398.9	58.6	**48194.9**	**48214.9**	**48221.7**	28.7	48194.9	48225.7	48224.4	19.6	48209.7	48298.5	48291.2	30.6	48209.7	48270.2	48261.7	27.3
i1	108616.5	109392.8	109618.3	735.8	**107225.3**	109349.4	108800.4	962.6	108217.7	109167.8	109126.8	498.6	107442.3	108459.1	**108616.0**	817.0	107259.6	**108446.1**	108740.2	948.8
i2	6894.6	6923.7	6921.9	12.8	**6823.4**	**6844.5**	**6843.9**	10.7	6888.0	6915.9	6914.9	11.4	6863.2	6897.2	6896.6	14.2	6862.2	6895.6	6893.7	11.8
i3	42546.5	42709.8	42716.7	85.5	**42173.0**	**42279.8**	**42274.5**	60.8	42348.6	42614.6	42608.2	81.4	42467.0	42589.7	42576.6	63.1	42375.1	42512.2	42500.6	62.1
i4	9009.4	9048.3	9047.0	15.4	**8922.5**	**8943.0**	**8943.8**	11.2	8958.2	8973.4	8975.9	11.5	8985.8	9016.0	9012.9	14.4	8963.5	9004.1	9004.2	16.4
i5	58192.9	58612.5	58592.8	156.2	**57294.8**	**57560.0**	**57551.4**	141.8	58207.2	58766.1	58743.6	234.4	57975.3	58184.4	58197.8	137.1	57779.9	57977.5	58006.1	153.2
i6	55088.6	58971.0	58648.4	1749.7	53077.3	54605.6	54902.8	1189.3	53779.3	**54030.7**	**54031.2**	111.1	53096.6	56898.8	56873.8	1882.7	**52857.0**	54295.0	54257.8	781.0
i7	68711.3	69749.9	69723.7	474.9	**65789.9**	**66301.9**	**66296.2**	233.7	68339.4	68560.9	68593.5	158.3	67049.2	67786.3	67892.4	443.9	66088.5	67098.6	67047.9	399.6
i8	**21090984.4**	21380811.8	21396672.8	187926.6	21145557.5	21375242.0	21394858.1	165005.6	21164043.9	**21284114.6**	**21276739.0**	53894.4	21145557.5	21374787.3	21400271.2	160545.3	21210701.9	21364765.7	21377212.8	144263.6
i9	672053.3	676463.7	676895.2	2362.8	673105.2	676356.6	677073.4	2506.1	673688.4	675793.9	675873.1	1241.5	673583.9	676599.4	677243.7	2427.2	**671431.6**	**675118.2**	**675141.7**	1863.9

Table 6 Statistics on the best objective value achieved by the developed HHILS variants on all problem instances of the VRP domain

	HHILS				HHILS-AP				HHILS-MAB				HHILS-PM				HHILS-SA			
I.	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ	min	m	μ	σ
i0	5130.6	5165.7	5164.1	13.5	5115.0	5156.9	5155.6	13.9	**4250.7**	5156.7	**5124.0**	162.7	5114.0	**5156.5**	5158.0	14.4	5130.6	5165.4	5163.4	10.6
i1	20655.0	20657.4	20657.8	1.7	20652.2	**20654.0**	**20654.5**	3.1	20652.6	20656.5	20721.0	247.1	20652.5	20655.2	20655.0	0.6	**20652.2**	20654.7	20654.5	0.6
i2	12290.7	12323.8	12451.4	349.5	**12274.4**	**12294.6**	**12293.2**	15.6	12305.4	13343.0	13242.6	404.9	12282.3	12303.7	12305.0	16.0	12281.3	12307.1	12306.7	12.8
i3	5348.1	5373.7	**5374.6**	10.0	5351.4	6250.1	5939.5	435.2	**5321.4**	6245.4	6158.2	268.2	5326.7	**5366.1**	5453.2	277.0	5341.9	5367.3	5481.2	310.8
i4	13281.3	13293.6	13356.8	244.9	**13272.0**	**13286.9**	13695.2	493.3	13291.5	14280.2	14217.5	355.9	13277.3	13288.1	**13352.1**	245.8	13275.1	13287.5	13383.9	294.6
i5	197926.3	204256.8	204910.1	3616.1	142488.6	145038.2	**144824.7**	1402.2	210603.3	219067.4	219536.2	3525.1	143961.3	148171.3	148519.8	3054.0	**142479.0**	**144184.5**	144854.3	1676.7
i6	64490.5	67479.9	67417.3	1335.8	61316.4	64272.3	64115.7	1719.2	71101.8	72849.9	73027.1	1233.4	60121.0	62336.5	62360.7	988.9	**59730.3**	**61651.6**	**61552.1**	951.9
i7	173238.4	177164.0	176870.5	1251.6	157870.2	158386.5	158516.4	579.8	184077.7	188024.2	187791.9	1622.3	157881.5	158794.6	158839.0	545.1	**157770.8**	**158339.2**	**158343.0**	360.4
i8	165077.3	171211.4	171229.5	2639.7	143085.2	146652.3	147012.3	1496.2	188337.3	196181.7	196299.1	3476.2	143584.5	148048.2	148395.3	2280.6	**142928.5**	**144561.6**	**144347.1**	1222.1
i9	159486.8	162101.1	162205.1	1293.7	144086.8	145586.4	145595.4	668.4	165525.7	168651.6	168616.4	1493.4	144244.0	146098.4	146209.1	916.1	**143565.7**	**144827.5**	**144974.6**	862.5

of the studied cases, while HHILS-PM shows slight improvement gains against them for the BP, FS, and TSP domains (in terms of mean and median objective values). However, in general, the performance of HHILS-PM is more robust when compared with both HHILS and HHILS-MAB, since the mean objective values and their standard deviations are lower for the majority of the considered benchmarks.

It is worth noting that in the SAT problem domain, most of the HHILS variants behave equally well, since for the majority of the problem instances, they exhibit similar mean and median performance values. In addition, for most of the problem instances, they successfully reach the best objective value achieved in this study. Notice that observable performance gains have been shown mostly by HHILS-SA which reached the smallest (min) objective value for all problem instances.

To facilitate easy comparisons of the performance of each algorithm across all problem domains, instances, and independent runs, their objective values are normalized in a common range of values. As such, for a given problem instance, the obtained objective values of all algorithms defined on a range $O = [O_{\min}, O_{\max}]$ are transformed linearly to the normalized range $N = [N_{\min}, N_{\max}]$, where $O_{\min}$ and $O_{\max}$ are the smallest and the largest objective value observed by the considered algorithms, and $N_{\min} and N_{\max}$ are the lower and upper bound of the new normalized range. Here, the normalized range $N = [0, 1]$ has been used to keep calculations simple. Strictly speaking, let $y \in O \subset \mathbb{R}$ be an objective value obtained by an algorithm for a given instance, and let $f : O \rightarrow N$ be a linear function that transforms its input according to $\xi = f(y) = (y - O_{\min})/(O_{\max} - O_{\min})$. As such, the performance of each algorithm now can be computed by the set $Y_{\text{Alg}} = \{\xi_1, \xi_2, \ldots, \xi_n\}$ that consists of the normalized objective values across all problem domains, problem instances, and independent execution runs, where n indicates the total number of sample values and is equal to the number of problem instances for each domain, times the number of independent runs, summed up over all problem domains. Notice that this normalization enables a summarizing comparison of the algorithms across all problem domains and instances. Intuitively, the smallest normalized objective values indicate better performance.

To graphically demonstrate the distributions of the performance for each algorithm per problem domain, a summarizing illustration is also provided in Fig. 1. This demonstrates box-plot graphs of the normalized performance for the developed hyper-heuristics across all domains. In each box-plot graph, the mean value of the underlying distribution is additionally marked with a diamond.

Notice that the previously described behavior is clearly captured by the summarizing box plots illustrated in Fig. 1. On the whole, it can be easily identified that HHILS-AP exhibits great performance gains in three problem domains (BP, FS, and TSP), while HHILS-SA in two problem domains (VRP and PS). The next best-performing algorithm seems to be HHILS-PM which exhibits competitive performance in BP, PS, SAT, and VRP. HHILS and HHILS-MAB behave quite similarly, and their performance gains cannot be distinguished by the graphical illustration. Interestingly, it can be additionally observed that the performance distribution of HHILS is the most robust against the remaining HHILS variants on the SAT domain.

