Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Hyper-heuristics: A survey and taxonomy

Tansel Dokeroglu[a], Tayfun Kucukyilmaz[b,*], El-Ghazali Talbi[c]

[a] *Software Engineering Department, Cankaya University, Ankara, Turkey*
[b] *Department of Technology and Operations Management, Erasmus University, Netherlands*
[c] *University of Lille, CNRS UMR 9189, Centre de Recherche en Informatique Signal et Automatique de Lille (CRIStAL), Lille F-59000, France*

ARTICLE INFO

Keywords:
Hyper-heuristics
Metaheuristics
Survey
Optimization

ABSTRACT

Hyper-heuristics are search techniques for selecting, generating, and sequencing (meta)-heuristics to solve challenging optimization problems. They differ from traditional (meta)-heuristics methods, which primarily employ search space-based optimization strategies. Due to the remarkable performance of hyper-heuristics in multi-objective and machine learning-based optimization, there has been an increasing interest in this field. With a fresh perspective, our work extends the current taxonomy and presents an overview of the most significant hyper-heuristic studies of the last two decades. Four categories under which we analyze hyper-heuristics are selection hyper-heuristics (including machine learning techniques), low-level heuristics, target optimization problems, and parallel hyper-heuristics. Future research prospects, trends, and prospective fields of study are also explored.

1. Introduction

Hyper-heuristics have evolved as a means to increase the generality of search and optimization algorithms. The term “hyper-heuristics” was first mentioned by studies in the 1960s (Crowston et al., 1963). Today, there is an ongoing interest in this field, leading to an increasing number of high-quality studies. As of May 2023, when the keyword “*hyper-heuristic*” is searched, more than 9,660 and 1,003 studies are reported on Google Scholar and Web of Science websites, respectively. Hyper-heuristics work on top of a set of heuristics to select, generate, and sequence the best problem-specific heuristics to solve a particular problem. They are distinct methodologies developed to solve hard optimization problems. In order to be able to use a metaheuristic for solving a specific issue, domain knowledge is necessary. However, hyper-heuristics are innovative rather than attempting to solve the problem optimally. The algorithms look for the most appropriate approach or sequence of heuristics. They can determine and apply the most effective heuristic to address a specific issue. The flowchart of a hyper-heuristic can be seen in Fig. 1.

As new heuristics and (meta)-heuristics are being proposed daily due to widespread interest, many new opportunities for hyper-heuristics emerge, capturing the attention of research communities.

Hyper-heuristics generally use lower-level heuristics and/or metaheuristics by selecting/generating the most suitable (meta)-heuristics rather than traversing the search space itself, as this can be performed

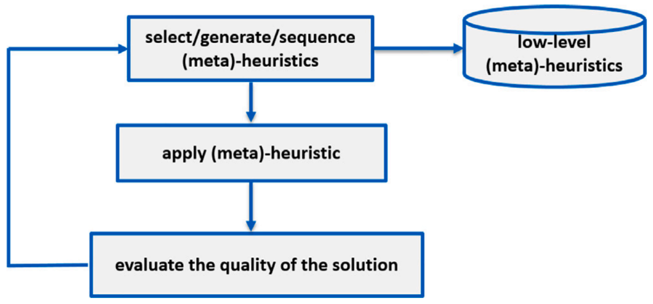
select/generate/sequence (meta)-heuristics
low-level (meta)-heuristics
apply (meta)-heuristic
evaluate the quality of the solution

Fig. 1. The flowchart of the hyper-heuristic algorithms using selection/generation of heuristics.

more efficiently by a lower-level heuristic and/or a metaheuristic algorithm. Unlike traditional hyper-heuristics, recent state-of-the-art hyper-heuristic studies are mostly inclined to solve multi-objective optimization, machine-learning-supported techniques, and parallel computing. We have decided that it is necessary to extend the existing taxonomy of hyper-heuristics by preparing a new survey, considering these new issues that have not been mentioned in previous surveys. Our study is the first to categorize the most influential hyper-heuristic studies in the last two decades with a new approach. We examine the hyper-heuristics under four new headings: selection hyper-heuristics,

* Corresponding author.
E-mail address: kucukyilmaz@rsm.nl (T. Kucukyilmaz).

https://doi.org/10.1016/j.cie.2023.109815
Received 18 November 2022; Received in revised form 11 October 2023; Accepted 2 December 2023
Available online 4 December 2023
0360-8352/© 2023 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

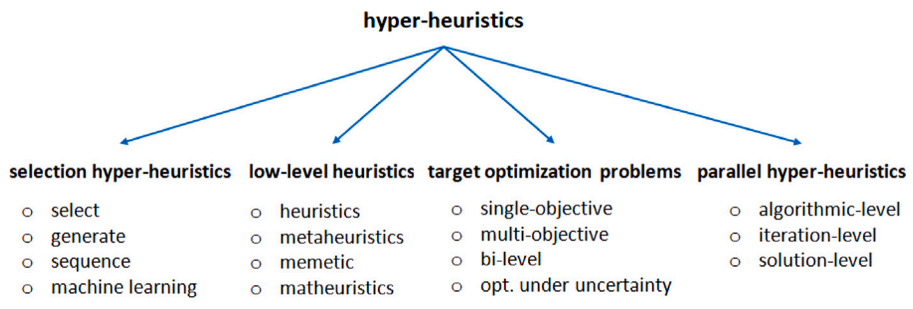

Fig. 2. Our proposed classification of hyper-heuristics.

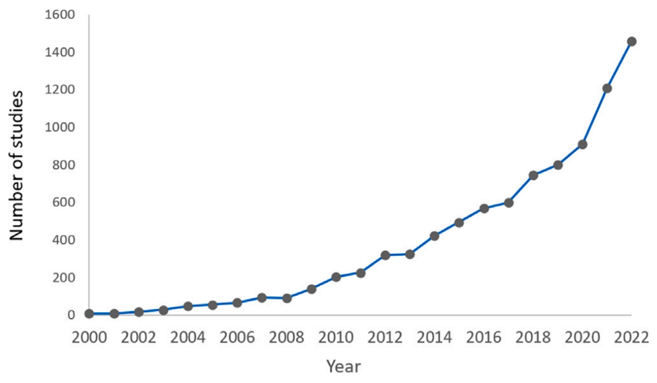

Fig. 3. The increasing number of hyper-heuristic studies between 2000 and 2021 (Google scholar).

low-level heuristics, target optimization problems, and parallel hyper-heuristics (see Fig. 2 for the details of the proposed classification).

Wolpert and Macready (1997) prepared a set of rules called No Free Lunch (NFL) theorems in 1997 to illustrate the components of information-theoretic optimization and performance benchmarking. According to these theorems, every improvement in one class of jobs is offset by an improvement in another. If algorithm *A* can outperform algorithm *B* on a specific problem, then algorithm *B* can outperform algorithm *A* on other problem sets. Numerous search problems, their geometry, and information-theoretic aspects are investigated in this study. This strategy allows researchers to develop mathematical rules for evaluating the performance of a certain search method. The NFL theorem indicates hyper-heuristics have a stronger theoretical foundation than heuristic and metaheuristic algorithms. Therefore, the NFL theorem is an important theoretical criterion to justify the performance of hyper-heuristics

Fig. 3 gives the number of academic studies on hyper-heuristics using Google Scholar from 2000 until 2022. It is observed that the number of studies has been increasing at a higher rate with every passing year. After 2017, the increase in studies becomes more visible, and this tendency is likely to continue in the upcoming years. Unlike previous hyper-heuristic surveys written by the same researchers repeatedly, we think our survey brings a different perspective to the developments in the hyper-heuristics of the last two decades. More than 150 recent papers are inspected in our survey.

The rest of this study is organized as follows: Section 2 gives brief information about recent reviews, state-of-the-art studies, and frameworks related to hyper-heuristics. Section 3 gives detailed information about the hyper-heuristics of our extended classification regarding four subsections: selection hyper-heuristics, low-level heuristics, target optimization problems, and parallel hyper-heuristics. The last section presents our concluding remarks, new research areas, and future work.

2. Previous surveys, state-of-the-art studies and frameworks of hyper-heuristic algorithms

This section gives detailed information about previous surveys/reviews and proposed frameworks of hyper-heuristics (see Table 1) in the literature in chronological order.

Burke et al. (2003) introduced the main concepts of hyper-heuristics, briefly explained these algorithms, and reviewed the latest studies in the field. Özcan, Bilgin, and Korkmaz (2008) investigated the necessity for domain expertise while using a metaheuristic to solve problems. The authors proposed hyper-heuristics as a new search optimization strategy. A new hyper-heuristic technique function was proposed on top of a series of heuristics. In their work, hyper-heuristics are subjected to a thorough examination. The best technique was consistently compared to genetic and memetic algorithms. New frameworks of hyper-heuristics were also explored to cast doubt on the concept of issue independence. Burke et al. (2009) addressed genetic programming as the most employed approach. The methods required to use this methodology, several example case studies, and a discussion of important concerns were covered in depth. Their presentation aimed to illustrate this novel technique's tremendous possibilities for automating the heuristic design process.

Ouelhadj and Petrovic (2010) studied the low-level heuristics' role of interaction and cooperation. The authors developed a population of heuristic agents, called "*cooperative agents*", as a framework of hyper-heuristics. In their model, the cooperative agent uses a set of low-level heuristic solutions to decide the best heuristics. The proposed cooperative framework outperformed sequential hyper-heuristics in studies conducted on flow shop benchmarks. Burke, Hyde, et al. (2010) gave a categorization of hyper-heuristics and a description that encompasses the current research on this topic. The authors divided hyper-heuristics into selection and creation categories. Each category was described in length with several exemplary cases. The authors explained the key aspects of existing methodologies and proposed new avenues for researching hyper-heuristics. Burke, Curtois, et al. (2010) studied permutation flow shop, bin packing, and scheduling problems (with Hyper-heuristics Flexible framework, HyFlex). Durillo and Nebro (2011) developed jMetal, a hyper-heuristic for multi-objective optimization problems. jMetal uses state-of-the-art optimization tools. The framework presents a user-friendly graphical interface and statistical values of the results using multi-core processors.

Kheiri, Özcan, and Parkes (2012) surveyed the well-known hyper-heuristics of high school scheduling problems. In their experiments, the authors searched for the best heuristics. Burke et al. (2013) analyzed the selection hyper-heuristics and explored recent frameworks. The survey included research trends and future works. Furthermore, the current taxonomy of selection hyper-heuristics was reported. Pappa et al. (2014) studied automated algorithm design and similarities between supervised machine learning and hyper-heuristics. The authors discussed research directions and challenges in meta-learning and hyper-heuristics. Burke, Burke, Kendall, and Kendall (2014) prepared a tutorial for implementing hyper-heuristics and discussed relevant research issues in application domains. The study gives short hyper-heuristics with examples. Ryser-Welch and Miller (2014) examined novel frameworks for hyper-heuristics and the field's general background. This research adds to current reviews and gives the scholarly community a different viewpoint on this important and emerging topic. The authors pointed out that some frameworks are heavily limited, remarking on the difficulty of performing better than state-of-the-art algorithms. Sabar, Ayob, Kendall, and Qu (2014a) offered a gene expression programming approach for generating the hyper-heuristic framework's high-level heuristic throughout the instance-solving process. The created heuristic takes information from the current issue and chooses the best low-level heuristic. The suggested hyper-universality heuristics are tested using the HyFlex framework (Ochoa et al., 2012) to solve well-known combinatorial optimization problems. According to empirical data, the proposed technique performs well for multiple cases across many domains.

López-Camacho, Terashima-Marin, Ross, and Ochoa (2014) developed a hyper-heuristic framework that uses a deterministic technique to produce solutions for packing and cutting problems. The framework can solve one and two-dimensional issues involving irregular concave polygons. Thousands of problems with concave polygons are used to validate the technique. Branke, Nguyen, Pickardt, and Zhang (2015) investigated the topic of production scheduling. The authors provided a comprehensive overview of the design choices and crucial challenges that arise throughout the development of such systems. The authors outlined state-of-the-art techniques and provided a taxonomy and suggestions for creating hyper-heuristics in scheduling algorithms. The challenges, open questions, and numerous avenues for further research are mentioned.

Elhag and Özcan (2015) presented a new selection hyper-heuristic framework for solving grouping problems using grouping low-level heuristics and a move acceptance mechanism. High-quality solutions are retained, reflecting the trade-off between the number of groups and the extra target for the specific grouping issue. The move acceptance mechanism incorporates a local search strategy capable of advancing improvements on such trade-off solutions. The performance of selection hyper-heuristics is examined on graph coloring as a grouping problem. The experimental findings demonstrate the framework's usefulness in grouping hyper-heuristics to create high-quality answers.

Pillay (2016) introduced hyper-heuristics for educational scheduling and its research areas to give a generic solution. Kheiri and Özcan (2016) studied designing an adaptive heuristic selection method to improve the performance of a selection hyper-heuristic. They described an iterated multi-stage hyper-heuristic that iterates through interacting hyper-heuristics. This study is based on the principle that not all low-level heuristics are useful for the search process. Gómez and Coello (2017) presented a hyper-heuristic for continuous search spaces that combined various scalarizing functions to generate Pareto optimal solutions in multi-objective optimization problems. The selection of scalarizing functions is guided by the s-energy measure, which evaluates the distribution of points in k-dimensional manifolds. The proposed approach outperforms other state-of-the-art algorithms in the majority of global test problems. Pillay and Qu (2018) investigated recent techniques to solve real-world optimization problems. Theory, fundamentals, and applications of hyper-heuristics are analyzed in biology, optimization, and operations research. The paper by Choong, Wong, and Lim (2018) discusses the automation of hyper-heuristic methodologies for solving optimization problems. Traditional models use a high-level heuristic with heuristic selection and move acceptance components. The article proposes an automatic design method using reinforcement learning, specifically Q-learning, to guide the selection of components. Extensive evaluations on benchmark instances from six problem domains show the method's comparability to top-performing hyper-heuristic models in the existing literature.

