IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 December 2024, accepted 17 January 2025, date of publication 20 January 2025, date of current version 24 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3532201

RESEARCH ARTICLE

Hyper-Heuristics and Scheduling Problems: Strategies, Application Areas, and Performance Metrics

ALONSO VELA[1], GERARDO HUMBERTO VALENCIA-RIVERA[1], JORGE M. CRUZ-DUARTE[2], (Senior Member, IEEE), JOSÉ CARLOS ORTIZ-BAYLISS[1], (Member, IEEE), AND IVAN AMAYA[1], (Senior Member, IEEE)

[1]School of Engineering and Sciences, Tecnologico de Monterrey, Monterrey 64700, Mexico
[2]CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, University of Lille, F-59000 Lille, France

Corresponding author: Ivan Amaya (iamaya2@tec.mx)

This work was supported in part by the Mexican National Council of Humanities, Science and Technology CONAHCyT under Basic Science Project under Grant 287479; and in part by the Research Group in Advanced Artificial Intelligence at Tecnológico de Monterrey under Grant NUA A00815751 and Grant NUA A00834075.

ABSTRACT Scheduling problems, which involve allocating resources to tasks over specified time periods to optimize objectives, are crucial in various fields. This work presents hyper-heuristic applications for scheduling problems, analyzing 215 peer-reviewed publications over the last decade. We categorize and examine the prevailing strategies and configurations of hyper-heuristics, mainly focusing on their application across diverse scheduling scenarios such as job shop, flow shop, timetabling, and project scheduling. Our findings reveal a strong inclination towards selection and perturbative hyper-heuristics, with evolutionary computation emerging as the most commonly employed technique in this context, particularly in job shop and timetabling problems. Despite the robust development in hyper-heuristic methodologies, our analysis indicates an under-representation of multi-objective optimization and a limited use of performance metrics beyond makespan and tardiness. We also identify potential areas for future research, such as expanding hyper-heuristic applications to underexplored industries and exploring less conventional performance metrics. By providing a comprehensive overview of the current landscape and outlining future research directions, we aim to guide and inspire ongoing innovations in scheduling problem research.

INDEX TERMS Combinatorial optimization problems, hyper-heuristics, job shop scheduling, scheduling problems.

I. INTRODUCTION

By the end of 2023, the global population exceeded 8 billion, highlighting the urgent need for efficient resource management in our consumer-driven world. The Cambridge Dictionary defines efficiency as ''the good use of time and energy in a way that does not waste any.'' This is a principle essential to solving Scheduling Problems (SPs). SPs involve allocating limited resources to tasks over time, aiming to optimize one or more objectives, such as minimizing total operational time, reducing costs, or maximizing overall efficiency [2].

There are three types of algorithms for tackling SPs: exact algorithms, approximation algorithms, and heuristic algorithms. Exact algorithms can find optimal solutions, and some examples include depth-first search and breadth-first search [3], [4]. However, as the problem grows (*i.e.*, considering more resources and tasks), finding the optimal solution in a feasible amount of time becomes nearly impossible. Approximation algorithms, in contrast, produce solutions that are guaranteed to be close to the actual optimum. They are also considered fast, as they run in polynomial time [5]. For

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Abdur Razzaque.

example, Lenstra et al. proposed a polynomial algorithm that constructs a job-shop schedule shorter than twice the optimal makespan [5].

Finally, heuristic algorithms produce feasible solutions in a short period. However, such solutions are not guaranteed to be close to any optima. Their strength, thus, comes from the fact that they can be used to tackle larger problems due to their simplicity [6]. Some sources include heuristic algorithms within the category of approximation algorithms [7], [8], [9], [10]. However, considering the specific characteristics outlined earlier —that they are not guaranteed to provide solutions close to an optimal value and their computational efficiency in handling larger problems— we categorize them separately. Furthermore, heuristic algorithms can be classified as constructive and perturbative. Constructive heuristics build a solution from scratch by sorting all tasks according to some given rules. They also include dispatching rules, for which some examples include the *Service In Random Order (SIRO)*, which schedules a task randomly, and the *Earliest Release Date first (ERD)*, which schedules the task with the earliest release date. On the other hand, perturbative heuristics modify an already existing solution. Examples of these are local search methods, such as iterative improvement [11], threshold accepting [12], and genetic operators [13].