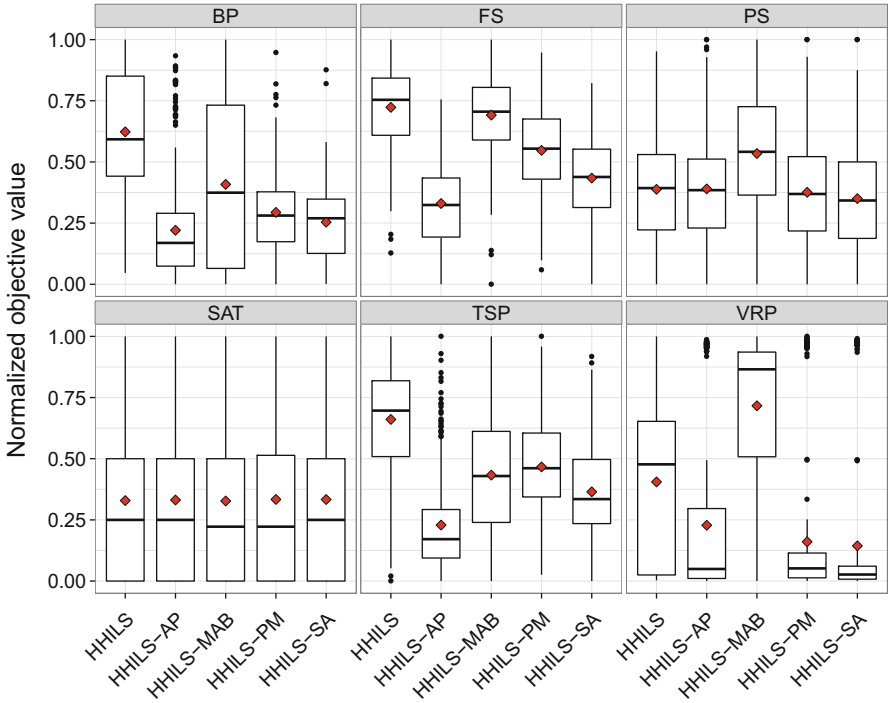

Fig. 1 Boxplot of the normalized objective values of all hyper-heuristics per problem domain

To assess the existence of statistically significant differences, of the observed performance values, between at least two hyper-heuristics across all considered problem domains and instances, the Friedman rank sum test is applied [59]. For each algorithm, the distribution sample of its mean normalized performance per problem instance across all instances and all domains considered in this study is used. The null hypothesis of the Friedman test states that the distributions of all samples are the same, against the alternative hypothesis which states that at least two samples are not the same [59]. Given the existence of significant differences in performance, a post hoc analysis is employed to determine which two algorithms exhibit significant differences in performance. As such, pairwise comparisons are conducted, and for each comparison, the p-values (p_w) calculated by the nonparametric Wilcoxon-signed rank test are reported, since the underlying sample distributions do not follow a normal distribution (tested with the Shapiro-Wilk normality test [59]). Furthermore, to alleviate from the problem of having type I errors in multiple comparisons with a higher probability, the Bonferroni correction method is applied, and the adjusted p-values (p_{bonf}) are reported.

The Friedman test strongly suggests the existence of statistically significant differences in the performances between the studied hyper-heuristics ($p = 0.0000$, with $\chi^2 = 83.071$). Therefore, a post hoc statistical analysis is employed to de-

termine which two algorithms significantly differ in performance. Table 7 presents summarizing statistics and p-values from pairwise statistical significance tests for all considered hyper-heuristics over all problem domains and instances. The left-hand side of Table 7 demonstrates the median (m_f), the mean (μ_f), and the standard deviation (σ_f) of the normalized objective values over all problem domains and instances, while the right-hand side presents the p-values obtained from the Wilcoxon-signed rank test (p_w) and the corresponding Bonferroni correction method (p_{bonf}). The presented statistics show that, on average, HHILS-SA and HHILS-AP are the most promising approaches. However, the statistical tests reveal that there are not statistically significant differences between their performances for the studied cases. Additionally, the pairwise comparisons indicate that HHILS and HHILS-MAB behave equally, while there exist statistically significant differences in performances for the remaining pairs of hyper-heuristics. This verifies the aforementioned reported performances observed in the analyses on each problem domain.

To further demonstrate the underlying performance distribution of the considered hyper-heuristics, Fig. 2 presents two graphical illustrations of their overall performances: a box-plot graph (left) and an empirical cumulative distribution function (ECDF) graph (right) on their normalized objective values over all the considered problem domains and instances. Such graphical illustrations are useful in displaying the characteristics of the distribution and the frequency of the observed performance values. Specifically, having measured a set of performance values (e.g., the normalized objective values) for an algorithm A, $M_A = \{\xi_1, \xi_2, \ldots, \xi_n\}$, the ECDF is defined as $F_n(\xi) = \frac{1}{n}\sum_{i=1}^{n} I_{(-\infty,\xi]}(\xi_i)$, where $I_{(-\infty,\xi]}(\cdot)$ is the indicator function which equals to one if $\xi_i \preceq \xi$ and to zero otherwise. Intuitively, ECDF captures the empirical probability of observing a value that is less than or equal to ξ. The larger a ECDF value corresponding to a given value ξ, the higher is the empirical probability of observing ξ in Y_A.

The box-plot graph (on the left-hand side of Fig. 2) clearly illustrates the differences between the performance distributions of the considered algorithms. Based on the illustrated distributions as well as the previous mentioned results and statistical analyses, it can be safely concluded that HHILS-AP and HHILS-SA demonstrate the most promising performance across all domains. Although, from the analyses conducted per problem domain, HHILS-PM has not exhibited any superior performance gains, it can be observed that its average performance is very promising. Finally, the remaining two hyper-heuristics HHILS and HHILS-MAB perform on average quite similarly. Moreover, the ECDF graph interestingly reveals that almost all algorithms are able to reach best-performing results with quite similar frequency across all problem domains and instances. However, it has to be noticed that HHILS-AP, HHILS-SA, and HHILS-PM have more chance to locate a better solution for the problem under investigation. For example, HHILS-AP has a chance of more than 50% to achieve a quite competitive performance value across all studied problem domains, i.e., there is a 50% chance that the obtained performance value will be in the first quartile of the observed performance values of all algorithms across all studied problem domains.

Table 7 Summarizing statistics of the normalized objective values of all hyper-heuristics over all problem domains and instances (left). Pairwise statistical significance tests of the normalized performances for all hyper-heuristics over all problem domains and instances (right)

Algorithm	m_f	μ_f	σ_f	HHILS		HHILS-AP		HHILS-MAB		HHILS-PM		
HHILS	0.542	0.521	0.293	p_w	p_{bonf}	p_w	p_{bonf}	p_w	p_{bonf}	p_w	p_{bonf}	
HHILS-AP	**0.222**	**0.292**	0.257	HHILS-AP	**0.000**	**0.000**	–	–	–	–	–	–
HHILS-MAB	0.533	0.515	0.309	HHILS-MAB	0.517	1.000	**0.000**	**0.000**	–	–	–	–
HHILS-PM	0.347	0.366	0.263	HHILS-PM	**0.000**	**0.000**	**0.001**	**0.005**	**0.000**	**0.001**	–	–
HHILS-SA	**0.297**	**0.317**	0.251	HHILS-SA	**0.000**	**0.000**	0.604	1.000	**0.000**	**0.000**	**0.000**	**0.000**