Burke et al. (2019) overviewed categorizations of hyper-heuristics and provided a unified classification. They distinguished between heuristic selection and generation. Some examples are discussed, and recent trends are highlighted. In their study, Abd Elaziz and Mirjalili (2019) enhanced the Whale Optimization Algorithm (WOA) by introducing a hyper-heuristic called DEWCO. This approach automatically selects a suitable chaotic map and a portion of the population using the Differential Evolution (DE) algorithm. DEWCO improves exploration and avoids local optima in WOA, as demonstrated through experiments on 35 standard CEC2005 functions using seven algorithms. The results confirm the superior performance of DEWCO in determining optimal solutions for test function problems. In a study by Hao, Qu, and Liu (2020) highlights the growing interest in hyper-heuristics, which aim to provide generalized solutions to optimization problems by searching in a higher-level space of heuristics. The article proposes a unified framework called evolutionary multitasking graph-based hyper-heuristic (EMHH) that combines the advantages of knowledge transfer and cross-domain optimization from evolutionary multitasking with heuristic search in hyper-heuristics. The effectiveness and generality of the EMHH framework are demonstrated through experiments on exam timetabling and graph-coloring problems, showing improved efficiency compared to single-tasking hyper-heuristics.

Nesi and da Rosa Righi (2020) demonstrated a hyper-heuristic that employs stochastic automata networks with learning to govern a series of meta-heuristics. This research looks at the search space motions as examined by heuristic methods. The approach works with eight meta-heuristics. The findings illustrated the effectiveness of the proposed hyper-heuristic framework. An efficient hyper-heuristic capable of selecting and setting the parameters of low-level heuristics is developed. Drake, Kheiri, Ozcan, and Burke (2020) analyzed current selection hyper-heuristics and described new frameworks of hyper-heuristics. The current taxonomy of selection hyper-heuristics was expanded to include the present research issues. This study concentrated on selection hyper-heuristics and included open questions. Sánchez et al. (2020) studied hyper-heuristics proposed for solving the most challenging combinatorial optimization problems. In their framework, the authors studied well-known combinatorial optimization problems. The authors identified problem domains that might assist hyper-heuristics to have a more significant impact. Mısır (2021) gave an overview of the shortcomings and recommendations for future hyper-heuristic studies. In a recent study, Duflo, Danoy, Talbi, and Bouvry (2022) prepared a framework of hyper-heuristics with Q-Learning, Hyper-Heuristic Framework (QLHHF) that reduces any problem to a graph-based problem. Therefore, many (multi-objective and multi-agent) optimization problems can be modeled, requiring only the problem's parameters. Karimi-Mamaghan, Mohammadi, Meyer, Karimi-Mamaghan, and Talbi (2022) prepared a review on machine learning in metaheuristics. This review has many new ideas about the selection of heuristics, fitness evaluation (surrogate, approximate fitness), population initialization, evolution, parameter tuning, and cooperation of individuals. Most of these topics are related to the design of hyper-heuristics. The authors defined a new taxonomy and a technical discussion on future research directions.

Li, Özcan, Drake, and Maashi (2023) introduced existing learning automata-based multiobjective hyper-heuristic. The authors investigated the generality of the proposed method and another learning automata-based selection hyper-heuristic for multiobjective evolutionary algorithms.

As a novel contribution, this survey differs from previous surveys regarding categorization. We reveal a new and extended classification of hyper-heuristics. The best practices of selection hyper-heuristics, low-level heuristics, target optimization problems, and parallel hyper-heuristics of the last twenty years are listed and examined in detail.

Table 1 presents the list of well-known hyper-heuristic frameworks (and the problem domains).

Table 1
The list of well-known hyper-heuristic frameworks.

Study	Name	Problem domain
Burke, Curtois, et al. (2010)	HyFlex	flow shop, bin packing, scheduling
Van Onsem and Demoen (2013)	ParHyFlex	maximum satisfiability
Durillo and Nebro (2011)	jMetal	(multi-objective) generic
Swan, Özcan, and Kendall (2011)	Hyperion	generic
Urra, Cabrera-Paniagua, and Cubillos (2013)	hMod	generic
Bleuler, Laumanns, Thiele, and Zitzler (2003)	PISA	(multi-objective) generic
Cora, Uyar, and Etaner-Uyar (2013)	HH-DSL	generic
Xu, Hutter, Hoos, and Leyton-Brown (2008)	SATzilla	satisfiability problems
Pillay and Beckedahl (2017)	EvoHyp	generic
Cruz-Duarte, Amaya, Ortiz-Bayliss, Terashima-Marín, and Shi (2020)	CUSTOMHyS	generic
Cruz-Duarte, Ortiz-Bayliss, and Amaya (2022)	MatHH	job shop scheduling

3. Extended taxonomy and recent applications of hyper-heuristics

This section examines the most cited/valuable state-of-the-art hyper-heuristics published in the best journals and conferences for the last twenty years. According to our classification, we give the details of applied selection hyper-heuristics, low-level heuristics, target optimization problems, and parallel hyper-heuristics. It should be noted that in this study, we intentionally defer from using the term *taxonomy*, as some of the hyper-heuristics can be labeled with more than a single category. For example, a low-level heuristic can also be a member of parallel hyper-heuristics.

3.1. Selection hyper-heuristics

The main goal of selection hyper-heuristics is to find the best technique or sequence of heuristics that will use a generic method to produce solutions with appropriate quality. Low-level heuristics are used by the selection heuristic while performing these activities. The selection hyper-heuristics perform selection, generation, and sequencing on low-level heuristics (the layer that focuses on optimizing the problem). Sequencing the (meta)-heuristics, i.e., selecting a sequence of heuristics though a single solution process, can be considered a different category than the selection and generation of the (meta)-heuristics because this process can provide additional means to compensate the downsides of each heuristic in the sequence rather than relying on the performance of a single heuristic on each run.

A selection hyper-heuristic handles a set of low-level heuristics to select the best one using performance measures for low-level heuristics. The stages of hyper-heuristics include the selection of heuristics and move acceptance strategies. The main purpose of selection hyper-heuristics is to manage this process according to the success of low-level heuristics.

The selection or generation of heuristics has always been one of the most important issues of this domain since the introduction of hyper-heuristics. Fig. 4 presents an overview of the selection and low-level categories of the proposed hyper-heuristics' taxonomy. The machine learning component learns (or a learning layer might not be used) from the results of the examined heuristics and suggests new methods. The hyper-heuristic layer uses a selection strategy to select the best method depending on the information coming from the machine learning layer. This process is usually an iterative task to accept the candidate solution.

Cowling, Kendall, and Soubeiga (2000) introduced a choice function for heuristic selection in hyper-heuristics. Since then, the choice function has been a well-known method, maintaining a score for each low-level heuristic. A low-level heuristic's score is determined by whether the heuristic provides improvement when employed alone, in combination with other heuristics, or how much effort has been invested. The success of the choice function is verified by many studies using different move acceptance methods on various problems.

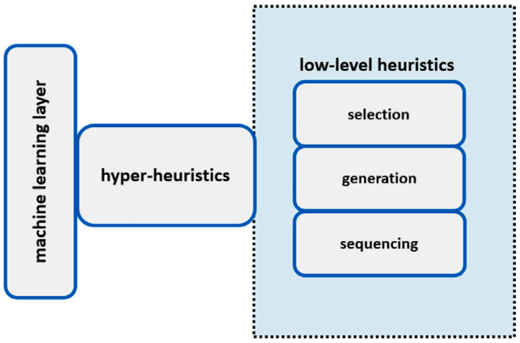

Fig. 4. The overview of selection and low-level categories of the proposed hyper-heuristics' taxonomy.

In order to design better hyper-heuristics, the selection heuristics level should have as much information as possible about the problem domain. Limited information is required in classical hyper-heuristic frameworks (the number of low-level heuristics, the type of optimization problem, and the objective function) (Kendall et al., 2002).

Branke et al. (2015) gave a heuristic-generation review with a focus on the design of construction heuristics. The authors developed the representation of heuristics, the optimization algorithm, and the fitness value function. The authors classified hyper-heuristics based on learning methods. Pillay (2016) provided an overview of constructive hyper-heuristics for educational timetabling with selection and generation of constructive/perturbative hyper-heuristics. Interestingly, Kotthoff (2016) addressed the selection hyper-heuristic issues from the perspective of Operations Research and prepared a survey to determine the most suitable algorithm for solving a problem. He showed significant performance improvements in algorithm selection techniques, closely related to our survey.

Ross, Schulenburg, Marín-Bläzquez, and Hart (2002) proposed to generate a solution process applicable to many problem instances using a hyper-heuristic to remove the drawbacks of the evolutionary algorithms. The authors addressed the long execution times and no guarantee of finding the optimal solutions. Simple non-evolutionary heuristics were applied, and the proposed method was successfully tested on 1D bin packing problems. Bilgin, Özcan, and Korkmaz (2006) analyzed common heuristic selection techniques and moved acceptance methods of hyper-heuristics. Various methods were examined on exam timetabling problem instances. The authors implemented many heuristic selection techniques and acceptance criteria. Their study on 21 exam timetabling problem instances comprehensively compared these methods.

Iterative selection hyper-heuristics use different operators to perform single-point search strategies, heuristic selection, and move acceptance. In order to improve performance, learning mechanisms are introduced by recent studies. For example, Burke, Kendall, Mısır, and Özcan (2012) proposed Monte Carlo-based hyper-heuristics. One of the selection hyper-heuristics works with the simulated annealing move acceptance method. This method has a significant improvement in the experiments. Experiments revealed that the choice function is the best alternative for simulated annealing.

To develop better scheduling for cloud computing, Tsai, Huang, Chiang, Chiang, and Yang (2014) introduced a new hyper-heuristic (HHSA). The authors selected low-level strategies to locate superior candidate solutions. The results were compared with some cutting-edge scheduling methods. The findings demonstrated that the HHSA could dramatically shorten the make-span of task scheduling on Hadoop.

Metaheuristics are also being used as selection hyper-heuristics. Koulinas, Kotsikas, and Anagnostopoulos (2014) suggested a hyper-heuristic using the well-known metaheuristic Particle Swarm Optimization (PSO) for resource-restricted project planning. The proposed hyper-heuristic manages low-level heuristics. The method was evaluated using a collection of typical issue situations from the PSPLIB library and contrasted with other methods in the literature. The outcomes confirmed the efficacy of the suggested strategy. Chen, Li, Yang, and Rudolph (2015) studied the energy optimization problems on a quantum-inspired learning approach. The rapid solution assessment improved search speed, and the suggested technique had higher convergence performance. Chen, Ding, Qin, and Zhang (2021) proposed a hyper-heuristic with genetic programming for solving project scheduling problems with Simulated Annealing (to avoid local optima). Shao, Shao, and Pi (2022) suggested a selection hyper-heuristic framework for a flow-shop scheduling problem. The algorithm included high and low-level heuristics. The findings revealed that the algorithm outperformed the Gurobi solver substantially.

Reward-based selection heuristics are important components of hyper-heuristics. Sabar, Ayob, Kendall, and Qu (2014b) developed a new reward-based heuristic selection technique. A gene expression programming was proposed for the acceptance criteria. The results verified that the framework could generalize well for different domains. In a recent study, Lara-Cárdenas et al. (2020) studied the Job-Shop Scheduling Problem. The authors intended to create hyper-heuristics using reinforcement and unsupervised learning. The solution employs a feature space clustering algorithm via a reward-based system.

Tree-based and hierarchical selection heuristics can be seen in hyper-heuristics. Sabar and Kendall (2015) suggested a Monte Carlo tree search to determine the best heuristic sequence. The authors link it with a memory system that uses various population update algorithms. The test suites for the hyper-heuristic competition were used to illustrate the generality of the framework. The outcomes showed that the suggested method generalizes successfully across all six domains and produces competitive results. Guerriero and Saccomanno (2022) tackled the 2D irregular bin packing problem. A dynamic hierarchical strategy was provided to handle the problem. The low-level heuristics were chosen according to the primary characteristics of the case to be solved. The hyper-heuristic can execute simple or recursive heuristics. A comparison with cutting-edge techniques is also performed. The computational findings were promising. Ochoa et al. (2021) proposed machine learning techniques to speed up the generation of algorithm selection strategies to improve reproducibility and modularity. The techniques are implemented on a domain-independent module. The authors produced four new algorithm selectors for constraint satisfaction problems.

Machine learning-based hyper-heuristics can combine the advantages of low-level heuristics when they are used as a high-level selection strategy after learning about the characteristics of optimization methods. Reinforcement learning, Q-learning, and deep learning are the most used machine learning methods in the literature that achieve significant improvements compared to single metaheuristics.

Özcan, Misir, Ochoa, and Burke (2012) developed a hyper-heuristic with a great deluge on examination timetabling problems using heuristic selection methods. Hunt, Neshatian, and Zhang (2012) suggested a hyper-heuristic for the feature selection problem using genetic programming. Using some fundamental components, the suggested technique generates new heuristics. The study developed heuristic functions as new search techniques that can explore the subsets of features. Classifier performance was then increased by employing tiny subsets of features discovered through evolving heuristics. Maashi, Özcan, and Kendall (2014) presented a selection function to solve multi-objective problems. This high-level technique controlled and combined the low-level heuristics, which regulates and combines the advantages of multi-objective evolutionary algorithms. The performance was investigated on the Walking Fish Group test suite benchmark (Meneghini, Alves, Gaspar-Cunha, & Guimaraes, 2020). Furthermore, the hyper-heuristic is tested on the automobile crashworthiness design problem.

Deep learning (LeCun, Bengio, & Hinton, 2015) extracts hidden patterns, and the advantages of low-level heuristics are combined. The proposed algorithm outperformed existing metaheuristic algorithms and conventional methods on large-scale problems. Abd Elaziz and Mirjalili (2019) improved the Whale Optimization Algorithm (WOA) by developing a hyper-heuristic to mitigate the disadvantages of employing the Differential Evolution (DE) (Price, 2013) to select a chaotic map and a subset of the population.

Qin, Zhuang, Huang, and Huang (2021) investigated a vehicle routing problem with varying capacities serving customers while minimizing the maximum service time. For this purpose, the study developed a deep reinforcement learning hyper-heuristic. In their method, metaheuristics were considered low-level, whereas reinforcement learning was used as a high-level heuristic selection strategy. Lassouaoui, Boughaci, and Benhamou (2022) developed a novel approach for the feature selection problem using a selection hyper-heuristic with Thompson sampling. The method employed reinforcement learning to evaluate low-level heuristics and predict the most efficient one. The proposed method was used in conjunction with a single nearest neighbor classifier. It reports the best subset of features for maximizing accuracy. Experiments show that the proposed method can perform better than many conventional classifiers compared to existing approaches. Lin, Li, and Song (2022) presented a hyper-heuristic with Q-learning (QHH) to address the final testing scheduling of semiconductors. The Q-learning algorithm is used to select low-level heuristics. The selected heuristic is performed on the solution space of the optimization process. The results verify the high performance of the QHH method.