One way to improve heuristics is to use the so-called *Hyper-heuristics (HHs)*, which were introduced over 25 years ago by Dezinger et al. [14], who described them as a heuristic able to choose other heuristics. Therefore, they are also within the category of heuristic algorithms. Since then, this concept has been used to tackle various problems, including scheduling problems. Recently, Fan et al. [15] investigated a genetic programming-based HH for automatically designing dispatching rules to solve the Dynamic Job Shop Scheduling Problem (DJSP). In another work, Vela et al. went beyond the traditional HH model and introduced the idea of a *Squared Hyper-Heuristic (SHH)*, *i.e.*, a hyper-heuristic able to choose other hyper-heuristics for solving job shop scheduling problems [16]. Moreover, Song et al. proposed a hyper-heuristic-based memetic algorithm to solve the distributed assembly permutation flow-shop scheduling problem to minimize the maximum completion time [17]. Although the list of relevant works goes on and on, we omit a discussion of them for the sake of brevity.

Over the years, many HH approaches have been used to tackle scheduling problems. Hence, our motivation to systematically review them. The work from Bagheri is an example of a related systematic review, where the author focused on the multi-factory scheduling problems [18]. Similarly, Sánchez et al. conducted a systematic review of hyper-heuristic solvers for combinatorial optimization problems [19]. In a broader scope, Abiodun et al. presented a systematic review of emerging feature selection optimization methods for optimal text classification [20]. There are many more systematic reviews in the literature. However, none of them focus on HHs to solve scheduling problems. Therefore, this work aims to fill that gap by analyzing and evaluating the publications from the last decade. Moreover, our systematic review comprises a data collection stage that aims to answer the following research questions:

1) According to the classification proposed by Burke et al. [21], What kind of HHs have been implemented to solve SPs?
2) What strategies have been used to refine HHs in the context of SPs?
3) What types of SPs are currently being addressed using hyper-heuristics?
4) Are single- or multiple-objective HHs more prevalent in solving SPs?
5) Within the context of SPs: what sub-areas implement hyper-heuristics?
6) What are the primary optimization objectives commonly targeted in solving SPs using HHs?

The rest of this work is organized as follows: Section II presents some concepts necessary to understand better the analysis performed in this work. Section III explains the methodology we followed. Section IV shows the resulting data and analysis. Finally, Section V summarizes this work and outlines future research paths.

II. FUNDAMENTALS

This section presents fundamental concepts about hyper-heuristics and scheduling problems. Notwithstanding, we do not cover basic computation concepts, *e.g.*, NP-hard, problem-constraints, and optimization. For an in-depth description of computational theory, we refer the reader to [2], [22], and [23].

A. HEURISTICS

Heuristics are low-level solvers that operate using previous knowledge about the problem. Hence, they might contain multiple rules and may include an adaptive nature [24]. Heuristics can be classified into two groups: perturbative and constructive. On the one hand, the former modifies an existing solution to the problem. For example, in a job shop, a perturbative heuristic might take an already finished schedule and switch the first two activities of the machine with the largest makespan. On the other hand, constructive heuristics build a solution from scratch. Therefore, a constructive heuristic begins with an empty schedule and allocate activities with, *e.g.*, the shortest processing times.

In terms of literature, some works stand out. For example, Misir et al. studied the effect of low-level heuristics on the performance of selection hyper-heuristics [25]. Moreover, the literature also contains efforts to summarize heuristic contributions. An early example is Blackstone's work, which surveyed and categorized dispatching rules for the JSSP [26]. More recently, Van Hoorn analyzed the current bounds on benchmark instances for Job Shop Scheduling Problems (JSSPs) [25]. Also, Branke et al. stated a review on automated design of production scheduling heuristics [27].