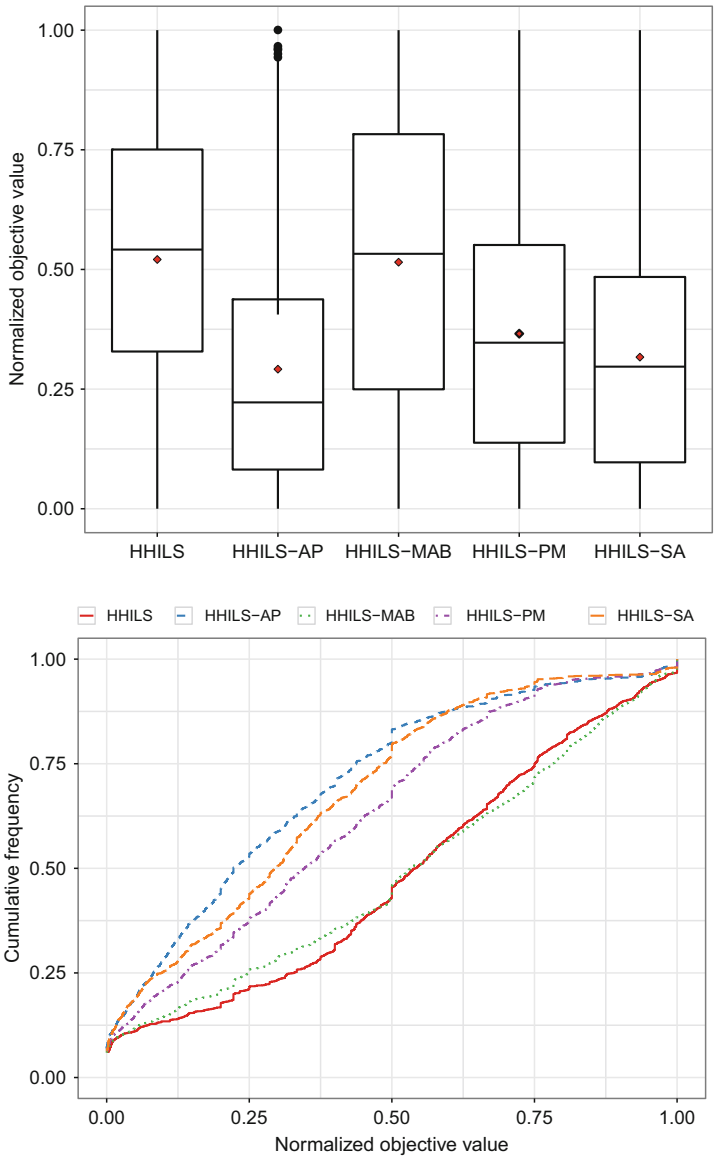

Fig. 2 Summarizing graphical illustrations: (Left) Box-plot graph, (Right) ECDF graph of the normalized objective values of all hyper-heuristics over all problem domains and instances

Comparison to HyFlex State-of-the-Art Hyper-heuristics

In this section, the performance of the HHILS variants is studied and compared against several state-of-the-art hyper-heuristics that participated in the CHeSC 2011

competition. To this end, the same experimental setup has been implemented as provided and described in the website of the CHeSC 2011 competition[1]. Specifically, five problem instances have been chosen (three tests and two hidden instances) for each of the six problem domains, resulting in total 30 different problem instances. Each hyper-heuristic is applied to solve the problem instance at hand for an allowed time period. To calculate the performance of each hyper-heuristic, 31 independent runs are conducted for each problem instance. The median objective value obtained out of these independent runs corresponds to the score of the hyper-heuristic for the specific problem instance at hand. As such, the competition entries are ranked based on their median objective value according to a Formula 1 pointing system. A thorough description of the methodology used for the CHeSC 2011 competition can be found in [98].

Table 8 lists the total and the per-problem domain ranking scores of the CHeSC 2011 competition entries along with the HHILS variants. Specifically, the rows of the table are sorted based on the total score (*total*) of each hyper-heuristic and rank them (*rank*) based on a descending order of their overall score. The remaining columns represent the obtained score of each hyper-heuristic across the different problem domains (for clarity, abbreviations SAT, BP, PS, FS, TSP, and VRP stand for Max-SAT, bin packing, personnel scheduling, flow shop, traveling salesman problem, and vehicle routing with time windows problem domain, respectively).

The scores in Table 8 suggest that the most competitive hyper-heuristics developed in this study, HHILS-SA and HHILS-AP, exhibit very competitive performance compared against CHeSC 2011 entries. HHILS-SA and HHILS-AP hold the second and third position, respectively, on the leaderboard. Notice that the score difference between HHILS-SA/HHILS-AP and the first ranked approach (AdaptHH) is small (9–11 units difference), while there is a large gap between them and the fourth entry (ML), (35–38 units difference). AdaptHH algorithm produces the best performance in two out of the six problem domains (BP and TSP), and it performs really well on the instances used for the FS domain. HHILS-SA exhibits superior performance in the PS and the VRP problem domains and high-performance gains in the SAT problem domain. Similarly, HHILS-AP shows superior performance in the SAT problem domain, while it shows quite competitive scores for the PS, the FS, and the VRP cases.

HHILS-PM is able to exhibit promising performance only in the SAT domain and high scores in the PS and VRP cases, while its contribution to the other domains is minimal. Notice that almost all HHILS variants show superior performance in the SAT domain, with HHILS-AP and HHILS-PM to share the first position in the ranking for this specific domain. In general, it can be observed that the HHILS variants were competitive entries for the SAT, PS, FS, and VRP domains. However, they were not successful in producing a competitive performance, against the other entries, in the BP and TSP cases.

It is worth noting that most of the CHeSC 2011 competitors have a very complex structure that demands a lot of time, effort, and experimentation in the development process, which is a natural process for participating in a competition. However, as suggested by the ranking scores, simple approaches, such as the

Table 8 Total and per problem domain ranking scores of the developed hyper-heuristics as competitors on the CHeSC 2011 competition

Rank	*Algorithm*	*Total*	SAT	BP	PS	FS	TSP	VRP
1	AdaptHH	137.25	8.50	**45.00**	3.50	33.00	**38.25**	9.00
2	HHILS-SA	128.35	31.85	12.00	**37.50**	11.00	4.00	**32.00**
3	HHILS-AP	125.95	**32.95**	8.00	25.50	21.50	10.00	28.00
4	ML	90.50	1.00	6.00	23.50	**35.00**	12.00	13.00
5	VNS-TW	86.75	11.00	2.00	24.50	31.00	16.25	2.00
6	HHILS-PM	78.45	**32.95**	3.00	24.50	0.00	0.00	18.00
7	EPH	68.75	0.00	6.00	3.00	16.50	34.25	9.00
8	PHUNTER	68.75	1.00	3.00	7.50	8.00	24.25	25.00
9	NAHH	63.60	6.60	19.00	0.00	22.00	12.00	4.00
10	HHILS	50.45	29.45	0.00	17.00	0.00	0.00	4.00
11	ISEA	48.00	0.00	28.00	11.00	0.00	9.00	0.00
12	HAEA	36.33	0.00	2.00	0.00	5.33	11.00	18.00
13	HHILS-MAB	35.10	23.10	8.00	0.00	0.00	3.00	1.00
14	ACO-HH	32.33	0.00	18.00	0.00	8.33	6.00	0.00
15	HAHA	19.33	7.00	0.00	7.00	0.33	0.00	5.00
16	KSATS-HH	19.00	3.00	8.00	0.00	0.00	0.00	8.00
17	DynILS	18.00	0.00	7.00	0.00	0.00	11.00	0.00
18	GenHive	16.00	0.00	9.00	0.00	3.00	1.00	3.00
19	XCJ	15.00	0.00	11.00	0.00	0.00	0.00	4.00
20	GISS	10.00	0.00	0.00	4.00	0.00	0.00	6.00
21	AVEG-Nep	9.60	6.60	0.00	0.00	0.00	0.00	3.00
22	SA-ILS	9.50	0.00	0.00	6.50	0.00	0.00	3.00
23	SelfSearch	3.00	0.00	0.00	0.00	0.00	3.00	0.00
24	Ant-Q	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25	MCHH-S	0.00	0.00	0.00	0.00	0.00	0.00	0.00

tutorial demonstration hyper-heuristics in this chapter, might lead to quite efficient search methodologies with competitive performance gains. Notice that several approaches in the class of generation hyper-heuristics try to (semi-)automate the design algorithmic phase in order to alleviate the researcher from having to fine-tune the considered search methodology for a specific (or not) class of problems. Representative examples of automatic algorithmic design can be found in [33, 34, 36].

Conclusions

Hyper-heuristics range over a broad class of search methodologies. Many hyper-heuristic methods can be characterized by their general applicability and robustness across different application domains. The main feature of a hyper-heuristic is that it operates on a space of heuristics rather than on a solution space. Although there has been a significant level of research on hyper-heuristics during the last 15 years or so, several important and challenging research directions in the field are in their infancy and demand further exploration.