Juntama, Delahaye, Chaimatanan, and Alam (2022) addressed a strategic planning problem based on linear dynamical systems to minimize traffic complexity. A hyper-heuristic framework based on reinforcement learning was employed to improve the searching capacity. The approach reduced air traffic complexity by 92.8%, according to the results. Zhang, Bai, Qu, Tu, and Jin (2022) created a hyper-heuristic framework using deep reinforcement to deal with various levels of uncertainty. The approach had a data-driven heuristic selection component with low-level heuristics to improve uncertainties while optimizing many problems (see Fig. 5 for the deep reinforcement learning hyper-heuristic architecture (deep RL)). The actions are low-level heuristics, and the selection of the actions is based on the historical experience of the deep RL agent. Venske, Almeida, Lüders, and Delgado (2022) developed a Multi-objective Evolutionary Algorithm (MOEA) and Differential Evolution for the QAP up to ten objectives. The proposed algorithm provides information to find the best crossover operator, and according to the results, it outperforms the other methods with classical crossover operators.

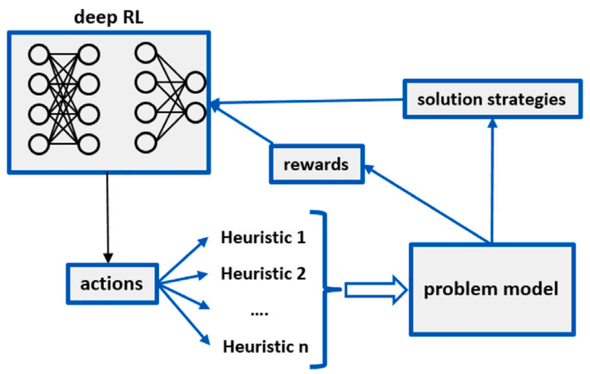

Fig. 5. The deep reinforcement learning-based hyper-heuristic architecture.

3.2. Low-level heuristics

Low-level heuristics are mostly simple methods for visiting the neighboring solutions with swap, add, and delete operators and local search methods. However, our survey extends this category according to the new studies that use simple heuristics, metaheuristics, memetic algorithms, and matheuristics. State-of-the-art approaches and the best practices of the low-level heuristic applications are explained with summaries of recent papers.

3.2.1. Heuristics

The heuristics we have examined in this category are simple methods, local search techniques (visiting neighboring solutions using swap, add, and delete techniques) and they cannot be called metaheuristics or their derivatives.

Ouelhadj and Petrovic (2010) looked at how low-level heuristics might cooperate. A set of heuristics and a cooperative hyper-heuristic algorithm made up their framework. By sharing the results, the heuristics interact with each other. The hyper-heuristic used a set of low-level heuristics for the general selection and the exchange of solutions.

Hybrid heuristics mostly give better results than single heuristics (Blum, Puchinger, Raidl, & Roli, 2011). Therefore, these methods have been used successfully in recent studies. Sabar, Ayob, Qu, and Kendall (2012) researched how to solve exam scheduling issues using a novel graph coloring constructive hyper-heuristic. The authors use four low-level graph coloring methods (largest, saturation, largest colored, and largest enrollment degrees) that have been hierarchically hybridized. The suggested method was evaluated using the 2007 International Timetabling Competition datasets. Results show that constructive hyper-heuristic graph coloring gives better solutions than other procedures.

Routing/scheduling problems is another field where heuristic-based hyper-heuristics are applied frequently. For resolving challenging cases of the dynamic vehicle routing issue, Garrido and Riff (2010) devised and examined an evolution-based hyper-heuristic strategy. Perturbative, constructive, and noise were low-level heuristics used in their study. The heuristics are used complexly to create a new hyper-heuristic algorithm.

Pickardt, Hildebrandt, Branke, Heger, and Scholz-Reiter (2013) developed one of the first examples of hyper-heuristics that combine heuristic generation and selection in their study. The authors proposed a new hyper-heuristic for the generation of efficient dispatching rules. The method uses an evolutionary algorithm to search for a good assignment of rules. The results demonstrated that hyper-heuristics could generate rules with lower mean weighted tardiness than benchmark rules. The two-stage algorithm outperformed the evolutionary algorithm hyper-heuristic.

Ahmed, Mumford, and Kheiri (2019) examined selection hyper-heuristics for the bus network route design problem to minimize the passenger journey time and operator expenses. Combining a sequence-based selection with the great flood acceptance technique yielded greater outcomes in shorter run times than the best-known options. Timetabling and scheduling problems are staple problems for examining the performance of heuristic-based hyper-heuristics in the literature, presenting numerous works. For example, Olgun, Koç, and Altıparmak (2021) investigated the minimization of fuel consumption costs. The authors developed a hyper-heuristic with variable neighborhood heuristics and iterative local search. According to the findings, the green objective function significantly impacted total consumption costs. Compared to other state-of-the-art heuristics, the algorithm can produce competitive results.

3.2.2. Metaheuristics

Metaheuristics are state-of-the-art methods for solving NP-Hard optimization problems (Boussaïd, Lepagnot, & Siarry, 2013; Dokeroglu, Sevinc, Kucukyilmaz, & Cosar, 2019; Talbi, 2009). Selecting the most appropriate metaheuristic from a pool of solutions often provides better solutions than working with simple heuristics. These features make them a valuable solution. In addition to this, hybrid metaheuristics that combine various metaheuristics and exact algorithms are known to provide excellent solutions (Blum et al., 2011). In this part of our survey, metaheuristics are considered low-level heuristics of hyper-heuristic algorithms, and their advantages and disadvantages are examined.

A study on the past literature indicates that there are three main categories of strategies where metaheuristics are utilized as hyper-heuristics: solver selection out of a pool of metaheuristics, using metaheuristics parameter selection, and using metaheuristics for candidate solution generation.

Using metaheuristic algorithms for tuning parameters of complex problems in the presence of extensive number of heuristic- or domain-dependent parameters, or when these parameters do not require domain-specific knowledge, is accepted as an attractive solution in many studies. Cutillas-Lozano, Giménez, and Almeida (2015) used the unified parameterized scheme to implement hyper-heuristics on top of parameterized metaheuristics to select the best set of parameters. The authors verified the proposed hyper-heuristic to solve computational optimization problems (the power consumption in wells, the determination of the chemical reaction kinetic constants and the maximum diversity problem) using Greedy Randomized Adaptive Search Process (GRASP) (Feo & Resende, 1995), tabu search, genetic algorithms and Scatter Search. The hyper-heuristic algorithm provides satisfying results for the metaheuristic with the selected parametrization. Navajas-Guerrero, Manjarres, Portillo, and Landa-Torres (2022) designed a hyper-heuristic to set the time series parameters to enable failure prediction. In their study, the authors use harmony search and feature-based machine learning algorithms for optimizing failure prediction parameters such as window size, time series-related features, and thresholds. Abd Aziz (2015) created a hyper-heuristic technique based on Ant Colony Optimization (ACO) (Dorigo, Birattari, & Stutzle, 2006) for the traveling salesman problem. The optimization focused on automated tuning for non-domain-specific parametrization setups. Two novel pheromone updating procedures were also included in the proposed hyper-heuristic.

Some hyper-heuristics also adopt metaheuristics as low-level solution strategies. Cruz-Duarte et al. (2021) proposed a strategy using a single-solution metaheuristic that has adopted ten different search techniques while using simulated annealing to set the parameters of metaheuristics. In their experiments, the authors validated the results on one hundred-seven continuous problems having 50 dimensions. Gölcük and Ozsoydan (2021) proposed a recommendation architecture with Q-learning to decide the best algorithm from metaheuristics, Manta Ray Foraging Optimization, Artificial Bee Colony (ABC), Whale Optimization (WOA), and Salp Swarm (SS). The hyper-heuristic selects the optimizer using a comparative judgment strategy. The performances were assessed using an experimental analysis that included appropriate statistical tests on knapsack problems. Tabataba and Mousavi (2012) proposed a beam search metaheuristic-based hyper-heuristic algorithm for the longest common subsequence (LCS) problem. The algorithm was tested on real-life biological sequences. Experiments showed that the hyper-heuristic could perform better than the state-of-the-art approaches in the literature.

Some hyper-heuristic approaches in the literature also examine the problem of achieving a high-quality initial solution set to reduce the search space complexity, especially when the optimization problems involve a high level of complexity. Pandiri and Singh (2018) addressed the k-interconnected multi-depot multi-traveling salesman problem. Experimental results showed that the approach could outperform other state-of-the-art methods.

3.2.3. *Memetic algorithms*

The evolutionary algorithms implemented with local search methods are called memetic algorithms (Knowles & Corne, 2000; Krasnogor & Smith, 2005). The memetic algorithms are based on the notion of a meme, which is a unit of cultural development capable of local improvement. The memetic algorithms use local refinement, perturbation, or constructive methods. Recently, there have been some studies to integrate these state-of-the-art algorithms into hyper-heuristic methods. It is seen that very impressive results have been obtained in this regard. This section presents brief information about hyper-heuristics developed with memetic algorithms.

In the most intuitive sense, most memetic algorithms are evolutionary algorithms where local search is handled separately via the feedback from other algorithms. Wang and Tang (2017) studied the permutation flowshop scheduling problem using a genetic algorithm supported by machine learning techniques for speeding up and improving local search. In the proposed method, while the evolutionary algorithm processes the solution instances, they are associated with an archive where all generated solution is stored for further processing and machine learning techniques are applied for selecting high-quality but non-dominating solutions that may have merit. With a similar notion, jing Wang and Wang (2022) presented a solution for a distributed flow-shop scheduling problem. In their setting, the authors examined the job sequencing and machine assignment problems simultaneously and proposed heuristic decomposition techniques for solving the problem. The proposed strategy is to record the overall search history and further examine non-dominant solutions via heuristics based on their Pareto distributions. Although the study does not involve an evolutionary algorithm, the framework proposes two simple assembly scheduling and local intensification heuristics that involve stochastic local search operations very reminiscent of the mutation operators in an evolutionary algorithm.

Pereira, Ritt, and Vásquez (2018) proposed using memetic optimization for a robotic assembly line balancing problem. In their problem definition, the authors consider the assembly line balancing problem and equipment scheduling decisions together and propose a memetic approach that is based on a genetic algorithm. In their approach, a solution for both with and without uncertainty is generated where the problem without uncertainty is solvable with dynamic programming while the case with uncertainties is NP-hard. Leite, Fernandes, Melício, and Rosa (2018) examined the challenging exam timetabling problem due to the existence of deterministic hard constraints. The authors propose using a novel cellular evolutionary algorithm for generating solutions while enhancing the local searches using a Threshold Annealing heuristic over feasible sub-optimal solutions.

Lei, Gong, Jiao, and Zuo (2015) studied the examination timetabling problem. The authors applied the evolutionary strategy and tabu search (variable neighborhoods) to improve global search performance. The evolutionary operators (crossover and mutation) are implemented to obtain appropriate heuristics. Experiments conducted with benchmark instances demonstrated that the algorithm could outperform the other approaches. Beyaz, Dokeroglu, and Cosar (2015) studied the well-known offline 2D bin packing problem (2DBPP). The authors developed new hyper-heuristics that select/combine heuristics and local search methods to minimize the number of bins. The hyper-heuristics use the first fit, next fit, best fit, tabu search, genetic and memetic algorithms. The authors concluded that the new hyper-heuristics could obtain most of the optimal solutions to problem instances.

Drake, Özcan, and Burke (2016) investigated how the crossover is controlled at the hyper-heuristic level where no specific information about the problem exists. The proposed framework is compared to other recent hyper-heuristic frameworks using knapsack problems. In this problem domain, controlling crossover at the domain level outperforms managing crossover at the hyper-heuristic level. Ersoy, Ozcan, and Etaner-Uyar (2007) designed a new hyper-heuristic method that uses multiple hill climbers in a memetic algorithm. The hyper-heuristic is customized for the memetic algorithm and makes greater use of each hill climber's power without modifying the memetic algorithm's framework. The proposed hyper hill-climber algorithm is validated on exam timetabling problems.

Özcan (2006) presented an empirical study of memetic algorithms. The experiments using a collection of well-known benchmark functions with a memetic algorithm are discussed in detail. In the study, adaptive heuristics are classified as hyper-heuristics. Memetic algorithms that run a single hill climber independently are tested on a set of nurse rostering problem cases.

3.2.4. *Matheuristics*

Matheuristics is an emerging optimization research area that focuses on the cooperation of metaheuristics and mathematical programming techniques (Fischetti & Fischetti, 2018; Maniezzo, Stützle, & Voß, 2021). Although there are few hyper-heuristic applications with matheuristics, it is an exciting field. New studies have been proposed to develop hyper-heuristics using matheuristics recently. Some studies performed with Mixed Integer Linear Programs (MILPs) exist.

In a recent study, Archetti and Speranza (2014) prepared a survey on matheuristics that solve vehicle routing problems. The authors proposed classification with three matheuristics classes; decomposition, improvement heuristics and branch-and-price/column generation. A structured overview of the ideas that combine heuristics and mathematical programming models is offered.

Gonzalez, López-Espín, Aparicio, and Talbi (2022) focused on MILPs, and the authors proposed a new hyper-matheuristic for a MILP-based decomposition. An evolutionary algorithm, tabu search, scatter search, and GRASP metaheuristics are used in the proposed algorithm. The experimental results showed that the solutions outperformed those obtained by a metaheuristic. The best algorithm selection is obtained for a set of cooperative metaheuristics.

Steenson, Özcan, Kheiri, McCollum, and McMullan (2022) investigated the multi-level hyper-heuristics arranging low-level heuristics using the Constraint Programming (CP) Solver that performs local search metaheuristics and hill climbing. The authors focused on exam and course scheduling problems. The proposed method used a matrix containing transitional probabilities between low-level heuristics. The best hyper-heuristic for selection was examined for the timetabling problem. The findings showed that the proposed method could yield better solutions than other proposed solvers and CP Solver. In another study, Vela, Cruz-Duarte, Ortiz-Bayliss, and Amaya (2022) developed an extra layer of generalization to select an appropriate solver. The new model outperformed the base hyper-heuristics. In a recent study, Doerner and Schmid (2010) examined the evolution of hybrid solution strategies for several vehicle routing challenges. Only hybrid heuristics and exact solution strategies are taken into account. Most extant hybrid methods rely on set-covering formulations, local branching, or other decomposition strategies. Amaya, Cruz-Duarte, and Ortiz-Bayliss (2022) presented MatHH algorithm, a Matlab-based application that allows rapid hyper-heuristics prototyping. The authors provide an overview of the architecture and some examples of its usage. Interesting research questions that can be examined with MatHH are discussed in this study.