B. METAHEURISTICS

Metaheuristics (MHs) are high-level algorithms that do not guarantee the discovery of an optimal solution but aim to find suitable solutions within a reasonable time frame. They mimic a natural phenomenon using an encoding scheme for representing a solution and use different operators, such as crossovers, mutations, selections, and swaps, to systematically alter the solution (*i.e.*, they operate in the solution space). To avoid becoming stuck at local minima, these search operators aim to identify high-quality answers by diversifying their search, intensifying their search, or exploring new candidate solutions. Thus, an effective algorithm balances intensification with diversification. Consequently, MHs are adept at exploring complex, multi-modal landscapes common in many real-world problems [28].

Most of the well-known metaheuristics are inspired by nature. Yet, some have found inspiration in several kinds of processes, such as those related to physics, chemistry, social sciences, and sports, among others [29]. Genetic Algorithms (GA), Simulated Annealing (SA), and Ant-Colony Optimization (ACO) are examples of nature-inspired algorithms. In contrast, the Gravitational Search Algorithm (GSA) is an example of a physics-based algorithm. Furthermore, the Chemical Reaction Optimization (CRO) algorithm and the Imperialist Competitive Algorithm (ICA) are metaheuristics inspired by Chemistry and social dynamics, respectively. For a deeper understanding of each MH, we suggest reading the work by Ezugwu et al., who presented a survey on these algorithms [29].

In addition, multi-objective algorithms are designed to address problems involving multiple, often conflicting, objective functions. Multi-objective MHs are specifically tailored to generate a set of optimal solutions, known as Pareto-optimal solutions. These solutions form the Pareto front, which represents the trade-offs between the competing objectives, where no other solution is superior in all objectives simultaneously. A solution is considered Pareto-optimal if no solution improves one objective without degrading another [30], [31]. Some examples are Multi Objective Genetic Algorithm (MOGA) by Murata and Ishibuchi [32], Non-dominated Sorting Genetic Algorithm (NSGA-II) by Deb et al. [33], Strength Pareto Archive Algorithm (SPEA2) by Zitzler et al. [34], and Multi-Objective Evolutionary Algorithm based upon Decomposition (MOEA/D) by Zhang and Li [35].

C. HYPER-HEURISTICS

Hyper-Heuristics (HHs) were introduced in 1997 to describe a system that combines several methods [14]. Notwithstanding, the term was used to describe ''heuristics to choose heuristics'' some years later [36]. HHs have been used to tackle various problems through selection and generation strategies. For example, Sim and Hart [37] described an immune-inspired hyper-heuristic system that produces new heuristics for the bin-packing and job-shop scheduling problems. Nguyen et al. proposed a genetic programming-based HH approach for three combinatorial optimization problems, *i.e.*, Max-SAT, one-dimensional bin packing, and permutation flow shop [38]. Another example is the Monte Carlo tree-search HH presented by Sabar and Kendall [39], which evolves heuristics that work on five problems: Max-SAT, one-dimensional bin packing, permutation flow shop, traveling salesperson, and personnel scheduling.

In addition to the definitions and examples mentioned above, there are two broad classes of hyper-heuristics [21]:

- *Generation*: HHs create new rules or adaptive heuristics for solving the problem instances.
- *Selection*: HHs pick heuristics from a set of existing heuristics.

Moreover, since heuristics are also sorted into two categories (*i.e.*,constructive or perturbative), we end up with a broader classification for HHs: Generation Constructive, Generation Perturbative, Selection Perturbative, and Selection Constructive [40]. Do note that such schemes for representing HHs are constantly evolving. For example, Burke et al. added a classification based on the feedback obtained from the evolution of the problem solution [40]:

- *Online Learning:* The model learns while solving each instance, and instances may arrive at different times.
- *Offline Learning:* The instances are given beforehand and split into training and testing sets. Learning results from continuously solving the training set, seeking to improve the model after each solution.
- *No Learning:* There is no feedback from the problem that guides the algorithm to adapt and improve.

It is relevant to mention that Drake et al. recently expanded upon these categories, providing perspectives related to the feedback, the number of objectives, and the way parameters are set, among others [36].