This chapter presents a timely and thorough literature review of the main advances in the field of hyper-heuristics since the publication of [36] in 2013. In addition, a simple hyper-heuristic framework has been developed and evaluated as a tutorial-style introduction to the field. The chapter has been divided into three parts. The first part reviews a wide spectrum of advances in the field of hyper-heuristics since 2012 and discusses the current trends and ideas that have been considered by the community. In particular, various studies have been identified that investigate the foundational ideas of the field, consider novel methodologies for automatic design of algorithms and parameter control, provide theoretical analysis, introduce novel hyper-heuristic methodologies and algorithms, develop hyper-heuristic frameworks in multi-objective and dynamic problem formulations, and successfully tackle real-world applications in various fields.

In the second and third parts, a simple but efficient selection hyper-heuristic framework is developed, as a tutorial-style introduction to the field, and evaluates its performance on six problem domains. The hyper-heuristic framework incorporates into the iterated local search algorithm an action selection model to adaptively choose the most promising perturbation heuristic based on feedback during the search process. Various state-of-the-art action selection models have been employed and evaluated. Experimental results and statistical analyses on six different problem domains verify its efficiency and performance. Comparisons with state-of-the-art hyper-heuristics in the field demonstrate its strength despite its relative simplicity.

The literature review highlights various interesting advances and limitations of the current state of the art, which can be summarized as follows.

The current formulation and classification of hyper-heuristics have stimulated the interest of the community. In particular, reformulations and extensions of the definition of hyper-heuristics have been proposed (see, e.g., [128]). A productive criticism on the level of the domain barrier revealed that both high and low levels of abstraction in the problem domain can enhance the search efficiency of hyper-heuristics. Problem domain information is sometimes able to strengthen the search dynamics of a hyper-heuristic without limiting its generality and robustness. Recent advances include unified representations as well as the embedding of mechanisms of common knowledge across different problem domains (see, e.g., [49, 129]).

From the methodological point of view, a variety of novel algorithms and frameworks have recently been published. Heuristic generation methodologies have been developed not only as hyper-visors to generate low-level heuristics

but also as mechanisms to generate high-level models [110]. Moreover, hyper-heuristic frameworks that are able to learn the feature space of a problem class and its representative search strategies have been proposed [57, 120–122]. These frameworks are capable of generating new search strategies and self-adapting their search characteristics based on new problem instances. Furthermore, various analyses have been published that study either the characteristics of the heuristic space, such as its diversity, or the feedback information that can be obtained through the search process (e.g., see [39, 54, 55, 115, 124]). It has also been identified that current developments in the field are mostly empirical, while the theoretical foundations and analyses of hyper-heuristics represent significant open research challenges. A few representative examples of recent theoretical work include the runtime analysis of hyper-heuristic algorithms [8, 74].

A comparison between the fields of meta-learning and hyper-heuristics has been performed that advocates their common objective of automating the algorithm design process [100]. Recent advances in the automatic design of algorithms include various simple, general, and efficient frameworks based on stochastic local search algorithms and novel gramma-based representations of heuristic algorithms among others (e.g., see [6, 80, 85]). Additionally, the commonalities in memetic computing and hyper-heuristics have been reviewed, and new frameworks that combine ideas from both fields have been presented [48]. Various other recent developments in the automated design of algorithms generate efficient algorithms for specific classes of problems, including (among others) packing and cutting problems [116, 132].

A significant number of real-world applications demand high-quality decisions. However, their formulations might include multiple objectives, uncertainty, or dynamically changing environments. Recent trends in the field promote research on both multi-objective and dynamic or uncertain problems. An effective multi-objective algorithm should possess good search characteristics in terms of both solution diversity and convergence on the Pareto-front set. Current developments include multi-objective hyper-heuristic approaches that operate on search strategies to efficiently guide search to diverse Pareto sets that converge to the Pareto front (e.g., see [37, 76, 83, 84, 91, 92, 139]). Current methodologies mostly include selective mechanisms that combine the strengths of predefined search operators or algorithms, while heuristic generation approaches are in their infancy. Clearly, selection hyper-heuristics play a major role in this direction of research. However, the design of a multi-objective algorithm demands expert knowledge of the field, and it is a very demanding and time-consuming development process. Thus, automatic design processes that will efficiently address major issues will arguably have a tremendous impact on the multi-objective community. Hyper-heuristic methodologies for automatically designing and generating multi-objective algorithms are definitely a worthwhile (and growing) research direction.

Hyper-heuristic methodologies that employ adaptivity and cope with dynamically changing problem scenarios have been recently developed to address applications with uncertainty and dynamic changes in time. Most of the developed approaches utilize heuristic selection mechanisms of specialized meta-heuristics

that address dynamic problems across different types of dynamic environments [7, 126, 133]. Exploration of this area is only just beginning.

A major direction that draws upon the interdisciplinary scope of hyper-heuristics is the incorporation of machine learning methodologies to enhance hyper-heuristic efficiency and effectiveness. Machine learning methodologies have been incorporated mostly into the higher level of hyper-heuristics. The main aim of such methodologies is to learn the behavior of the underlying heuristics and be able to make effective decisions to guide the search toward the most promising areas of the search space. Recent advances include methodologies that consider tensor analysis, Inverse Reinforcement Learning theory, Monte Carlo Tree search, and grammatical evolution (among others, e.g., [17–19, 21, 25, 27, 109, 111]). Moreover, hyper-heuristics and the automatic design of algorithms have influenced the machine learning community too. Various developed methodologies include the automatic design of classification algorithms, such as decision trees, to generate more general and robust methodologies for specific class of problems. Clearly, established methodologies in machine learning can potentially boost the strength of hyper-heuristic approaches. As foreseen in [34], it can be recognized that research at the interface of machine learning and hyper-heuristic methodologies has significant potential.

Hyper-heuristics have been successfully applied in a wide variety of diverse application areas. Complex applications that can be modeled as combinatorial optimization problems such as scheduling, timetabling, cutting, and packing problems represent the early adopted application areas of hyper-heuristic approaches. Recent developments not only continue to study these application domains but are also successfully applied to a wide variety of other applications, such as games, engineering, informatics, and parallel and/or distributed applications.

Hyper-heuristic research lies at the interface between operational research and computer science, and it has an impact on a broad spectrum of interdisciplinary application areas. The interdisciplinary character of the field promotes research that combines advanced knowledge from a variety of research areas with the common objective of successfully solving complex real-life optimization problems.

Cross-References

- ▸ Adaptive and Multilevel Metaheuristics
- ▸ Automatic Tuning of Parameters
- ▸ Cutting and Packing
- ▸ Data Mining in Stochastic Local Search
- ▸ Genetic Algorithms
- ▸ Implementation of Metaheuristics
- ▸ Iterated Local Search
- ▸ Matheuristics
- ▸ Memetic Algorithms
- ▸ Multiobjective Optimization

- Network Optimization
- Particle Swarm Methods
- Restart Strategies
- Simulated Annealing
- Tabu Search
- Theoretical Analysis of Stochastic Search Algorithms
- Variable Neighborhood Search
- Vehicle Routing

Acknowledgments Michael G. Epitropakis is supported by a grant from the Engineering and Physical Sciences Research Council (EPSRC Grant No. EP/J017515/1), by a Microsoft Azure Grant 2014, and by a Lancaster University Early Career Internal Grant (A100699). This support is gratefully acknowledged.