Matheuristics are commonly used in routing problems. Kramer, Subramanian, Vidal, and Lucídio dos Anjos (2015) prepared a Vehicle Routing Problem with environmental aspects to minimize the costs of driver wages and fuel consumption while respecting time windows and capacity constraints. The authors proposed a method using a local search-based metaheuristic with an integer programming approach. The proposed algorithm reports new best solutions for all tested problems.

3.3. *Target optimization problems*

In this section, we examine the application areas of hyper-heuristics as a target optimization problem in more detail. We explained this section under four main subsections (single-objective, multi-objective, bi-level optimization problems, and optimization under uncertainty problems).

3.3.1. *Single-objective problems*

Previous survey sections mainly discussed single-objective algorithms without mentioning this categorization as a target optimization problem. An optimization problem is a single-objective optimization problem if it minimizes or maximizes a single target value. We see that hyper-heuristics have a very wide application area in this field. Some of the single-objective problems that have been solved efficiently by the hyper-heuristics are Exam scheduling, vehicle routing, generating efficient dispatching rules, bus network route design, minimization of fuel consumption costs, determination of the chemical reaction kinetic constants, finding the longest common sub-sequence problem, traveling salesman problem, examination timetabling problem, 2D bin packing problem, knapsack problems, timetabling problem, resource-restricted project planning (Pour, Drake, & Burke, 2018), energy optimization problems, project scheduling problems, quadratic assignment (Venske et al., 2022) etc. As seen, single-objective optimization problem applications of hyper-heuristic algorithms are quite extensive. In our classification, bi-level optimization under uncertainty, parallel applications, low-level heuristics, and selection hyper-heuristics categorizations present several studies of recent single-objective problems.

Most recent studies have reported that hyper-heuristics obtained better results than metaheuristics or other state-of-the-art optimization approaches. Many hyper-heuristic studies for the solutions to single-objective problems are presented in other sections of our survey (they will not be mentioned in this section again).

3.3.2. *Multi-objective problems*

This section focuses on new multi-objective hyper-heuristic research. Many researchers believe that applying hyper-heuristics to multi-objective problems is an appealing alternative. This has become one of the most extensively researched topics in recent years. We compiled a list of the most cited and cutting-edge multi-objective hyper-heuristics studies. When we consider a multi-objective hyper-heuristic, we refer to a problem with two or more objectives.

Burke, Silva, and Soubeiga (2005) offered a novel hyper-heuristic approach with a tabu search to pick the best heuristic for optimizing a particular individual objective. They examined scheduling and space allocation problem instances. The outcomes demonstrated that the performance of the multi-objective hyper-heuristic strategy is superior to techniques in the literature. Gómez and Terashima-Marín (2012) built multi-objective hyper-heuristics using NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002) to solve 2D irregular cutting stock problems. Instead of using a single heuristic consistently throughout the placement process, this variation applied a heuristic according to the state of the problem. Maashi et al. (2014) presented a learning selection hyper-heuristic. This method combined and controlled the advantages of three multi-objective metaheuristic algorithms. The Walking Fish Group was used for the performance analysis of the proposed learning hyper-heuristic. In another study, Maashi, Kendall, and Özcan (2015) presented a study combining a selection heuristic with the great deluge and late acceptance. The Walking Fish Group tests were used during the experiments. The results show that the method was effective for various multi-objective problems.

Hitomi and Selva (2015) investigated various combinations of credit aggregation, definition, and heuristic selection procedures. Instead of traditional benchmarking tasks, all algorithms were evaluated on a climate monitoring satellite constellation design issue. A multi-objective hyper-heuristic method for the integration and test order problems (Assunção, Colanzi, Vergilio, & Pozo, 2014) was developed by Guizzo, Fritsche, Vergilio, and Pozo (2015). The algorithm used selection functions to choose the best low-level heuristic to apply to the problem. The authors suggested a quality metric to evaluate the effectiveness of the selection process. The process used the NSGA-II algorithm, and the proposed hyper-heuristic reported the best results in the literature. Gonçalves, Kuk, Almeida, and Venske (2015) enhanced the MOEA/D framework for developing a multi-objective selection hyper-heuristic. They evaluated MOEA/D-HH on scenarios from the competition problem set. The comparison of MOEA/D-HH with various significant multi-objective optimization techniques yields encouraging results.

Walker and Keedwell (2016) combined different comparison operators and examined them in conjunction with a hyper-heuristic to optimize many-objective problems. They found that the favor relation or hypervolume yields the best results. Kumari and Srinivas (2016) proposed a hyper-heuristic algorithm to cluster the software modules into clusters according to their cohesion and coupling values. They reported significant improvements in their study. Qian, Tang, and Zhou (2016) deployed MOEA components (selection, mutation, and acceptance procedures) as low-level heuristics, showing that heuristic selection might be quicker than a single low-level heuristic. The findings give a theoretical basis for multi-objective selection hyper-heuristics. A multi-objective wind farm layout optimization is addressed by Li, Özcan, and John (2017). They used a hyper-heuristic to select low-level (meta)-heuristics that work on combined solutions controlled by selection hyper-heuristics. The empirical findings demonstrate the efficacy of tackling this challenging computational issue. Guizzo, Vergilio, Pozo, and Fritsche (2017) suggested a hyper-heuristic solution for the Integration and Test Order Problem, which uses selection techniques to pick search operators adaptively while MOEAs are conducted. The results of the experiments reveal that the suggested genetic algorithm outperforms standard MOEAs.

Yao, Peng, and Xiao (2018) proposed a hyper-heuristic framework for route planning problems. To generate new routes, they created a collection of low-level heuristics. Additionally, they used a reinforcement learning process to pick effective low-level heuristics and speed up searches. Extensive testing demonstrated that the suggested method outperforms the exact multi-objective optimization approach regarding speed. Zhang, Mei, and Zhang (2019) dealt with the multi-objective job shop scheduling. They provided various trade-offs between competing objectives. In terms of both training performance and generalization, experimental data demonstrated that the NSGA-II technique implemented in Genetic Programming Hyper-Heuristic (GPHH) outperforms both the weighted sum approaches and SPEA2-based (Zitzler, Laumanns, & Thiele, 2001) GPHH. Zhou, Yang, and Zheng (2019) focused on the multi-objective flexible job shop scheduling using real-life data originating from an aero-engine blade production facility. To evolve online scheduling rules, new multi-agent hyper-heuristics are presented. In their technique, agents were connected with the past data of the shop floor.

Almeida, Gonçalves, Venske, Lüders, and Delgado (2020) studied numerous hyper-heuristics for the multi-objective permutation flow shop problem. On 220 case studies having two and three objectives, the effectiveness of the suggested techniques was evaluated using the indicator of hypervolume and non-parametric tests. Results demonstrate that this best strategy was comparable with the state-of-the-art in situations with two and three objectives, including Pareto and decomposition techniques. Alshareef and Maashi (2022) employed a multi-objective hyper-heuristic technique to address the module clustering issue to assure high modularization quality. The experimental findings showed that the proposed technique outperformed the separate multi-objective evolutionary methods.

de Santiago Júnior, Özcan, and de Carvalho (2020) introduced a selection hyper-heuristic that combines Reinforcement Learning (Kaelbling, Littman, & Moore, 1996), (meta)-heuristic selection, and group decision-making. They provided two iterations of the late acceptance technique and a brand-new quality indicator supporting the initialization of selection hyper-heuristics with little computational expense. Duflo, Danoy, Talbi, and Bouvry (2020) proposed a Q-Learning-based hyper-heuristic for generating distributed coverage of connected-unmanned aerial vehicles swarm heuristics to automate the design of unmanned aerial vehicle swarming behaviors by developing a multi-objective optimization problem. Experimental results verified the capacity of the proposed algorithm to generate heuristics for the problem instances.

de Carvalho, Özcan, and Sichman (2021) presented a novel cross-domain evaluation method for multi-objective optimization that examines hyper-heuristics for real-world optimization problems with multiple criteria needing to be optimized in conjunction. The results showed that hyper-heuristics could outperform single metaheuristics in many situations. Cheng, Tang, Zhang, and Zhang (2022) studied production scheduling problems, and the authors developed a multi-objective hyper-heuristic using Q-learning. Pareto criteria were implemented to provide diversity during the heuristic selection process, where a reward mechanism selects an optimizer for the exploration and exploitation stages.

Maashi et al. (2015) proposed a selection hyper-heuristics using a choice function with great deluge. The metric of hypervolume is employed in the move acceptance techniques. The algorithm's performance is tested with the Walking Fish Group test suite. The results demonstrated the performance of the new move acceptance method in selection hyper-heuristics.

3.3.3. *Bi-level problems*

In a bi-level optimization problem, one optimization problem is embedded (nested) within another optimization problem (Talbi, 2013). In this regard, bi-level optimization is applied to problems with a hierarchical nature where multiple decisions have to be taken one after the other. Often in such problems, the higher-level decisions (termed as "upper-level decisions") are constraining the outcome of the lower-level decisions. Combinatorial bi-level optimization problems are especially challenging when the lower level is characterized by some NP-Hard complexity since evaluation depends on the higher-level decisions. Studies that produce solutions using hyper-heuristics in bi-level optimization have attracted more attention in recent years. This area of research appears to be filled with many new opportunities that may interest researchers.

Yang, Zhong, Dessouky, and Postolache (2018) examined simultaneous scheduling problems in automated control terminals where automated guided vehicles and cranes need to work in tandem to manage operations. Their study involves two integrated problems: a scheduling problem and a vehicle routing problem. To minimize the completion time of operations, the authors propose a bi-level model where the former problem is addressed with a congestion prevention rule-based genetic algorithm and the latter problem is addressed with a rolling horizon heuristic. On a similar topic, Shouwen, Di, Zhengrong, and Dong (2021) examined the joint problem of integrated scheduling and vehicle routing problems to minimize the temporal costs of the operations and proposed using an adaptive heuristic for the vehicle routing problem and a genetic algorithm for solving the integrated scheduling problem.

In his thesis, Turky (2019) studied the selection of local search algorithms for combinatorial optimization problems. The performance of the local search algorithms is closely related to the operators and parameters that should be set. The author proposed a bi-level hyper-heuristic algorithm with various operators for solving combinatorial optimization problems. The search spaces are designed as bi-level heuristic areas, and new hyper-heuristics are introduced to solve this problem. The hyper-heuristics use a local search to select the local search algorithm, and the algorithms are based on ant colonies that combine many local search techniques. Bin packing and machine reassignment problems are examined during the experiments. The proposed frameworks obtained the best-known methods in the literature.

Kieffer, Danoy, Brust, Bouvry, and Nagih (2019) tackled large-scale combinatorial bi-level problems using Genetic Programming hyper-heuristics. This approach trains heuristics as in a machine learning algorithm. The authors considered heuristic generation. Greedy heuristics are trained to make them more reliable for the optimization problem. The authors solved a bi-level cloud computing pricing optimization between a service provider and customers. Results demonstrated that the trained heuristics could cope with the bi-level optimization problems, successfully outperforming classical heuristics and metaheuristics.

3.3.4. *Optimization under uncertainty*

As practical applications of forecasting, decision-making, and process systems engineering problems in real-life conditions often involve uncertainties, deterministic optimization techniques could only produce sub-optimal or infeasible solutions. Motivated by such practical concerns, optimization techniques focusing on uncertain problems attracted much attention within the last decade. As several past surveys also indicate (De Maio, Laganà, Musmanno, & Vocaturo, 2021; Ning & You, 2019), it is possible to categorize past studies on optimization under uncertainty under three categories: stochastic programming (Laporte, Musmanno, & Vocaturo, 2010; Wen, Linde, Ropke, Mirchandani, & Larsen, 2016; Zhang et al., 2022), chance constrained optimization (Oyebolu, Allmendinger, Farid, & Branke, 2019; Renaud, Absi, & Feillet, 2017) and robust optimization techniques (Amini, Moghaddam, & Ebrahimnejad, 2020; Jiang, Han, Liu, & Liu, 2008; Löfberg, 2008; van der Weide, Deng, & Santos, 2022). Although numerous studies are involved in each category, we only focus on hyper-heuristics involving uncertainty in this part of the study.

The core difference between these categories is how uncertainties are modeled within the proposed studies. In the stochastic programming approach, the uncertainty is modeled with probability distributions, explaining possible randomness in problem parameters. Laporte et al. (2010) propose a novel solution to the waste collection problem by modeling a stochastic version of arc routing problems. In their model, edges in the waste collection graph have an associated probability of failing. In their case, the route can be extended via a visit to an intermediate facility, i.e., a dump site. The authors propose adapting a neighborhood-based search heuristic for considering uncertainty using a constructive stochastic path scanning algorithm. The algorithm is reminiscent of evolutionary approaches in that insertion and update heuristics are applied iteratively until a feasible solution is achieved. More recently, Wen et al. (2016) applied a similar Adaptive Large Neighborhood Search heuristic (ALNS) for the multi-objective electric vehicle scheduling problem where the objective is to minimize the number of required vehicles to fulfill trips and minimize the total traveling distance. The proposed approach uses an iterative strategy that starts from an initial solution and refines it by involving diversification techniques for obtaining a versatile set of solutions and a probabilistic selection mechanism for the best solution for the ALNS.

Unlike stochastic programming, chance-constrained optimization aims to solve the optimization problem while ensuring that the constraints are mostly satisfied within an acceptable margin of uncertainty. Renaud et al. (2017) proposed a solution for the meter reading problem where geographically placed nodes in a map contain information to be collected by a vehicle. The problem is posed as finding the minimal tour between sites. The non-determinism of the problem comes from the fact that transmission of the information is probabilistic and is proportional to the distance between the node and the vehicle at their closest distance within the tour. In their ILP-based heuristic solution, the authors impose a set of probabilistic constraints that impose that the probability of transmission for each node must be over some threshold. Oyebolu et al. (2019) examined the problem of optimizing process run times in a multi-product assembly line with possible process failure events and uncertain demand in the pharmaceutical sector. In the study, the authors propose using evolutionary hyper-heuristics, where optimization algorithms evaluate the performance of scheduling policies using an event simulation framework.