D. SCHEDULING PROBLEMS

Scheduling is a decision-making process used regularly in many manufacturing and service industries. It allocates resources to tasks over given periods and aims to optimize one or more objectives, *i.e.*,makespan, tardiness, and lateness [2].

Scheduling Problems (SPs) were initially labeled such as manufacturing problems [41]. Nowadays, there are many fields in which they are used. In healthcare, for example, effective scheduling is essential for managing the allocation of medical staff and equipment, directly impacting patient care and hospital throughput. Universities rely on scheduling to efficiently manage class timetables, examination periods, and facility utilization, facilitating smooth academic operations. Additionally, in the entertainment industry, such as film and television production, scheduling is crucial for coordinating the availability of cast, crew, and locations to ensure that productions are completed on time and within budget. Furthermore, as industries continue to evolve and new complexities arise, the development and refinement of

scheduling techniques remain an ongoing area of research and innovation.

Several studies have explored various scheduling problems. For example, Babou et al. addressed the Open Shop Scheduling Problem (OSSP) by incorporating a preparation phase distinct from the processing phase [42]. Recently, Planinic et al. proposed a two-part simplification method to address the bloating issue in evolving dispatching rules for the Parallel Machine Scheduling Problem (PMSP) [43]. Furthermore, Habibi et al. [44] reviewed the Resource-Constrained Project Scheduling Problem (RCPSP), evaluating 217 articles published between 1980 and 2017. They categorized the literature into four primary areas: characteristics, certainty, objective function, and resources. Lastly, Escott et al. used a transfer learning assisted Genetic Programming Hyper-heuristic (GPHH) for solving the Dynamic Multi-Workflow Scheduling Problem. They maintained performance and diminished computational costs by applying transfer learning to the genetic programming aspect of their hyper-heuristic model [45].

The primary optimization objectives in scheduling problems typically focus on:

1) **Minimizing Makespan:** Reducing the total time required to complete all tasks or jobs.
2) **Minimizing Tardiness:** Ensuring tasks or jobs are completed by or before their due dates. Hence, this metric only considers positive values for the difference between a job's completion and due dates. In other words, for this metric, it is equivalent if a job finishes with plenty of leeway or finishes in the nick of time.
3) **Minimizing Lateness:** Akin to the previous metric, but also considering negative values. Hence, this metric favors jobs that finish ahead of time.

These objectives drive the development of scheduling solutions tailored to meet specific operational efficiencies required in various industries [46]. Moreover, machine-based SPs usually exhibit one or more of the following constraints:

- *Precedence:* An operation o_l from job J_k cannot be processed until the previous operation o_{l-1} has been completed.
- *Capacity:* Each machine performs one activity at a time. This means that each machine can only process a single operation of a single job simultaneously.
- *Non-Preemption:* Operations must be completed once started.

Following, we present the concepts of some of the most relevant SPs.

1) JOB SHOP SCHEDULING PROBLEM

The Job Shop Scheduling Problem (JSSP) is an NP-hard optimization challenge [47] with a set of jobs $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$, each consisting of a sequence of operations $\mathcal{O} = \{o_1, o_2, \ldots, o_l\}$. Such operations must be processed on specific machines $\mathcal{M} = \{m_1, m_2, \ldots, m_i\}$. Additionally, each operation has a processing time $p_{k,l}$ that requires a particular machine to complete without interruption. The primary goal is typically to minimize the makespan, *i.e.*, the total time required to complete all jobs. Other less frequent objectives aim to minimize tardiness or machine idle times. The JSSP is highly applicable in manufacturing. It assigns multiple tasks across various machines while optimizing production efficiency and resource utilization. Due to its NP-hard nature, exact solutions are computationally infeasible for large instances, prompting heuristic and metaheuristic approaches to find satisfactory solutions within reasonable time frames [48].