References

1. (2011) CHeSC 2011: cross-domain heuristic search challenge. http://www.asap.cs.nott.ac.uk/external/chesc2011/. Accessed 25 Mar 2015
2. (2011) HyFlex competition instance summary. http://www.asap.cs.nott.ac.uk/external/chesc2011/reports/CHeSCInstanceSummary.pdf. Accessed 25 Mar 2015
3. (2014) CHeSC 2014: the second cross-domain heuristic search challenge. http://www.hyflex.org/chesc2014/. Accessed 25 Mar 2015
4. (2014) HyFlex API: hyper-heuristics flexible framework API. http://www.hyflex.org/. Accessed 25 Mar 2015
5. Adriaensen S, Brys T, Nowe A (2014) Designing reusable metaheuristic methods: a semi-automated approach. In: 2014 IEEE congress on evolutionary computation (CEC), pp 2969–2976. https://doi.org/10.1109/CEC.2014.6900575
6. Adriaensen S, Brys T, Nowé A (2014) Fair-share ILS: a simple state-of-the-art iterated local search hyperheuristic. In: Proceedings of the 2014 conference on genetic and evolutionary computation (GECCO'14). ACM, New York, pp 1303–1310. https://doi.org/10.1145/2576768.2598285
7. Akar E, Topcuoglu HR, Ermis M (2014) Hyper-heuristics for online UAV path planning under imperfect information. In: Esparcia-Alcázar AI, Mora AM (eds) Applications of evolutionary computation. Lecture notes in computer science. Springer, Berlin/Heidelberg, pp 741–752
8. Alanazi F, Lehre PK (2014) Runtime analysis of selection hyper-heuristics with classical learning mechanisms. In: 2014 IEEE congress on evolutionary computation (CEC), pp 2515–2523. https://doi.org/10.1109/CEC.2014.6900602, 00000
9. Aleti A, Moser I (2013) Entropy-based adaptive range parameter control for evolutionary algorithms. In: Proceedings of the 15th annual conference on genetic and evolutionary computation (GECCO'13). ACM, New York, pp 1501–1508. https://doi.org/10.1145/2463372.2463560
10. Aleti A, Moser I (2013) Studying feedback mechanisms for adaptive parameter control in evolutionary algorithms. In: 2013 IEEE congress on evolutionary computation (CEC), pp 3117–3124. https://doi.org/10.1109/CEC.2013.6557950
11. Aleti A, Moser I, Meedeniya I, Grunske L (2013) Choosing the appropriate forecasting model for predictive parameter control. Evol Comput 22(2):319–349. https://doi.org/10.1162/EVCO_a_00113
12. Allen J (2014) A framework for hyper-heuristic optimisation of conceptual aircraft structural designs. Doctoral, Durham University

13. Allen JG, Coates G, Trevelyan J (2013) A hyper-heuristic approach to aircraft structural design optimization. Struct Multidiscip Optim 48(4):807–819. https://doi.org/10.1007/s00158-013-0928-3, 00001
14. Anwar K, Awadallah M, Khader A, Al-betar M (2014) Hyper-heuristic approach for solving nurse rostering problem. In: 2014 IEEE symposium on computational intelligence in ensemble learning (CIEL), pp 1–6. https://doi.org/10.1109/CIEL.2014.7015743
15. Anwar K, Khader AT, Al-Betar MA, Awadallah MA (2014) Development on harmony search hyper-heuristic framework for examination timetabling problem. In: Tan Y, Shi Y, Coello CAC (eds) Advances in swarm intelligence. Lecture notes in computer science, vol 8795. Springer International Publishing, Cham, pp 87–95
16. Aron R, Chana I, Abraham A (2015) A hyper-heuristic approach for resource provisioning-based scheduling in grid environment. J Supercomput 1–24. https://doi.org/10.1007/s11227-014-1373-9
17. Asta S, Özcan E (2014) An apprenticeship learning hyper-heuristic for vehicle routing in HyFlex. In: 2014 IEEE symposium on evolving and autonomous learning systems (EALS), pp 65–72. https://doi.org/10.1109/EALS.2014.7009505
18. Asta S, Özcan E (2014) A tensor-based approach to nurse rostering. In: 10th international conference on the practice and theory of automated timetabling (PATAT 2014), pp 442–445
19. Asta S, Özcan E (2015) A tensor-based selection hyper-heuristic for cross-domain heuristic search. Inf Sci 299:412–432. https://doi.org/10.1016/j.ins.2014.12.020
20. Asta S, Özcan E, Parkes AJ (2013) Batched mode hyper-heuristics. In: Nicosia G, Pardalos P (eds) Learning and intelligent optimization. Lecture notes in computer science. Springer, Berlin/Heidelberg, pp 404–409
21. Asta S, Özcan E, Parkes AJ, Etaner-Uyar S A (2013) Generalizing hyper-heuristics via apprenticeship learning. In: Middendorf M, Blum C (eds) Evolutionary computation in combinatorial optimization. Lecture notes in computer science, vol 7832. Springer, Berlin/Heidelberg, pp 169–178
22. Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. Mach Learn 47(2–3):235–256. https://doi.org/10.1023/A:1013689704352, 01559
23. Bäck T, Fogel DB, Michalewicz Z (eds) (1997) Handbook of evolutionary computation. Oxford University Press, New York
24. Banerjea-Brodeur M (2013) Selection hyper-heuristics for healthcare scheduling. PhD thesis, University of Nottingham
25. Barros RC, Basgalupp MP, Carvalho ACPLFd (2014) Investigating fitness functions for a hyper-heuristic evolutionary algorithm in the context of balanced and imbalanced data classification. Genet Program Evolvable Mach 1–41. https://doi.org/10.1007/s10710-014-9235-z
26. Bartz-Beielstein T, Lasarczyk C, Preuss M (2010) The sequential parameter optimization toolbox. In: Bartz-Beielstein T, Chiarandini M, Paquete L, Preuss M (eds) Experimental methods for the analysis of optimization algorithms. Springer, Berlin/Heidelberg, pp 337–362, 00031
27. Basgalupp MP, Barros RC, Barabasz T (2014) A grammatical evolution based hyper-heuristic for the automatic design of split criteria. In: Proceedings of the 2014 conference on genetic and evolutionary computation (GECCO'14). ACM, New York, pp 1311–1318. https://doi.org/10.1145/2576768.2598327
28. Battiti R, Protasi M (2001) Reactive local search for the maximum clique problem. Algorithmica 29(4):610–637
29. Battiti R, Brunato M, Mascia F (2009) Reactive search and intelligent optimization. Operations research/computer science interfaces series, vol 45. Springer, Boston, 00000
30. Boughaci D, Lassouaoui M (2014) Stochastic hyper-heuristic for the winner determination problem in combinatorial auctions. In: Proceedings of the 6th international conference on management of emergent digital EcoSystems (MEDES'14). ACM, New York, pp 11:62–11:66. https://doi.org/10.1145/2668260.2668268

31. Branke J, Hildebrandt T, Scholz-Reiter B (2014) Hyper-heuristic evolution of dispatching rules: a comparison of rule representations. Evol Comput 1–29. https://doi.org/10.1162/EVCO_a_00131
32. Burke EK, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Hyper-heuristics: an emerging direction in modern search technology. In: Glover F, Kochenberger GA (eds) Handbook of metaheuristics. International series in operations research & management science, vol 57. Springer, Boston, pp 457–474
33. Burke EK, Hyde MR, Kendall G, Ochoa G, Özcan E, Woodward JR (2009) Exploring hyper-heuristic methodologies with genetic programming. In: Mumford CL, Jain LC (eds) Computational intelligence. Intelligent systems reference library, vol 1. Springer, Berlin/Heidelberg, pp 177–201
34. Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E, Woodward JR (2010) A classification of hyper-heuristic approaches. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics. International series in operations research & management science, vol 146. Springer, Boston, pp 449–468
35. Burke EK, Qu R, Soghier A (2012) Adaptive selection of heuristics for improving exam timetables. Ann Oper Res 218(1):129–145. https://doi.org/10.1007/s10479-012-1140-3
36. Burke EK, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E, Qu R (2013) Hyper-heuristics: a survey of the state of the art. J Oper Res Soc 64(12):1695–1724. https://doi.org/10.1057/jors.2013.71
37. Castro OR, Pozo A (2014) A MOPSO based on hyper-heuristic to optimize many-objective problems. In: 2014 IEEE symposium on swarm intelligence (SIS), pp 1–8. https://doi.org/10.1109/SIS.2014.7011803
38. Chakhlevitch K, Cowling P (2008) Hyperheuristics: recent developments. In: Cotta C, Sevaux M, Sorensen K (eds) Adaptive and multilevel metaheuristics. Studies in computational intelligence, vol 136. Springer, Berlin/Heidelberg, pp 3–29
39. Consoli PA, Minku LL, Yao X (2014) Dynamic selection of evolutionary algorithm operators based on online learning and fitness landscape metrics. In: Dick G, Browne WN, Whigham P, Zhang M, Bui LT, Ishibuchi H, Jin Y, Li X, Shi Y, Singh P, Tan KC, Tang K (eds) Simulated evolution and learning. Lecture notes in computer science, vol 8886. Springer International Publishing, Cham, pp 359–370, 00000
40. Cowling P, Kendall G, Soubeiga E (2001) A hyperheuristic approach to scheduling a sales summit. In: Burke EK, Erben W (eds) Practice and theory of automated timetabling III. Lecture notes in computer science, vol 2079. Springer, Berlin/Heidelberg, pp 176–190
41. Crowston WBS (1963) Probabilistic and parametric learning combinations of local job shop scheduling rules. Carnegie Institute of Technology and Graduate School of Industrial Administration, Pittsburgh
42. Dong B, Jiao L, Wu J (2015) Graph-based hybrid hyper-heuristic channel scheduling algorithm in multicell networks. Trans Emerg Telecommun Tech n/a–n/a. https://doi.org/10.1002/ett.2923
43. Drake JH, Özcan E, Burke EK (2015) Modified choice function heuristic selection for the multidimensional knapsack problem. In: Sun H, Yang CY, Lin CW, Pan JS, Snasel V, Abraham A (eds) Genetic and evolutionary computing. Advances in intelligent systems and computing, vol 329. Springer International Publishing, Cham, pp 225–234
44. Eiben AE, Smit SK (2011) Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol Comput 1(1):19–31
45. Epitropakis MG, Plagianakos VP, Vrahatis MN (2009) Evolutionary adaptation of the differential evolution control parameters. In: IEEE congress on evolutionary computation (CEC'09), pp 1359–1366
46. Epitropakis MG, Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2012) Tracking differential evolution algorithms: an adaptive approach through multinomial distribution tracking with exponential forgetting. In: Maglogiannis I, Plagianakos V, Vlahavas I (eds) Artificial intelligence: theories and applications. Lecture notes in computer science, vol 7297. Springer, Berlin/Heidelberg, pp 214–222