Robust optimization models uncertainty as an additional set of parameters representing possible uncertainty relations. Jiang et al. (2008) examined the nonlinear interval number programming problem where only the bounds of the uncertain coefficients are known and not the probability distributions or membership functions. In this study, an order relation of interval number is used to generate two separate deterministic objectives, simultaneously representing the median and boundaries of uncertainties from the uncertain objective function. The proposed approach is based on an evolutionary algorithm, micro GA, which is reportedly capable of avoiding premature convergence and leverages a small population size for computational requirements. Two numerical examples are investigated to demonstrate the effectiveness of the present method. Another study (Löfberg, 2008), proposes using robust optimization for problems involving control and systems theory problems. For many such problems, a convenient strategy aims to model and convert uncertain components in the problem space to a robust counterpart containing no uncertainties. As a wide range of solutions or tools is available for problem models without uncertainties, it is relatively easy to solve the problem next. In their work, the authors present a robust optimization framework in the modeling language YALMIP, which automates the process of uncertainty elimination, allowing the user to concentrate on the high-level model instead. Amini et al. (2020) examined a robust location-arc routing problem with uncertain demand and traversing costs. In this study, the authors employed three meta-heuristic approaches: hill climbing, late acceptance hill climbing, and tabu search for enhancing solution quality. van der Weide et al. (2022) analyzed a scheduling problem for aircraft maintenance checks where sub-steps of maintenance operations may involve temporal deviations, resulting in a need for frequent schedule adjustments. The authors model the problem as a min–max optimization approach and apply a genetic algorithm-based technique to find robust solutions.

3.4. Parallel hyper-heuristics

Modern parallel computation facilities offer many new opportunities for hyper-heuristics algorithms. The fact that many (meta)-heuristics can be run concurrently ensures that the results are obtained in shorter times. Moreover, because the early convergence and stagnation to local optima are handled better, the quality of solutions will also improve. This section gives detailed information about the recent parallel hyper-heuristic studies.

High-performance computing architectures are rapidly emerging, and parallel machine classification significantly impacts the performance and implementation of parallel hyper-heuristics. This area has critical aspects of shared or distributed memory, heterogeneous machines, cloud computing, and the communication network. A global memory links the processors in a shared memory architecture, whereas each processor in a distributed memory architecture has its local memory. Network topologies such as stars, rings, trees, and hypercubes connect the processors. Communication is handled through message passing. These architectures are more scalable than shared memory machines.

The uniformity of processors and networks is critical for parallel hyper-heuristic performance. Local-area networks (LANs) and wide-area networks (WANs) are loosely coupled and used in grid systems with distributed heterogeneous machines in multiple domains. Parallel hyper-heuristics can use modern embedded systems such as Graphical Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs) and ARM processors (Moreno, Ortega, Filatovas, Martínez, & Garzón, 2017).

The parallel architecture affects the implementation of parallel hyper-heuristics. Shared and distributed memory are the popular programming paradigms. Multi-threading, OpenMP and CUDA are the shared-memory programming languages, whereas MPI (Message Passing Interface) is for distributed memory architectures.

The taxonomy of parallel hyper-heuristics presented in our survey is inspired by the study of Talbi (2019). In our taxonomy, we describe three main models.

3.4.1. Algorithmic-level parallelism

This model executes different (meta)-heuristic algorithms in parallel. The algorithms may be independent or cooperative. The model modifies the hyper-heuristic's behavior and enhances the solutions' quality. In the independent algorithmic model, various (meta)-heuristics are executed without any information exchange. The (meta)-heuristics are started with multiple populations and parameters (the probability of crossover and mutation). Worker processors run (meta)-heuristics and the master node sets the parameters of the workers and receives the best results. Good performance of speed up and robustness[1] are provided in this model.

In the cooperative model, (meta)-heuristics can exchange solutions for better results in single or multi-objective hyper-heuristic algorithms. When to communicate, which communication topology to be used, what knowledge to be exchanged, and the consolidation policy is the critical issues of the cooperative hyper-heuristic models.

The partitioning of the decision space can be used to decompose the objective problem. As a result, a hyper-heuristic can narrow its search to a subset (or partition) of the decision space.

3.4.2. Iteration-level parallelism

The iteration-based model parallelizes a single iteration of a hyper-heuristic algorithm. The primary goal of the design is to accelerate the process by manipulating a large population. A processor can compute the fitness of each solution. Generally, the fitness evaluation is a costly operation repeated by hyper-heuristics. This model is independent of the target problem. The surrogate-based computation used to obtain the fitness values more simply can be a method of these parallel hyper-heuristics (Jin, 2011).

The population of the hyper-heuristics may be updated in parallel because it can be a costly operation most of the time. The master initializes the population, selects the candidate solutions and inserts the new individuals into the population while the worker processors perform the fitness calculations. The master node can wait for all the workers to finish their calculations synchronously or start a new iteration asynchronously without waiting for all the workers to finalize

[1] Here, robustness of parallel algorithms refers to potential improvements on result quality (Cantu-Paz, 2000), having a natural affinity to alleviate the bloat phenomenon (Fernández, Galeano, & Gómez, 2002; Kucukyilmaz & Kiziloz, 2018; Ruciński, Izzo, & Biscani, 2010; Trujillo, Muñoz, Galván-López, & Silva, 2016), and having a better scalability potential (Dokeroglu & Cosar, 2016; Liu & Wang, 2015). Especially when using evolutionary approaches, several studies indicate that parallel frameworks maintain population diversity better than non-parallel counterparts, allowing better exploration for the framework and hence improvements in terms of result quality (Andre & Koza, 1996; Fernández, Gil, Baños, & Montoya, 2013; Fernández, Tomassini, Punch, & Sánchez, 2000; Kucukyilmaz & Kiziloz, 2018).

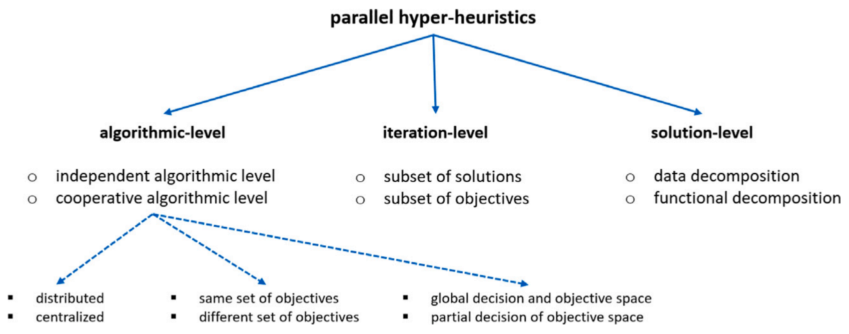
parallel hyper-heuristics
algorithmic-level
iteration-level
solution-level
independent algorithmic level
cooperative algorithmic level
subset of solutions
subset of objectives
data decomposition
functional decomposition
distributed
centralized
same set of objectives
different set of objectives
global decision and objective space
partial decision of objective space

Fig. 6. The taxonomy of parallel hyper-heuristics.

their estimates. In addition, some common knowledge of the (meta)-heuristics (like the pheromone matrix of Ant Colony Optimization) needs to be shared between the processors in this model.

3.4.3. Solution-level parallelism

In this design, because the fitness evaluation of solutions consumes most of the time, the algorithm handles every single solution in parallel. This model does not alter the behavior of the hyper-heuristic but shortens the processing time. Two decomposition strategies may be used, the decomposition of data and the decomposition of objective functions. The master node collects worker results and determines the best global fitness value.

In addition to these three models of hyper-heuristics, combinations are also possible using a hierarchical organization. Fig. 6 presents the proposed taxonomy of parallel hyper-heuristics.

This section will concentrate on efficiently implementing the proposed parallel hyper-heuristic models on various types of high-performance machines. The parallel hyper-heuristics' performance evaluation is mainly related to speed up, scalability, and energy consumption. Another aspect is the financial cost of the implementation in a commercial computation environment.

The granularity is the most critical aspect of parallel hyper-heuristic algorithms. It is the proportion of computation to communication. When the granularity is large, higher values of speed up are achieved. Algorithmic-level parallel hyper-heuristics have a better granularity than solution-level design approaches.

Parallel hyper-heuristics can use static, dynamic, and adaptive scheduling techniques. The processors' workload is decided at compile time in static scheduling, which may be inefficient for uneven load distributions. The job assignments are scheduled during execution in dynamic scheduling, and the number of processes is fixed. In the adaptive version, the number of processors and jobs change at run time.

In this part, we consider our proposed parallel hyper-heuristics models mentioned above from the aspects of granularity, degree of concurrency, asynchronous vs synchronous knowledge exchange, and scheduling.

The algorithmic-level parallelism has a bigger granularity than other models. Communication cost is reduced in this model; therefore, large-scale HPC machines' scalability performance becomes better. The number of involved parallel hyper-heuristics bounds the degree of concurrency. Synchronous or asynchronous communication can be used in this model. The synchronous version can be less efficient on a Grid. Using synchronous exchange in a heterogeneous environment will degrade the performance of parallel hyper-heuristics. The least powerful processor will dominate the performance. This model supports static, dynamic, and adaptive scheduling.

Iteration-level parallelism has a medium granularity. The model is effective depending on the size of the sub-populations or the time required for fitness value evaluations. Using large populations in this model enhances the scalability of the hyper-heuristics. Using asynchronous messaging increases the performance of the iteration-level parallelism model. Faster and less loaded processors can handle more problems than others in dynamic scheduling. As a result, compared to static scheduling, this approach reduces the execution time.

The solution-level parallelism has the smallest level of granularity of the three models. This model is slower for large-scale distributed machines with high communication costs. The number of data partitions/sub-functions limits the degree of concurrency (scalability). This model's implementation is master–worker and synchronous. Scheduling in the decomposition model corresponds to data partitions and sub-functions in the functional decomposition. For homogeneous non-shared machines, static scheduling appears to be efficient.

If we briefly summarize some studies in the literature: Crainic and Toulouse (2003) presented one of the first surveys of parallel metaheuristics and discussed the design principles of these algorithms. Parallel hyper-heuristics have much to benefit from this field. Rattadilok, Gaw, and Kwan (2004) investigated emerging search algorithms in their study. High-level heuristics were selected and executed to obtain optimal results, and a parallel architecture was used during the implementation. Significant performance improvements are obtained. Segura, Miranda, and León (2011) developed a parallel hyper-heuristic for the frequency assignment problem. For each instance to be solved, a memetic algorithm is used in the study and adapted to the solutions. The algorithm configured the settings that are executed on each island processor. The configurations were decided during the execution of the algorithms.

Van Onsem and Demoen (2013) presented a framework called ParHyFlex. The implementation of different hyper-heuristics was included in the software. It is tested on the Maximum Satisfiability Problem. Chana et al. (2013) proposed a hyper-heuristic based on bacterial foraging to schedule resources. The algorithm was compared to other recent heuristics. The results showed that the algorithm outperforms the others by minimizing the total cost. In another study, Borgulya (2014) presented a parallel hyper-heuristic for 2D strip-packing problems. An island parallel model with a memetic algorithm is proposed in the study. The new technique learned bad variable values based on the population's worst solutions and controlled the steps of each mutation operation. Dokeroglu and Cosar (2016) proposed a parallel hyper-heuristic to solve the Quadratic Assignment Problem (QAP). The algorithm uses ACO, simulated annealing, tabu search, and breakout local search metaheuristics. The best-performing (meta)-heuristic is decided using genetic approaches. The developed hyper-heuristic obtains the best results in the literature.

Alekseeva, Mezmaz, Tuyttens, and Melab (2017) offered a parallel multi-core hyper-heuristic for the flow-shop problem with the makespan criteria based on GRASP that incorporates a cost function based on a boundary operator. The suggested hyper-heuristic evaluated GRASP configurations and obtained the best setting. Parallel multi-core computing was utilized to implement the hyper-heuristic

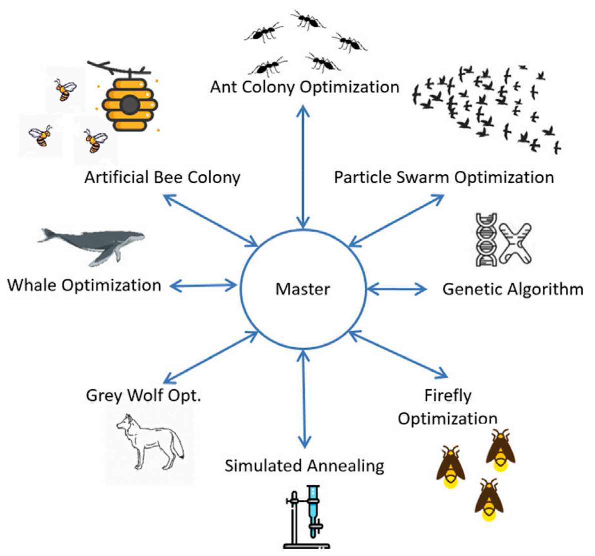

Fig. 7. The master and worker parallel computation environment of hyper-heuristics. At each node, a different metaheuristic algorithm can be executed simultaneously. This topology is considered to be one of the best communication topologies.

efficiently. Rodriguez, Oteiza, and Brignole (2019) proposed a study of the parallel combination of three metaheuristics. To escape from stagnation, the algorithm used genetic algorithms, simulated annealing, and ACO metaheuristics. The implementation executed the metaheuristics simultaneously. Therefore, it has increased the overall computation capacity significantly. The proposed optimization algorithm improved solution quality. Oteiza, Ardenghi, and Brignole (2021) developed a parallel cooperative hyper-heuristic framework for nonlinear algebraic equations with constraints. Genetic algorithms, PSO, and simulated annealing were used in this framework. The parameters of the methods were set during the execution of the optimization process (at run-time). A master and worker communication paradigm was implemented, and the master node ranked the candidates and communicated the best candidate to all other nodes. The results verified the efficiency of the approach because the parallel processing significantly increased the fitness evaluations of the candidate solutions and did not get into local optima as much as single metaheuristics. Duque Gallego (2022) proposed a framework to set the parameters of single solution metaheuristics in a parallel hybrid way for combinatorial optimization problems. The implementation used three heuristics. The author examines numerous parameter control strategies using parallel parameter settings of the metaheuristics.

Fig. 7 shows a master and worker parallel computation environment for hyper-heuristics, considered the most common computational topology used by parallel hyper-heuristics. At each node, a different (meta)-heuristic is executed efficiently while the master node selects, generates and coordinates the jobs (using dynamic scheduling) executed in the worker nodes. In Appendix, a summary of the state-of-the-art hyper-heuristic studies are listed in Tables A.2, A.3, A.4, and A.5.