Multiple kinds of Job Shop Scheduling Problems can be classified according to different characteristics [49]. For example, in the traditional JSSP, each operation must be performed in a specific machine, and only one operation of a given job can be performed at a time. The objective is to find a schedule that minimizes the makespan. Figure 1 shows a traditional JSSP instance distributed in two matrices (**M** and **P**). In both matrices, rows represent jobs (from j_1 to j_k) and columns represent operations (from o_1 to o_l). **M** contains an allocation of machines (from m_1 to m_i) to operations *i.e.*, which machine should perform each operation. **P** contains the information about the processing time ($p_{k,l} \in P$), *i.e.*, the time each operation takes to complete.

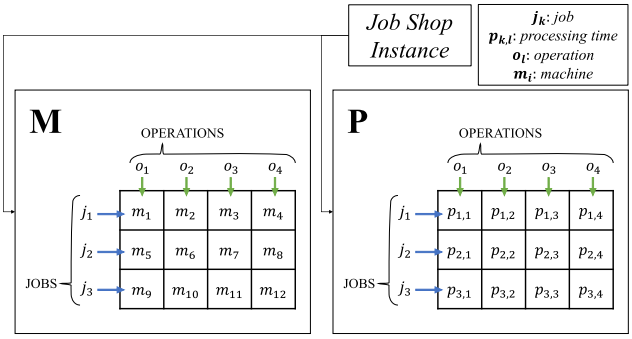

FIGURE 1. Illustrative example of a JSSP instance and the number of variables it contains.

2) DYNAMIC JOB SHOP SCHEDULING PROBLEM

The Dynamic Job Shop Scheduling Problem (DJSSP) extends the classical JSSP by incorporating elements that change over time, *e.g.*, jobs arriving over time, machine breakdowns, or changes in job priorities [50]. This brings the problem closer to reality and makes it more complex. For DJSSP, there is set of jobs $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$, which can be either finite or not. Every job j has an arrival time α_j, a due date d_j, and a weight w_j, which specifies the importance of the job. Besides, each job is comprised by sequences of operations $\mathcal{O} = \{o_1, o_2, \ldots, o_l\}$, where each operation has its own due date α_o and must be processed on specific machines $\mathcal{M} = \{m_1, m_2, \ldots, m_i\}$. Moreover, the dynamic nature provides an extra constraint: operation $o_{k,1}$ can only be scheduled once job j_k has arrived at the job shop. The primary

objective remains to minimize the makespan. Still, due to its dynamic nature, objectives such as minimizing total tardiness or maximizing responsiveness (quick adaptation to changes) become equally meaningful.

The DJSSP finds significant application in industries where production conditions are volatile and subject to frequent change, such as manufacturing and logistics [51]. Given the problem's complexity and NP-hard nature, solutions typically rely on advanced heuristic and metaheuristic methods, for they can adapt quickly to changes while providing feasible scheduling solutions. This can be attributed to the disruptions, delays, and real-time scheduling adjustments.

3) FLEXIBLE JOB SHOP SCHEDULING PROBLEM

The Flexible Job Shop Scheduling Problem (FJSSP) is an extension of the JSSP that introduces an additional layer of flexibility in machine assignments for operations [52]. The core formulation remains: a set of jobs $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$, each consisting of a sequence of operations $\mathcal{O} = \{o_1, o_2, \ldots, o_l\}$. Nevertheless, unlike the traditional JSSP where each operation is assigned to a specific machine, in the FJSSP, each operation can be executed on any machine from a given subset of machines $\mathcal{M} = \{m_1, m_2, \ldots, m_i\}$. Machine constraints from JSSP are equally applied to this problem. Moreover, common objectives include minimizing both the makespan and the tardiness. However, the nature of this variation leads to an extra objective: maximizing machine utilization. Finally, this problem is relevant in flexible manufacturing systems, where machines can perform different operations. Another application relies on workshops with pools of parallel machines, where multiple identical machines can perform only one kind of operation.