47. Epitropakis MG, Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2012) Tracking particle swarm optimizers: an adaptive approach through multinomial distribution tracking with exponential forgetting. In: 2012 IEEE congress on evolutionary computation (CEC), pp 1–8
48. Epitropakis MG, Caraffini F, Neri F, Burke EK (2014) A Separability prototype for automatic memes with adaptive operator selection. In: 2014 IEEE symposium on foundations of computational intelligence (FOCI), pp 70–77. https://doi.org/10.1109/FOCI.2014.7007809
49. Feng L, Ong Y, Lim M, Tsang I (2014) Memetic search with inter-domain learning: a realization between CVRP and CARP. IEEE Trans Evol Comput PP(99):1–1. https://doi.org/10.1109/TEVC.2014.2362558
50. Fialho A (2010) Adaptive operator selection for optimization. Ph.D. thesis, Université Paris-Sud XI, Orsay
51. Fialho A, Costa LD, Schoenauer M, Sebag M (2010) Analyzing bandit-based adaptive operator selection mechanisms. Ann Math Artif Intell 60(1-2):25–64. https://doi.org/10.1007/s10472-010-9213-y, 00032
52. Fisher H, Thompson GL (1963) Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF, Thompson GL (eds) Industrial scheduling. Prentice-Hall, Englewood Cliffs, pp 225–251
53. Gong W, Fialho A, Cai Z (2010) Adaptive strategy selection in differential evolution. In: Proceedings of the 12th annual conference on genetic and evolutionary computation (GECCO'10). ACM, New York, pp 409–416
54. Grobler J, Engelbrecht A, Kendall G, Yadavalli VSS (2014) The entity-to-algorithm allocation problem: extending the analysis. In: 2014 IEEE symposium on computational intelligence in ensemble learning (CIEL), pp 1–8. https://doi.org/10.1109/CIEL.2014.7015744
55. Grobler J, Engelbrecht AP, Kendall G, Yadavalli VSS (2015) Heuristic space diversity control for improved meta-hyper-heuristic performance. Inf Sci 300:49–62. https://doi.org/10.1016/j.ins.2014.11.012
56. Güney IA, Küçük G, Özcan E (2013) Hyper-heuristics for performance optimization of simultaneous multithreaded processors. In: Gelenbe E, Lent R (eds) Information sciences and systems 2013. Lecture notes in electrical engineering, vol 264. Springer International Publishing, Cham, pp 97–106, 00001
57. Hart E, Sim K (2014) On the life-long learning capabilities of a NELLI*: a hyper-heuristic optimisation system. In: Bartz-Beielstein T, Branke J, Filipč B, Smith J (eds) Parallel problem solving from nature – PPSN XIII. Lecture notes in computer science, vol 8672. Springer International Publishing, Cham, pp 282–291
58. Hildebrandt T, Goswami D, Freitag M (2014) Large-scale simulation-based optimization of semiconductor dispatching rules. In: Proceedings of the 2014 winter simulation conference (WSC'14). IEEE Press, Piscataway, pp 2580–2590, 00000
59. Hollander M, Wolfe DA, Chicken E (2013) Nonparametric statistical methods, 3rd edn. Wiley, Hoboken
60. Hoos H, Stützle T (2004) Stochastic local search: foundations & applications. Morgan Kaufmann Publishers Inc., San Francisco, 01275
61. Hutter F, Hoos HH, Leyton-Brown K (2011) Sequential model-based optimization for general algorithm configuration. In: Coello CAC (ed) Learning and intelligent optimization. Lecture notes in computer science, vol 6683. Springer, Berlin/Heidelberg, pp 507–523, 00149
62. Jackson WG, Özcan E, John RI (2014) Fuzzy adaptive parameter control of a late acceptance hyper-heuristic. In: 2014 14th UK workshop on computational intelligence (UKCI), pp 1–8. https://doi.org/10.1109/UKCI.2014.6930167, 00000
63. Karafotias G, Hoogendoorn M, Eiben AE (2013) Why parameter control mechanisms should be benchmarked against random variation. In: 2013 IEEE congress on evolutionary computation (CEC), pp 349–355. https://doi.org/10.1109/CEC.2013.6557590

64. Karafotias G, Eiben AE, Hoogendoorn M (2014) Generic parameter control with reinforcement learning. In: Proceedings of the 2014 conference on genetic and evolutionary computation (GECCO'14). ACM, New York, pp 1319–1326. https://doi.org/10.1145/2576768.2598360
65. Karafotias G, Eiben E, Hoogendoorn M (2014) Generic parameter control with reinforcement learning. In: Genetic and evolutionary computation conference (GECCO'14), Vancouver, 12–16 July 2014, pp 1319–1326
66. Karafotias G, Hoogendoorn M, Eiben A (2014) Parameter control in evolutionary algorithms: trends and challenges. IEEE Trans Evol Comput PP(99):1–1
67. Karafotias G, Hoogendoorn M, Eiben AE (2014) Parameter control in evolutionary algorithms: trends and challenges. IEEE Trans Evol Comput PP(99):1–1. https://doi.org/10.1109/TEVC.2014.2308294
68. Kheiri A, Özcan E (2014) Constructing constrained-version of magic squares using selection hyper-heuristics. Comput J 57(3):469–479. https://doi.org/10.1093/comjnl/bxt130, 00001
69. Kheiri A, Özcan E, Parkes AJ (2014) A stochastic local search algorithm with adaptive acceptance for high-school timetabling. Ann Oper Res https://doi.org/10.1007/s10479-014-1660-0
70. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680. https://doi.org/10.1126/science.220.4598.671, 00003
71. Koohestani B, Poli R (2014) Evolving an improved algorithm for envelope reduction using a hyper-heuristic approach. IEEE Trans Evol Comput 18(4):543–558. https://doi.org/10.1109/TEVC.2013.2281512, 00000
72. Koulinas G, Kotsikas L, Anagnostopoulos K (2014) A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. Inf Sci 277:680–693. https://doi.org/10.1016/j.ins.2014.02.155, 00008
73. Lassouaoui M, Boughaci D (2014) A choice function hyper-heuristic for the winner determination problem. In: Terrazas G, Otero FEB, Masegosa AD (eds) Nature inspired cooperative strategies for optimization (NICSO 2013). Studies in computational intelligence, vol 512. Springer International Publishing, Cham, pp 303–314
74. Lehre PK, 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 (FOGA XII'13). ACM, New York, pp 97–104. https://doi.org/10.1145/2460239.2460249, 00008
75. Li D, Li M, Meng X, Tian Y (2015) A hyperheuristic approach for intercell scheduling with single processing machines and batch processing machines. IEEE Trans Syst Man Cybern Syst 45(2):315–325. https://doi.org/10.1109/TSMC.2014.2332443
76. Li K, Fialho A, Kwong S, Zhang Q (2014) Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 18(1):114–130. https://doi.org/10.1109/TEVC.2013.2239648
77. Li S (2013) Hyper-heuristic cooperation based approach for bus driver scheduling. Ph.D. thesis, Université de Technologie de Belfort-Montbeliard
78. Liao X, Li Q, Yang X, Zhang W, Li W (2007) Multiobjective optimization for crash safety design of vehicles using stepwise regression model. Struct Multidiscip Optim 35(6):561–569. https://doi.org/10.1007/s00158-007-0163-x
79. Lobo F, Lima C, Michalewicz Z (eds) (2007) Parameter setting in evolutionary algorithms. Studies in computational intelligence, vol 54. Springer, Berlin/Heidelberg
80. López-Camacho E, Terashima-Marin H, Ross P, Ochoa G (2014) A unified hyper-heuristic framework for solving bin packing problems. Expert Syst Appl 41(15):6876–6889. https://doi.org/10.1016/j.eswa.2014.04.043, 00002
81. López-Ibáñez M, Dubois-Lacoste J, Stützle T, Birattari M (2011) The irace package, iterated race for automatic algorithm configuration. Technical report. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles
82. Lourenço HR, Martin O, Stützle T (2003) Iterated local search, handbook of meta-heuristics. Springer, Berlin/Heidelberg