4. Conclusions and open research issues

There is an increasing trend in developing hyper-heuristics to optimize hard optimization problems. To illustrate this trend quantitatively, we conducted a literature survey on Google Scholar, as illustrated in Fig. 8. In the survey, we included articles that appeared in scientific journals, international conferences, and books. To this end, we have extracted the relevant studies with a structured strategy using the Google Scholar API.[2] In our strategy, keywords of each publication cited in this work are extracted and searched for related publications in the related hyper-heuristic research category. In case a keyword appears in multiple categories, the keyword is moved upwards in the taxonomy presented in Fig. 2.

As indicated by Fig. 8, it is possible to observe this from the growing number of publications in recent years. Hyper-heuristics were initially considered niche solvers for limited hard optimization problems, but now they can be applied to many problems. This rapidly growing field of science aims to increase the generality level of optimization techniques for many problems by exploring the appropriate heuristics rather than solving single specific problems. That is, utilizing low-level heuristics instead of solution-spaced-based solvers.

Additionally, we extended the above survey and examined recent publications for the research objectives, type of selection algorithms, and in-case the publications include parallelization efforts and the type of parallelization techniques. The results of the rundown of the aforementioned categorical distributions are presented in Fig. 9. As the figure suggests, recent studies also indicate that advances in machine learning are attracting significant attention in this community for solving multi-objective problems with hyper-heuristic adaptations. The figure also indicates an important research gap in the literature: although by nature, hyper-heuristics are very suitable for parallelism, there is only a few studies that examine hyper-heuristics and their performance in parallel settings.

In our survey, we have covered and discussed state-of-the-art hyper-heuristic algorithms of the last two decades. To the best of our knowledge, there is a growing gap and a need for studies examining recent developments and cutting-edge research involving hyper-heuristics. Therefore, this study extends the existing taxonomy for hyper-heuristics, capable of capturing the current trends in the field. To this end, the proposed taxonomy involves four categories of hyper-heuristics, and examining these algorithms under these categories is the main contribution of this survey.

Hyper-heuristics involve two components: An algorithm that selects the low-level heuristics to solve a problem efficiently and a low-level repository and/or a generator for (meta)-heuristics. Developing new heuristics will always create attention toward hyper-heuristics as decisions involving how to select and sequence them to achieve the best results will be the focus of the optimization community. Configuring these techniques to achieve faster convergence, better results and lower resource requirements will ultimately decide which of the proposed techniques will find applications in real-life situations. In this regard, setting the parameters of (meta)-heuristics and finding novel strategies will always be an important point affecting the performance of the algorithms and will continue to be one of the high points of hyper-heuristics.

There is still not a solid and widely accepted theoretical understanding of hyper-heuristics. Therefore constructing the theoretical foundations of hyper-heuristics is still an open question. The speed of the fitness value calculation is another critical issue in many hyper-heuristic applications. Applying dynamic programming approaches seems to provide successful results in that aspect. It will positively affect the performance of these algorithms, especially the computation cost and shorter execution times. In this sense, studies involving NFL and fitness evaluation remain an important issue in hyper-heuristic research. Although exploration and exploitation are critical issues in hyper-heuristics, it seems that this issue has not been mentioned much in previous studies. In our opinion, determining the sequence of heuristics is closely related to the exploration and exploitation efforts of the optimization algorithms. We believe there is a strong correspondence between the exploration/exploitation and selection/generation and sequencing activities of (meta)-heuristics.

[2] https://serpapi.com/search?engine=google_scholar

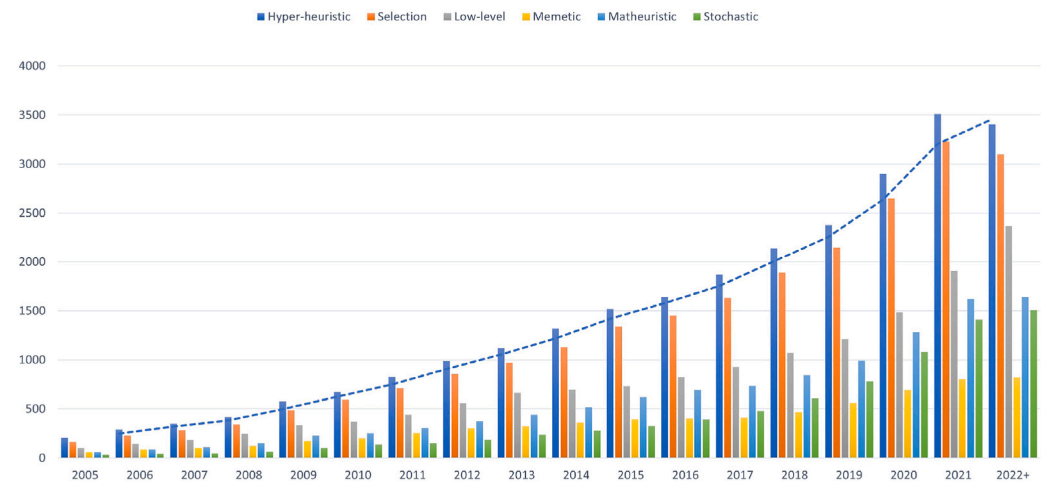

Fig. 8. Research trends on different branches of hyper-heuristics, displayed as the number of papers per year.

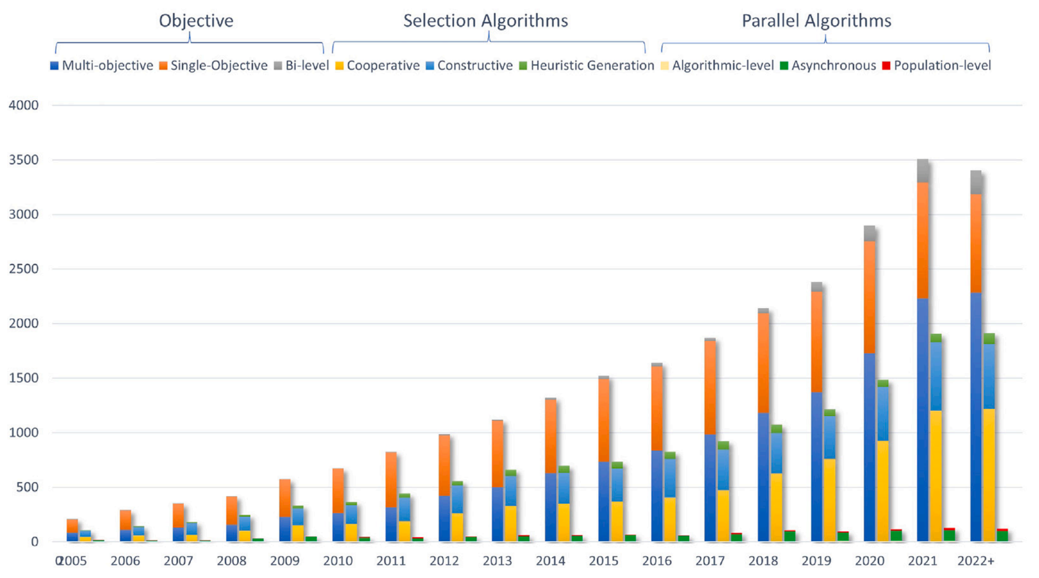

Fig. 9. Distribution of the selection of objective functions, selection algorithms, and parallel algorithms examined in the literature in the last two decades.

Stagnation (getting stuck into local optima) will always be a significant issue in both hyper-heuristics and (meta)-metaheuristic algorithms. Heuristics that deal better with this problem will have a higher chance of being selected by hyper-heuristics. Exascale supercomputers with millions of cores in multi-machines are currently on the horizon. This new architecture may open the way for the development of new high-performance and scalable hyper-heuristic algorithms. With their ever-increasing performance, GPUs are new technologies that can aid this research area.

Quantum-inspired algorithms incorporate quantum mechanics into classical metaheuristics using non-quantum machines (Dahi & Alba, 2022). Because quantum principles are unique, the inspiration of quantum phenomena and how it is done in fundamentally different non-quantum systems rather than real or simulated quantum computers introduces interesting and new issues about designing new algorithms in real or simulated quantum devices. In this sense, it is seen that the cooperation of quantum computation and hyper-heuristics will reveal many new research areas in the near future and will provide opportunities for higher-quality computation in optimization. New quantum-inspired metaheuristic studies started to appear in recent research activities.

The application of newly developed metaheuristics in the field of hyper-heuristics clearly shows the research gap in this field for future studies. For example, the inclusion of newly developed metaheuristics such as Harris Hawk (Heidari et al., 2019), Ant Lion (Mirjalili, 2015a), Moth Flame (Mirjalili, 2015b), Grey Wolf Optimizer (Mirjalili, Mirjalili, & Lewis, 2014), Dragonfly (Meraihi, Ramdane-Cherif, Acheli, & Mahseur, 2020), Grasshopper Optimization (Mirjalili, Mirjalili, Saremi, Faris, & Aljarah, 2018), Multi-Verse Optimizer (Mirjalili, Mirjalili, & Hatamlou, 2016), Sine Cosine (Mirjalili, 2016), Salp Swarm (Mirjalili et al., 2017), Whale Optimization (Mirjalili & Lewis, 2016), and Artificial Hummingbird (Zhao, Wang, & Mirjalili, 2022) into the repository of the hyper-heuristics will provide better results. Many problems in the field of metaheuristics are also closely related to hyper-heuristics, and any progress to be made will positively affect both sides.

As the last word, we think that hyper-heuristics are still in their infancy, and we expect developments in this field to shape the framework for tackling large-scale, real-life problems. We believe that the

Table A.2
Selected recent hyper-heuristic studies using metaheuristics.

Study	Metaheuristics	Multi-objec.	Mach. lear.	Problems solved
Tabataba and Mousavi (2012)	beam search	–	–	longest common subsequence
Abd Aziz (2015)	ACO	–	–	traveling salesman
Dokeroglu and Cosar (2016)	genetic, simulated annealing, ACO, breakout local search	–	–	quadratic assignment
Pandiri and Singh (2018)	ABC	–	–	traveling salesman
Navajas-Guerrero et al. (2022)	harmony search	–	–	prediction of failures
Cruz-Duarte et al. (2021)	gravitational, simulated annealing, genetic, differential evol. firefly, cuckoo search	–	–	testing scheduling
Gölcük and Ozsoydan (2021)	ABC, manta ray foraging, salp swarm, whale optimization	–	Q-learning	knapsack problems
Gonzalez et al. (2022)	evolution algorithm, scatter search, tabu, GRASP	–	–	MILP
Liu, Zhang, Zhang, Li, and Chen (2023)	aquila, arithmetic	–	Q-learning	global optimization

Table A.3
Selected recent multi-objective hyper-heuristic studies.

Study	Heuristic-based	Metaheuristics	Mach. lear.	Problems solved
Burke et al. (2005)	✔	–	–	scheduling, space allocation
Gómez and Terashima-Marín (2012)	✔	–	–	2D irregular cutting stock
Maashi et al. (2014) Maashi et al. (2015)	✔ ✔	– –	– –	walking fish group walking fish group
Gonçalves et al. (2015)	✔	–	–	CEC 2009 MOEA Competition
Guizzo et al. (2015)	✔	–	–	integration and test order
Walker and Keedwell (2016)	✔	–	–	DTLZ test suite (Deb, Thiele, Laumanns, & Zitzler, 2002)
Li et al. (2017)	–	✔	–	layout optimization
Zhang et al. (2019)	✔	–	–	job shop scheduling
Zhou et al. (2019)	✔	–	–	job shop scheduling
de Santiago Júnior et al. (2020)	–	✔	reinforcement	layout optimization
Almeida et al. (2020)	✔	–	–	permutation flow shop
de Carvalho et al. (2021)	✔	–	–	real-world
Alshareef and Maashi (2022)	✔	–	–	clustering
Venske et al. (2022)	✔	–	–	quadratic assignment
Heise and Mostaghim (2023)	✔	–	–	20 benchmark problems
Cao, Zhang, Zhou, and Tang (2023)	–	Simulated Annealing	–	structural damage identification

studies in multi-objective hyper-heuristic optimization will increase, and metaheuristic-based hyper-heuristics will be more prevalent in the future with the advances in metaheuristics.

Ethical approval

This article does not contain any studies with human participants performed by any of the authors.

This article does not contain any studies with animals performed by any of the authors.

This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent: There is no individual participant included in the study.

Funding

This study is not funded by any institution.

CRediT authorship contribution statement

Tansel Dokeroglu: Conceptualization, Methodology, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Tayfun Kucukyilmaz:** Methodology, Investigation, Resources, Data curation, Writing – review & editing, Visualization, Supervision. **El-Ghazali Talbi:** Conceptualization, Methodology, Investigation, Resources, Data curation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Table A.4
Selected recent hyper-heuristic studies using machine learning.

Study	Heuristic-based	Metaheuristics	Multi-objec.	Mach. lear.	Problems solved
Özcan et al. (2012)	✔	great deluge	–	reinforcement	exam timetabling
Hunt et al. (2012)	✔	genetic programming	–	–	feature selection
Maashi et al. (2014)	✔	NSGAII, SPEA2, MOGA	✔	–	walking fish group
Abd Elaziz and Mirjalili (2019)	–	whale optimization	–	–	initial population
Lassouaoui et al. (2022)	✔	–	–	reinforcement	feature selection
Lin et al. (2022)	✔	–	–	Q-learning	testing scheduling
Qin et al. (2021)	✔	✔	–	reinforcement deep learning	vehicle routing
Juntama et al. (2022)	✔	–	–	reinforcement	traffic complexity
Zhang et al. (2022)	✔	–	–	deep reinforcement	uncertainty

Table A.5
Selected recent parallel hyper-heuristic studies.