83. Maashi M, Özcan E, Kendall G (2014) A multi-objective hyper-heuristic based on choice function. Expert Syst Appl 41(9):4475–4493. https://doi.org/10.1016/j.eswa.2013.12.050, 00008
84. Maashi M, Kendall G, Özcan E (2015) Choice function based hyper-heuristics for multi-objective optimization. Appl Soft Comput 28:312–326. https://doi.org/10.1016/j.asoc.2014.12.012
85. Marmion ME, Mascia F, López-Ibáñez M, Stützle T (2013) Automatic design of hybrid stochastic local search algorithms. In: Blesa MJ, Blum C, Festa P, Roli A, Sampels M (eds) Hybrid metaheuristics. Lecture notes in computer science, vol 7919. Springer, Berlin/Heidelberg, pp 144–158
86. Marshall RJ, Johnston M, Zhang M (2014) A comparison between two evolutionary hyper-heuristics for combinatorial optimisation. In: Dick G, Browne WN, Whigham P, Zhang M, Bui LT, Ishibuchi H, Jin Y, Li X, Shi Y, Singh P, Tan KC, Tang K (eds) Simulated evolution and learning, Lecture notes in computer science, vol 8886. Springer International Publishing, Cham, pp 618–630
87. Marshall RJ, Johnston M, Zhang M (2014) Developing a hyper-heuristic using grammatical evolution and the capacitated vehicle routing problem. In: Dick G, Browne WN, Whigham P, Zhang M, Bui LT, Ishibuchi H, Jin Y, Li X, Shi Y, Singh P, Tan KC, Tang K (eds) Simulated evolution and learning, Lecture notes in computer science, vol 8886. Springer International Publishing, Cham, pp 668–679
88. Marshall RJ, Johnston M, Zhang M (2014) Hyper-heuristics, grammatical evolution and the capacitated vehicle routing problem. In: Proceedings of the 2014 conference companion on genetic and evolutionary computation companion (GECCOComp'14). ACM, New York, pp 71–72. https://doi.org/10.1145/2598394.2598407
89. Martin S, Ouelhadj D, Smet P, Vanden Berghe G, Özcan E (2013) Cooperative search for fair nurse rosters. Expert Syst Appl 40(16):6674–6683. https://doi.org/10.1016/j.eswa.2013.06.019
90. Mascia F, López-Ibáñez M, Dubois-Lacoste J, Stützle T (2014) Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. Comput Oper Res 51:190–199. https://doi.org/10.1016/j.cor.2014.05.020
91. McClymont K, Keedwell EC, Savić D, Randall-Smith M (2014) Automated construction of evolutionary algorithm operators for the bi-objective water distribution network design problem using a genetic programming based hyper-heuristic approach. J Hydroinf 16(2):302. https://doi.org/10.2166/hydro.2013.226, 00001
92. McClymont K, Keedwell E, Savic D (2015) An analysis of the interface between evolutionary algorithm operators and problem features for water resources problems. A case study in water distribution network design. Environ Model Softw. https://doi.org/10.1016/j.envsoft.2014.12.023
93. Misir M, Lau HC (2014) Diversity-oriented bi-objective hyper-heuristics for patrol scheduling. In: 10th international conference on the practice and theory of automated timetabling (PATAT 2014)
94. Misir M, Smet P, Vanden Berghe G (2014) An analysis of generalised heuristics for vehicle routing and personnel rostering problems. J Oper Res Soc https://doi.org/10.1057/jors.2014.11
95. Neamatian Monemi R, Danach K, Khalil W, Gelareh S, Lima Jr FC, Aloise DJ (2015) Solution methods for scheduling of heterogeneous parallel machines applied to the workover rig problem. Expert Syst Appl 42(9):4493–4505. https://doi.org/10.1016/j.eswa.2015.01.046
96. Nguyen S, Zhang M, Johnston M, Tan KC (2014) Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Trans Evol Comput 18(2):193–208. https://doi.org/10.1109/TEVC.2013.2248159, 00013
97. Ochoa G, Burke EK (2014) Hyperils: an effective iterated local search hyper-heuristic for combinatorial optimisation. In: 10th international conference on the practice and theory of automated timetabling (PATAT 2014)

98. Ochoa G, Hyde M, Curtois T, Vazquez-Rodriguez JA, Walker J, Gendreau M, Kendall G, McCollum B, Parkes AJ, Petrovic S, Burke EK (2012) HyFlex: a benchmark framework for cross-domain heuristic search. In: Hao JK, Middendorf M (eds) Evolutionary computation in combinatorial optimization. Lecture notes in computer science, vol 7245. Springer, Berlin/Heidelberg, pp 136–147
99. Ong YS, Keane AJ (2004) Meta-Lamarckian learning in memetic algorithms. IEEE Trans Evol Comput 8(2):99–110. https://doi.org/10.1109/TEVC.2003.819944, 00460
100. Pappa GL, Ochoa G, Hyde MR, Freitas AA, Woodward J, Swan J (2013) Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. Genet Program Evolvable Mach 15(1):3–35. https://doi.org/10.1007/s10710-013-9186-9, 00000
101. Park J, Nguyen S, Johnston M, Zhang M (2013) Evolving stochastic dispatching rules for order acceptance and scheduling via genetic programming. In: Cranefield S, Nayak A (eds) AI 2013: advances in artificial intelligence. Lecture notes in computer science, vol 8272. Springer International Publishing, Berlin, pp 478–489, 00001
102. Pillay N (2014) A review of hyper-heuristics for educational timetabling. Ann Oper Res 1–36. https://doi.org/10.1007/s10479-014-1688-1
103. Poli R, Graff M (2009) There is a free lunch for hyper-heuristics, genetic programming and computer scientists. Springer, Berlin/Heidelberg, pp 195–207
104. Qu R, Pham N, Bai R, Kendall G (2014) Hybridising heuristics within an estimation distribution algorithm for examination timetabling. Appl Intell 1–15. https://doi.org/10.1007/s10489-014-0615-0
105. Ren Z, Jiang H, Xuan J, Hu Y, Luo Z (2014) New insights into diversification of hyper-heuristics. IEEE Trans Cybern 44(10):1747–1761. https://doi.org/10.1109/TCYB.2013.2294185, 00004
106. Ross P (2005) Hyper-heuristics. In: Burke EK, Kendall G (eds) Search methodologies, 1st edn. Springer, New York, pp 529–556
107. Ross P (2014) Hyper-heuristics. In: Burke EK, Kendall G (eds) Search methodologies, 2nd edn. Springer, New York, pp 611–638
108. Ryser-Welch P, Miller JF (2014) Plug-and-play hyper-heuristics: an extended formulation. In: 2014 IEEE eighth international conference on self-adaptive and self-organizing systems (SASO), pp 179–180. https://doi.org/10.1109/SASO.2014.33, 00000
109. Sá AGCd, Pappa GL (2014) A hyper-heuristic evolutionary algorithm for learning Bayesian network classifiers. In: Bazzan ALC, Pichara K (eds) Advances in artificial intelligence – IBERAMIA 2014. Lecture notes in computer science. Springer International Publishing, Cham, pp 430–442
110. Sabar N, Ayob M, Kendall G, Qu R (2014) The automatic design of hyper-heuristic framework with gene expression programming for combinatorial optimization problems. IEEE Trans Evol Comput PP(99):1–1. https://doi.org/10.1109/TEVC.2014.2319051
111. Sabar NR, Kendall G (2015) Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems. Inf Sci https://doi.org/10.1016/j.ins.2014.10.045
112. Sabar NR, Ayob M, Kendall G, Qu R (2015) A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. IEEE Trans Cybern 45(2):217–228. https://doi.org/10.1109/TCYB.2014.2323936
113. Salcedo-Sanz S, Matías-Román JM, Jiménez-Fernández S, Portilla-Figueras A, Cuadra L (2013) An evolutionary-based hyper-heuristic approach for the Jawbreaker puzzle. Appl Intell 40(3):404–414. https://doi.org/10.1007/s10489-013-0470-4, 00000
114. Salcedo-Sanz S, Jiménez-Fernández S, Matías-Román JM, Portilla-Figueras JA (2014) An educational software tool to teach hyper-heuristics to engineering students based on the Bubble breaker puzzle. Comput Appl Eng Educ n/a–n/a. https://doi.org/10.1002/cae.21597, 00000
115. Salhi A, Rodríguez JAV (2013) Tailoring hyper-heuristics to specific instances of a scheduling problem using affinity and competence functions. Memetic Comput 6(2):77–84. https://doi.org/10.1007/s12293-013-0121-7, 00000