Study	Heuristic-based	Metaheuristics	Multi-objec.	Mach. lear.	Problems solved
Crainic and Toulouse (2003)	–	genetic, simulated annealing, tabu	–	–	general
Rattadilok et al. (2004)	✔	–	–	–	timetabling, scheduling
Segura et al. (2011)	–	memetic	–	–	frequency assignment
Chana et al. (2013)	–	bacterial foraging	–	–	resource scheduling
Van Onsem and Demoen (2013)	–	genetic, simulated annealing, tabu	✔	–	maximum satisfiability
Borgulya (2014)	✔	memetic	–	–	2D strip-packing
Dokeroglu and Cosar (2016)	–	genetic, simulated annealing, ACO, Breakout Local Search	–	–	quadratic assignment
Alekseeva et al. (2017)	–	GRASP	–	–	flow-shop problem
Rodriguez et al. (2019)	✔	genetic, simulated annealing, tabu	–	–	urban transportation
Oteiza et al. (2021)	–	genetic, simulated annealing, PSO	–	–	nonlinear algebraic equations
Duque Gallego (2022)	–	extremal, multi-start Local, tabu	–	–	quadratic assignment
Liu, Zhang, Liu, Zhang, and Wu (2023)	–	simulated annealing variable neighborhood, tabu	–	Q-lear.	allocation and ordering

Appendix. The list of recent studies

Some of the state-of-the-art hyper-heuristic studies are summarized in this Appendix. We aim to provide detailed information about recent publications for interested readers. The metaheuristics used by the hyper-heuristics, multi-objective hyper-heuristics, applied machine learning techniques, parallel hyper-heuristics, the problems solved by these algorithms are listed in the Tables A.2, A.3, A.4, and A.5.

References

Abd Aziz, Z. (2015). Ant colony hyper-heuristics for travelling salesman problem. *Procedia Computer Science*, *76*, 534–538.

Abd Elaziz, M., & Mirjalili, S. (2019). A hyper-heuristic for improving the initial population of whale optimization algorithm. *Knowledge-Based Systems*, *172*, 42–63.

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

Alekseeva, E., Mezmaz, M., Tuyttens, D., & Melab, N. (2017). Parallel multi-core hyper-heuristic GRASP to solve permutation flow-shop problem. *Concurrency Computations: Practice and Experience*, *29*(9), Article e3835.

Almeida, C. P., Gonçalves, R. A., Venske, S., Lüders, R., & Delgado, M. (2020). Hyper-heuristics using multi-armed bandit models for multi-objective optimization. *Applied Soft Computing*, *95*, Article 106520.

Alshareef, H., & Maashi, M. (2022). Application of multi-objective hyper-heuristics to solve the multi-objective software module clustering problem. *Applied Sciences*, *12*(11), 5649.

Amaya, I., Cruz-Duarte, J. M., & Ortiz-Bayliss, J. C. (2022). MatHH: A matlab-based hyper-heuristic framework. *SoftwareX*, *18*, Article 101047.

Amini, A., Moghaddam, R. T., & Ebrahimnejad, S. (2020). A robust location-arc routing problem under uncertainty: mathematical model with lower and upper bounds. *Computational & Applied Mathematics*, *39*(4), 318.

Andre, D., & Koza, J. R. (1996). Parallel genetic programming: A scalable implementation using the transputer network architecture. In *Advances in genetic programming. Vol. 2* (pp. 317–337). MIT Press.

Archetti, C., & Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, *2*(4), 223–246.

Assunção, W. K. G., Colanzi, T. E., Vergilio, S. R., & Pozo, A. (2014). A multi-objective optimization approach for the integration and test order problem. *Information Sciences*, *267*, 119–139.

Beyaz, M., Dokeroglu, T., & Cosar, A. (2015). Robust hyper-heuristic algorithms for the offline oriented/non-oriented 2D bin packing problems. *Applied Soft Computing*, *36*, 236–245.

Bilgin, B., Özcan, E., & Korkmaz, E. E. (2006). An experimental study on hyper-heuristics and exam timetabling. In *International conference on the practice and theory of automated timetabling* (pp. 394–412). Springer.

Bleuler, S., Laumanns, M., Thiele, L., & Zitzler, E. (2003). PISA—a platform and programming language independent interface for search algorithms. In *International conference on evolutionary multi-criterion optimization* (pp. 494–508). Springer.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*(6), 4135–4151.

Borgulya, I. (2014). A parallel hyper-heuristic approach for the two-dimensional rectangular strip-packing problem. *Journal of Computing and Information Technology*, *22*(4), 251–265.

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117.

Branke, J., Nguyen, S., Pickardt, C. W., & Zhang, M. (2015). Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation*, *20*(1), 110–124.

Burke, E. K., Burke, E. K., Kendall, G., & Kendall, G. (2014). *Search methodologies: introductory tutorials in optimization and decision support techniques*. Springer.

Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., et al. (2010). Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In *IEEE congress on evolutionary computation* (pp. 1–8). IEEE.

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

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

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

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2019). A classification of hyper-heuristic approaches: revisited. In *Handbook of metaheuristics* (pp. 453–477). Springer.

Burke, E. K., Kendall, G., Mısır, M., & Özcan, E. (2012). Monte Carlo hyper-heuristics for examination timetabling. *Annals of Operations Research*, *196*(1), 73–90.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., & Schulenburg, S. (2003). Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of metaheuristics* (pp. 457–474). Springer.

Burke, E. K., Silva, J., & Soubeiga, E. (2005). Multi-objective hyper-heuristic approaches for space allocation and timetabling. In *Metaheuristics: progress as real problem solvers* (pp. 129–158). Springer.

Cantu-Paz, E. (2000). *Efficient and accurate parallel genetic algorithms*. Springer Science & Business Media.

Cao, P., Zhang, Y., Zhou, K., & Tang, J. (2023). A reinforcement learning hyper-heuristic in multi-objective optimization with application to structural damage identification. *Structural and Multidisciplinary Optimization*, *66*(1), 16.

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

Chen, H., Ding, G., Qin, S., & Zhang, J. (2021). A hyper-heuristic based ensemble genetic programming approach for stochastic resource constrained project scheduling problem. *Expert Systems with Applications*, *167*, Article 114174.

Chen, S., Li, Z., Yang, B., & Rudolph, G. (2015). Quantum-inspired hyper-heuristics for energy-aware scheduling on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems*, *27*(6), 1796–1810.

Cheng, L., Tang, Q., Zhang, L., & Zhang, Z. (2022). Multi-objective Q-learning-based hyper-heuristic with Bi-criteria selection for energy-aware mixed shop scheduling. *Swarm and Evolutionary Computation*, *69*, Article 100985.

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

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

Cowling, P., Kendall, G., & Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. In *International conference on the practice and theory of automated timetabling* (pp. 176–190). Springer.

Crainic, T. G., & Toulouse, M. (2003). Parallel strategies for meta-heuristics. In *Handbook of metaheuristics* (pp. 475–513). Springer.

Crowston, W. B., Glover, F., Trawick, J. D., et al. (1963). *Probabilistic and parametric learning combinations of local job shop scheduling rules*: *Technical Report*, Carnegie Inst of Tech Pittsburgh Pa Graduate School of Industrial Administration.

Cruz-Duarte, J. M., Amaya, I., Ortiz-Bayliss, J. C., Conant-Pablos, S. E., Terashima-Marín, H., & Shi, Y. (2021). Hyper-Heuristics to customise metaheuristics for continuous optimisation. *Swarm and Evolutionary Computation*, *66*, Article 100935.

Cruz-Duarte, J. M., Amaya, I., Ortiz-Bayliss, J. C., Terashima-Marín, H., & Shi, Y. (2020). CUSTOMHyS: Customising optimisation metaheuristics via hyper-heuristic search. *SoftwareX*, *12*, Article 100628. http://dx.doi.org/10.1016/j.softx.2020.100628.

Cruz-Duarte, J. M., Ortiz-Bayliss, J. C., & Amaya, I. (2022). MatHH: A matlab-based hyper-heuristic framework. *SoftwareX*, *18*, Article 101047. http://dx.doi.org/10.1016/j.softx.2022.101047.

Cutillas-Lozano, J.-M., Giménez, D., & Almeida, F. (2015). Hyperheuristics based on parametrized metaheuristic schemes. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation* (pp. 361–368).

Dahi, Z. A., & Alba, E. (2022). Metaheuristics on quantum computers: Inspiration, simulation and real execution. *Future Generation Computer Systems*, *130*, 164–180.

de Carvalho, V. R., Özcan, E., & Sichman, J. S. (2021). Comparative analysis of selection hyper-heuristics for real-world multi-objective optimization problems. *Applied Sciences*, *11*(19), 9153.

De Maio, A., Laganà, D., Musmanno, R., & Vocaturo, F. (2021). Arc routing under uncertainty: Introduction and literature review. *Computers & Operations Research*, *135*, Article 105442. http://dx.doi.org/10.1016/j.cor.2021.105442.

de Santiago Júnior, V. A., Özcan, E., & de Carvalho, V. R. (2020). Hyper-heuristics based on reinforcement learning, balanced heuristic selection and group decision acceptance. *Applied Soft Computing*, *97*, Article 106760.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002). Scalable multi-objective optimization test problems. In *Proceedings of the 2002 congress on evolutionary computation. CEC'02. Vol. 1* (pp. 825–830). IEEE.

Doerner, K. F., & Schmid, V. (2010). Survey: matheuristics for rich vehicle routing problems. In *International workshop on hybrid metaheuristics* (pp. 206–221). Springer.

Dokeroglu, T., & Cosar, A. (2016). A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, *52*, 10–25.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, *137*, Article 106040.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39.

Drake, J. H., Kheiri, A., Ozcan, E., & Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, *285*(2), 405–428.

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

Duflo, G., Danoy, G., Talbi, E.-G., & Bouvry, P. (2020). Automating the design of efficient distributed behaviours for a swarm of UAVs. In *2020 IEEE symposium series on computational intelligence* (pp. 489–496). IEEE.

Duflo, G., Danoy, G., Talbi, E.-G., & Bouvry, P. (2022). A framework of hyper-heuristics based on Q-learning. In *International conference in optimization and learning.*

Duque Gallego, J. (2022). *Parameter control strategies of parallel hybrid metaheuristics applied to the quadratic assignment problem*. Grupo de Investigación en Telecomunicaciones Aplicadas (GITA).

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760–771.

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

Ersoy, E., Ozcan, E., & Etaner-Uyar, A. (2007). Memetic algorithms and hyperhill-climbers. In P. Baptiste, G. Kendall, A. M. Kordon, & F. Sourd (Eds.), *Proc. of the 3rd multidisciplinary int. conf. on scheduling: Theory and applications* (pp. 159–166).

Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*(2), 109–133.

Fernández, F., Galeano, G., & Gómez, J. A. (2002). Comparing synchronous and asynchronous parallel and distributed genetic programming models. In *Genetic programming: 5th European conference, EuroGP 2002 Kinsale, Ireland, April 3–5, 2002 Proceedings* (pp. 326–335). Berlin, Heidelberg: Springer.

Fernández, A., Gil, C., Baños, R., & Montoya, M. G. (2013). A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing. *Expert Systems with Applications*, *40*(13), 5169–5180.

Fernández, F., Tomassini, M., Punch, W. F., & Sánchez, J. M. (2000). Experimental study of multipopulation parallel genetic programming. In *Genetic programming: European conference, EuroGP 2000, Edinburgh, Scotland, UK, April 15-16, 2000. Proceedings* (pp. 283–293). Berlin, Heidelberg: Springer.

Fischetti, M., & Fischetti, M. (2018). Matheuristics. In *Handbook of heuristics* (pp. 121–153). Springer.

Garrido, P., & Riff, M. C. (2010). DVRP: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics*, *16*(6), 795–834.

Gölcük, İ., & Ozsoydan, F. B. (2021). Q-learning and hyper-heuristic based algorithm recommendation for changing environments. *Engineering Applications of Artificial Intelligence*, *102*, Article 104284.

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

Gómez, J. C., & Terashima-Marín, H. (2012). Building general hyper-heuristics for multi-objective cutting stock problem. *Computación y Sistemas*, *16*(3), 321–334.

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

Gonzalez, M., López-Espín, J. J., Aparicio, J., & Talbi, E.-G. (2022). A hyper-matheuristic approach for solving mixed integer linear optimization models in the context of data envelopment analysis. *PeerJ Computer Science*, *8*, Article e828.

Guerriero, F., & Saccomanno, F. P. (2022). A hierarchical hyper-heuristic for the bin packing problem. *Soft Computing*, 1–14.

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

Guizzo, G., Vergilio, S. R., Pozo, A. T., & Fritsche, G. M. (2017). A multi-objective and evolutionary hyper-heuristic applied to the integration and test order-https://www.overleaf.com/project/630b534fdcc836f7b4a3a10d problem. *Applied Soft Computing*, *56*, 331–344.

Hao, X., Qu, R., & Liu, J. (2020). A unified framework of graph-based evolutionary multitasking hyper-heuristic. *IEEE Transactions on Evolutionary Computation*, *25*(1), 35–47.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*, *97*, 849–872.

Heise, J., & Mostaghim, S. (2023). Online learning hyper-heuristics in multi-objective evolutionary algorithms. In *Evolutionary multi-criterion optimization: 12th international conference, EMO 2023, Leiden, the Netherlands, March 20–24, 2023, Proceedings* (pp. 162–175). Springer.

Hitomi, N., & Selva, D. (2015). The effect of credit definition and aggregation strategies on multi-objective hyper-heuristics. In *International design engineering technical conferences and computers and information in engineering conference. Vol. 57083*. American Society of Mechanical Engineers, V02BT03A030.

Hunt, R., Neshatian, K., & Zhang, M. (2012). A genetic programming approach to hyper-heuristic feature selection. In *Asia-pacific conference on simulated evolution and learning* (pp. 320–330). Springer.

Jiang, C., Han, X., Liu, G., & Liu, G. (2008). A nonlinear interval number programming method for uncertain optimization problems. *European Journal of Operational Research*, *188*(1), 1–13.

Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, *1*(2), 61–70.

Juntama, P., Delahaye, D., Chaimatanan, S., & Alam, S. (2022). Hyperheuristic approach based on reinforcement learning for air traffic complexity mitigation. *Journal of Aerospace Information Systems*, 1–16.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*(2), 393–422.

Kendall, G., Cowling, P., Soubeiga, E., et al. (2002). Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-pacific conference on simulated evolution and learning* (pp. 667–671).

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

Kheiri, A., Özcan, E., & Parkes, A. J. (2012). *HySST: hyper-heuristic search strategies and timetabling*. SINTEF.

Kieffer, E., Danoy, G., Brust, M. R., Bouvry, P., & Nagih, A. (2019). Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic. *IEEE Transactions on Evolutionary Computation*, *24*(1), 44–56.

Knowles, J. D., & Corne, D. W. (2000). M-PAES: A memetic algorithm for multiobjective optimization. In *Proceedings of the 2000 congress on evolutionary computation. CEC00. Vol. 1* (pp. 325–332). IEEE.

Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In *Data mining and constraint programming* (pp. 149–190). Springer.