116. Segredo E, Segura C, León C (2013) Memetic algorithms and hyperheuristics applied to a multiobjectivised two-dimensional packing problem. J Glob Optim 58(4):769–794. https://doi.org/10.1007/s10898-013-0088-4, 00000
117. Segredo E, Segura C, Leon C (2014) Fuzzy logic-controlled diversity-based multi-objective memetic algorithm applied to a frequency assignment problem. Eng Appl Artif Intell 30:199–212. https://doi.org/10.1016/j.engappai.2014.01.005
118. Segredo E, Segura C, Leon C, Hart E (2014) A fuzzy logic controller applied to a diversity-based multi-objective evolutionary algorithm for single-objective optimisation. Soft Comput 1–19. https://doi.org/10.1007/s00500-014-1454-y
119. Segredo E, Segura C, Leon C (2014) Control of numeric and symbolic parameters with a hybrid scheme based on fuzzy logic and hyper-heuristics. In: 2014 IEEE congress on evolutionary computation (CEC), pp 1890–1897. https://doi.org/10.1109/CEC.2014.6900538, 00000
120. Sim K, Hart E (2014) An improved immune inspired hyper-heuristic for combinatorial optimisation problems. In: Proceedings of the 2014 conference on genetic and evolutionary computation (GECCO'14). ACM, New York, pp 121–128. https://doi.org/10.1145/2576768.2598241
121. Sim K, Hart E, Paechter B (2013) Learning to solve bin packing problems with an immune inspired hyper-heuristic. MIT Press, pp 856–863. https://doi.org/10.7551/978-0-262-31709-2-ch126
122. Sim K, Hart E, Paechter B (2014) A lifelong learning hyper-heuristic method for bin packing. Evol Comput 1–31. https://doi.org/10.1162/EVCO_a_00121
123. Smit SK, Eiben AE (2009) Comparing parameter tuning methods for evolutionary algorithms. In: IEEE congress on evolutionary computation (CEC'09), pp 399–406
124. Soria Alcaraz JA, Ochoa G, Carpio M, Puga H (2014) Evolvability metrics in adaptive operator selection. In: Proceedings of the 2014 conference on genetic and evolutionary computation (GECCO'14). ACM, New York, pp 1327–1334. https://doi.org/10.1145/2576768.2598220, 00001
125. Soria-Alcaraz JA, Ochoa G, Swan J, Carpio M, Puga H, Burke EK (2014) Effective learning hyper-heuristics for the course timetabling problem. Eur J Oper Res 238(1):77–86. https://doi.org/10.1016/j.ejor.2014.03.046, 00006
126. van der Stockt S, Engelbrecht AP (2014) Analysis of hyper-heuristic performance in different dynamic environments. In: 2014 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE), pp 1–8. https://doi.org/10.1109/CIDUE.2014.7007860
127. Sutton RS, Barto AG (1998) Introduction to reinforcement learning, 1st edn. MIT Press, Cambridge, 02767
128. Swan J, Woodward J, Özcan E, Kendall G, Burke EK (2013) Searching the hyper-heuristic design space. Cogn Comput 6(1):66–73. https://doi.org/10.1007/s12559-013-9201-8, 00000
129. Swiercz A, Burke EK, Cichenski M, Pawlak G, Petrovic S, Zurkowski T, Blazewicz J (2013) Unified encoding for hyper-heuristics with application to bioinformatics. CEJOR 22(3):567–589. https://doi.org/10.1007/s10100-013-0321-8, 00000
130. Thierens D (2005) An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 7th annual conference on genetic and evolutionary computation (GECCO'05). ACM, New York, pp 1539–1546
131. Thierens D (2007) Adaptive strategies for operator allocation. In: Lobo F, Lima C, Michalewicz Z (eds) Parameter setting in evolutionary algorithms. Studies in computational intelligence, vol 54. Springer, Berlin/Heidelberg, pp 77–90, 00042
132. Thomas J, Chaudhari NS (2014) Design of efficient packing system using genetic algorithm based on hyper heuristic approach. Adv Eng Softw 73:45–52. https://doi.org/10.1016/j.advengsoft.2014.03.003, 00000
133. Topcuoglu HR, Ucar A, Altin L (2014) A hyper-heuristic based framework for dynamic optimization problems. Appl Soft Comput 19:236–251. https://doi.org/10.1016/j.asoc.2014.01.037, 00003

134. Tsai CW, Huang WC, Chiang MH, Chiang MC, Yang CS (2014) A hyper-heuristic scheduling algorithm for cloud. IEEE Trans Cloud Comput 2(2):236–250. https://doi.org/10.1109/TCC.2014.2315797, 00003
135. Urra E, Cubillos C, Cabrera-Paniagua D (2015) A hyperheuristic for the dial-a-ride problem with time windows. Math Probl Eng 2015:e707056. https://doi.org/10.1155/2015/707056, 00000
136. Xie J, Mei Y, Ernst AT, Li X, Song A (2014) A genetic programming-based hyper-heuristic approach for storage location assignment problem. In: 2014 IEEE congress on evolutionary computation (CEC), pp 3000–3007. https://doi.org/10.1109/CEC.2014.6900604
137. Yarimcam A, Asta S, Özcan E, Parkes AJ (2014) Heuristic generation via parameter tuning for online bin packing. In: 2014 IEEE symposium on evolving and autonomous learning systems (EALS), pp 102–108. https://doi.org/10.1109/EALS.2014.7009510
138. Yin PY, Chuang KH, Hwang GJ (2014) Developing a context-aware ubiquitous learning system based on a hyper-heuristic approach by taking real-world constraints into account. Univ Access Inf Soc 1–14. https://doi.org/10.1007/s10209-014-0390-z
139. Yuen SY, Zhang X (2014) Multiobjective evolutionary algorithm portfolio: choosing suitable algorithm for multiobjective optimization problem. In: 2014 IEEE congress on evolutionary computation (CEC), pp 1967–1973. https://doi.org/10.1109/CEC.2014.6900470, 00000
140. Zheng YJ, Zhang MX, Ling HF, Chen SY (2015) Emergency railway transportation planning using a hyper-heuristic approach. IEEE Trans Intell Transp Syst 16(1):321–329. https://doi.org/10.1109/TITS.2014.2331239