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

Kramer, R., Subramanian, A., Vidal, T., & Lucídio dos Anjos, F. C. (2015). A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, *243*(2), 523–539.

Krasnogor, N., & Smith, J. (2005). A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, *9*(5), 474–488.

Kucukyilmaz, T., & Kiziloz, H. E. (2018). Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. *Computers & Industrial Engineering*, *125*, 157–170. http://dx.doi.org/10.1016/j.cie.2018.08.021.

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

Laporte, G., Musmanno, R., & Vocaturo, F. (2010). An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, *44*(1), 125–135.

Lara-Cárdenas, E., Silva-Gálvez, A., Ortiz-Bayliss, J. C., Amaya, I., Cruz-Duarte, J. M., & Terashima-Marín, H. (2020). Exploring reward-based hyper-heuristics for the job-shop scheduling problem. In *2020 IEEE symposium series on computational intelligence* (pp. 3133–3140). IEEE.

Lassouaoui, M., Boughaci, D., & Benhamou, B. (2022). A synergy Thompson sampling hyper-heuristic for the feature selection problem. *Computational Intelligence*, *38*(3), 1083–1105.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lei, Y., Gong, M., Jiao, L., & Zuo, Y. (2015). A memetic algorithm based on hyper-heuristics for examination timetabling problems. *International Journal of Intelligent Computing and Cybernetics*.

Leite, N., Fernandes, C. M., Melício, F., & Rosa, A. C. (2018). A cellular memetic algorithm for the examination timetabling problem. *Computers & Operations Research*, *94*, 118–138. http://dx.doi.org/10.1016/j.cor.2018.02.009.

Li, W., Özcan, E., Drake, J. H., & Maashi, M. (2023). A generality analysis of multiobjective hyper-heuristics. *Information Sciences*.

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

Lin, J., Li, Y.-Y., & Song, H.-B. (2022). Semiconductor final testing scheduling using Q-learning based hyper-heuristic. *Expert Systems with Applications*, *187*, Article 115978.

Liu, Y. Y., & Wang, S. (2015). A scalable parallel genetic algorithm for the generalized assignment problem. *Parallel Computing*, *46*(C), 98–119.

Liu, J., Zhang, Z., Liu, S., Zhang, Y., & Wu, T. (2023). Parallel hyper heuristic algorithm based on reinforcement learning for the corridor allocation problem and parallel row ordering problem. *Advanced Engineering Informatics*, *56*, Article 101977.

Liu, H., Zhang, X., Zhang, H., Li, C., & Chen, Z. (2023). A reinforcement learning-based hybrid Aquila Optimizer and improved Arithmetic Optimization Algorithm for global optimization. *Expert Systems with Applications*, *224*, Article 119898.

Löfberg, J. (2008). Modeling and solving uncertain optimization problems in YALMIP. *IFAC Proceedings Volumes*, *41*(2), 1337–1341.

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

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

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

Maniezzo, V., Stützle, T., & Voß, S. (2021). *Matheuristics*. Springer.

Meneghini, I. R., Alves, M. A., Gaspar-Cunha, A., & Guimaraes, F. G. (2020). Scalable and customizable benchmark problems for many-objective optimization. *Applied Soft Computing*, *90*, Article 106139.

Meraihi, Y., Ramdane-Cherif, A., Acheli, D., & Mahseur, M. (2020). Dragonfly algorithm: a comprehensive review and applications. *Neural Computing and Applications*, *32*(21), 16625–16646.

Mirjalili, S. (2015a). The ant lion optimizer. *Advances in Engineering Software*, *83*, 80–98.

Mirjalili, S. (2015b). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, *89*, 228–249.

Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163–191.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications*, *27*(2), 495–513.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61.

Mirjalili, S. Z., Mirjalili, S., Saremi, S., Faris, H., & Aljarah, I. (2018). Grasshopper optimization algorithm for multi-objective optimization problems. *Applied Intelligence*, *48*(4), 805–820.

Mısır, M. (2021). Hyper-heuristics: autonomous problem solvers. In *Automated design of machine learning and search algorithms* (pp. 109–131). Springer.

Moreno, J., Ortega, G., Filatovas, E., Martínez, J. A., & Garzón, E. M. (2017). Using low-power platforms for evolutionary multi-objective optimization algorithms. *The Journal of Supercomputing*, *73*(1), 302–315.

Navajas-Guerrero, A., Manjarres, D., Portillo, E., & Landa-Torres, I. (2022). A hyper-heuristic inspired approach for automatic failure prediction in the context of industry 4.0. *Computers & Industrial Engineering*, *171*, Article 108381.

Nesi, L. C., & da Rosa Righi, R. (2020). H2-SLAN: A hyper-heuristic based on stochastic learning automata network for obtaining, storing, and retrieving heuristic knowledge. *Expert Systems with Applications*, *153*, Article 113426.

Ning, C., & You, F. (2019). Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming. *Computers & Chemical Engineering*, *125*, 434–448. http://dx.doi.org/10.1016/j.compchemeng.2019.03.034.

Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J. A., Walker, J., Gendreau, M., et al. (2012). Hyflex: A benchmark framework for cross-domain heuristic search. In *European conference on evolutionary computation in combinatorial optimization* (pp. 136–147). Springer.

Olgun, B., Koç, Ç., & Altıparmak, F. (2021). A hyper heuristic for the green vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, *153*, Article 107010.

Ortiz-Bayliss, J. C., Amaya, I., Cruz-Duarte, J. M., Gutierrez-Rodriguez, A. E., Conant-Pablos, S. E., & Terashima-Marín, H. (2021). A general framework based on machine learning for algorithm selection in constraint satisfaction problems. *Applied Sciences*, *11*(6), 2749.

Oteiza, P. P., Ardenghi, J. I., & Brignole, N. B. (2021). Parallel hyper-heuristics for process engineering optimization. *Computers & Chemical Engineering*, *153*, Article 107440.

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

Oyebolu, F. B., Allmendinger, R., Farid, S. S., & Branke, J. (2019). Dynamic scheduling of multi-product continuous biopharmaceutical facilities: A hyper-heuristic framework. *Computers & Chemical Engineering*, *125*, 71–88. http://dx.doi.org/10.1016/j.compchemeng.2019.03.002.

Özcan, E. (2006). Memes, self-generation and nurse rostering. In *International conference on the practice and theory of automated timetabling* (pp. 85–104). Springer.

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

Özcan, E., Misir, M., Ochoa, G., & Burke, E. K. (2012). A reinforcement learning: great-deluge hyper-heuristic for examination timetabling. In *Modeling, analysis, and applications in metaheuristic computing: advancements and trends* (pp. 34–55). IGI Global.

Pandiri, V., & Singh, A. (2018). A hyper-heuristic based artificial bee colony algorithm for k-interconnected multi-depot multi-traveling salesman problem. *Information Sciences*, *463*, 261–281.

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

Pereira, J., Ritt, M., & Vásquez, Ó. C. (2018). A memetic algorithm for the cost-oriented robotic assembly line balancing problem. *Computers & Operations Research*, *99*, 249–261. http://dx.doi.org/10.1016/j.cor.2018.07.001.

Pickardt, C. W., Hildebrandt, T., Branke, J., Heger, J., & Scholz-Reiter, B. (2013). Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics*, *145*(1), 67–77.

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

Pillay, N., & Beckedahl, D. (2017). EvoHyp - a Java toolkit for evolutionary algorithm hyper-heuristics. In *2017 IEEE congress on evolutionary computation* (pp. 2706–2713). http://dx.doi.org/10.1109/CEC.2017.7969636.

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

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

Price, K. V. (2013). Differential evolution. In *Handbook of optimization* (pp. 187–214). Springer.

Qian, C., Tang, K., & Zhou, Z.-H. (2016). Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In *International conference on parallel problem solving from nature* (pp. 835–846). Springer.

Qin, W., Zhuang, Z., Huang, Z., & Huang, H. (2021). A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem. *Computers & Industrial Engineering*, *156*, Article 107252.

Rattadilok, P., Gaw, A., & Kwan, R. S. (2004). Distributed choice function hyper-heuristics for timetabling and scheduling. In *International conference on the practice and theory of automated timetabling* (pp. 51–67). Springer.

Renaud, A., Absi, N., & Feillet, D. (2017). The stochastic close-enough arc routing problem. *Networks*, *69*(2), 205–221. http://dx.doi.org/10.1002/net.21729.

Rodriguez, D. A., Oteiza, P. P., & Brignole, N. B. (2019). An urban transportation problem solved by parallel programming with hyper-heuristics. *Engineering Optimization*, *51*(11), 1965–1979.

Ross, P., Schulenburg, S., Marín-Bläzquez, J. G., & Hart, E. (2002). Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In *Proceedings of the 4th annual conference on genetic and evolutionary computation* (pp. 942–948).

Ruciński, M., Izzo, D., & Biscani, F. (2010). On the impact of the migration topology on the Island Model. *Parallel Computing*, *36*(10), 555–571.

Ryser-Welch, P., & Miller, J. F. (2014). A review of hyper-heuristic frameworks. *2014*, In *Proceedings of the Evo20 workshop, Aisb*.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2014a). Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, *19*(3), 309–325.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2014b). A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, *45*(2), 217–228.

Sabar, N. R., Ayob, M., Qu, R., & Kendall, G. (2012). A graph coloring constructive hyper-heuristic for examination timetabling problems. *Applied Intelligence*, *37*(1), 1–11.

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

Sánchez, M., Cruz-Duarte, J. M., carlos Ortíz-Bayliss, J., Ceballos, H., Terashima-Marin, H., & Amaya, I. (2020). A systematic review of hyper-heuristics on combinatorial optimization problems. *IEEE Access*, *8*, 128068–128095.

Segura, C., Miranda, G., & León, C. (2011). Parallel hyperheuristics for the frequency assignment problem. *Memetic Computing*, *3*(1), 33–49.

Shao, Z., Shao, W., & Pi, D. (2022). LS-HH: A learning-based selection hyper-heuristic for distributed heterogeneous hybrid blocking flow-shop scheduling. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Shouwen, J., Di, L., Zhengrong, C., & Dong, G. (2021). Integrated scheduling in automated container terminals considering AGV conflict-free routing. *Transportation Letters*, *13*(7), 501–513. http://dx.doi.org/10.1080/19427867.2020.1733199.

Steenson, A., Özcan, E., Kheiri, A., McCollum, B., & McMullan, P. (2022). An online learning selection hyper-heuristic for educational timetabling.

Swan, J., Özcan, E., & Kendall, G. (2011). Hyperion–a recursive hyper-heuristic framework. In *International conference on learning and intelligent optimization* (pp. 616–630). Springer.

Tabataba, F. S., & Mousavi, S. R. (2012). A hyper-heuristic for the longest common subsequence problem. *Computational Biology and Chemistry*, *36*, 42–54.

Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. John Wiley & Sons.

Talbi, E.-G. (2013). A taxonomy of metaheuristics for bi-level optimization. In *Metaheuristics for bi-level optimization* (pp. 1–39). Springer.

Talbi, E.-G. (2019). A unified view of parallel multi-objective evolutionary algorithms. *Journal of Parallel and Distributed Computing*, *133*, 349–358.

Trujillo, L., Muñoz, L., Galván-López, E., & Silva, S. (2016). Neat genetic programming: Controlling bloat naturally. *Information Sciences*, *333*, 21–43. http://dx.doi.org/10.1016/j.ins.2015.11.010.

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

Turky, A. (2019). *Bi-level hyper-heuristic approaches for combinatorial optimisation problems* (Ph.D. thesis), RMIT University.

Urra, E., Cabrera-Paniagua, D., & Cubillos, C. (2013). Towards an object-oriented pattern proposal for heuristic structures of diverse abstraction levels. In *XXI Jornadas Chilenas de Computación. Vol. 342*.

van der Weide, T., Deng, Q., & Santos, B. F. (2022). Robust long-term aircraft heavy maintenance check scheduling optimization under uncertainty. *Computers & Operations Research*, *141*, Article 105667. http://dx.doi.org/10.1016/j.cor.2021.105667, URL https://www.sciencedirect.com/science/article/pii/S0305054821003671.

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

Vela, A., Cruz-Duarte, J. M., Ortiz-Bayliss, J. C., & Amaya, I. (2022). Beyond hyper-heuristics: A squared hyper-heuristic model for solving job shop scheduling problems. *IEEE Access*, *10*, 43981–44007.

Venske, S. M., Almeida, C. P., Lüders, R., & Delgado, M. R. (2022). Selection hyper-heuristics for the multi and many-objective quadratic assignment problem. *Computers & Operations Research*, *148*, Article 105961.

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

Wang, X., & Tang, L. (2017). A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem. *Computers & Operations Research*, *79*, 60–77. http://dx.doi.org/10.1016/j.cor.2016.10.003.

jing Wang, J., & Wang, L. (2022). A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling. *Computers & Industrial Engineering*, *168*, Article 108126. http://dx.doi.org/10.1016/j.cie.2022.108126.

Wen, M., Linde, E., Ropke, S., Mirchandani, P., & Larsen, A. (2016). An adaptive large neighborhood search heuristic for the Electric Vehicle Scheduling Problem. *Computers & Operations Research*, *76*, 73–83. http://dx.doi.org/10.1016/j.cor.2016.06.013.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Xu, L., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2008). SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, *32*, 565–606.

Yang, Y., Zhong, M., Dessouky, Y., & Postolache, O. (2018). An integrated scheduling method for AGV routing in automated container terminals. *Computers & Industrial Engineering*, *126*, 482–493. http://dx.doi.org/10.1016/j.cie.2018.10.007.

Yao, Y., Peng, Z., & Xiao, B. (2018). Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city. *IEEE Transactions on Vehicular Technology*, *67*(11), 10307–10318.

Zhang, Y., Bai, R., Qu, R., Tu, C., & Jin, J. (2022). A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *European Journal of Operational Research*, *300*(2), 418–427.

Zhang, F., Mei, Y., & Zhang, M. (2019). Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics. In *2019 IEEE congress on evolutionary computation* (pp. 1366–1373). IEEE.

Zhao, W., Wang, L., & Mirjalili, S. (2022). Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Computer Methods in Applied Mechanics and Engineering*, *388*, Article 114194.

Zhou, Y., Yang, J.-J., & Zheng, L.-Y. (2019). Multi-agent based hyper-heuristics for multi-objective flexible job shop scheduling: A case study in an aero-engine blade manufacturing plant. *IEEE Access*, *7*, 21147–21176.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-Report*, *103*.