4) DYNAMIC FLEXIBLE JOB SHOP SCHEDULING PROBLEM

The Dynamic Flexible Job Shop Scheduling Problem (DFJSSP) combines both the flexible nature of FJSSP and the dynamic nature of DJSSP [53]. Therefore, it contains a set of finite or undetermined jobs $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$, where each job has an arrival time α_j, a due date d_j, and a weight w_j which specifies the importance of the job. Each job consists of a sequence of operations $\mathcal{O} = \{o_1, o_2, \ldots, o_l\}$, and each operation has a subset of available machines from the set $\mathcal{M} = \{m_1, m_2, \ldots, m_i\}$. As in DJSSP, operation $o_{k,1}$ can only be scheduled once job j_k arrives at the job shop. The main objectives of DFJSSP are minimizing both makespan and tardiness and maximizing machine utilization. Health systems are highly relatable to DFJSSP, where there is an undetermined set of patients (jobs) with different arrival times (dynamic nature). Each patient needs a different set of procedures (*e.g.*, operations), which can be performed by many identical machines (*e.g.*, doctors).

5) FLOW SHOP SCHEDULING PROBLEM

The Flow Shop Scheduling Problem (FSSP) is a specialized variant of the JSSP where each job has to get through the same sequence of machines [54]. Therefore, it has a set of jobs $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$, where each job consists of a sequence of operations $\mathcal{O} = \{o_1, o_2, \ldots, o_i\}$, which must be performed in a fixed set of machines $\mathcal{M} = \{m_1, m_2, \ldots, m_i\}$. The sequential nature of this problem rests in the distribution of machines, where machine m_i has to process operation o_i, machine m_{i-1} has to process operation o_{i-1}, and so on. Although each job must be processed through the same machine sequence, the processing times may vary. The complexity of this problem is NP-hard for more than two machines. However, for two machines, we can find the optimal makespan in polynomial time using Johnson's rule [55]. Similarly to JSSP, the most common objective of FSSP is to minimize the makespan. FSSP is present in most assembly industries, where the final product follows a sequential process [56].

6) PERMUTATION FLOW SHOP SCHEDULING PROBLEM

The Permutation Flow Shop Scheduling Problem (PFSSP) is a variant of FSSP, where the sequence of jobs for the first machine (m_1) has to remain for all subsequent machines [12]. Consequently, PFSSP is represented by a set of jobs $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$, where each job consists of a sequence of operations $\mathcal{O} = \{o_1, o_2, \ldots, o_i\}$, carried out in a set of machines $\mathcal{M} = \{m_1, m_2, \ldots, m_i\}$ in sequential order (see Section II-D5). Additionally, the distribution of jobs for m_1 is inherited by all machines. This constraint simplifies the problem's solution space. However, its complexity remains unchanged (NP-hard) [57]. Similarly to FSSP, Johnson's rule applies to the case of two machines. The automotive industry is an excellent example of an application area for PFSSP.

7) OPEN SHOP SCHEDULING PROBLEM

Open Shop Scheduling Problem (OSSP) is another variation of the JSSP [48], but contrary to JSSP, in OSSP operations are processed in arbitrary order, *i.e.*, the precedence machine constraint does not apply. This means that a set $\mathcal{M} = \{m_1, \ldots, m_i\}$ of i machines process a set $\mathcal{J} = \{j_1, \ldots, j_k\}$ of k jobs, and that each job has l operations from o_1 to o_l. An unfixed set of operations adds another decision layer to the problem. Thus, the complexity increases to being NP-hard even in the case of two machines [42]. A real-life scenario of this problem is found in computing environments, particularly in cloud or distributed computing, where tasks may be processed on different servers or processors without a predefined sequence.

8) PARALLEL MACHINE SCHEDULING PROBLEM

The Parallel Machine Scheduling Problem (PMSP) involves assigning a set of jobs to parallel, identical, or non-identical machines to optimize specific criteria, such as minimizing the makespan [58]. The PMSP might seem similar to the FJSSP, but the latter allows for different kinds of machines that may or may not have replicas. Another key difference is that the PMSP considers jobs as a whole, not as a set of operations.

Thereupon, this problem can be modeled as follows: a set $\mathcal{J} = \{j_1, \ldots, j_k\}$ of k jobs that must be processed by a set $\mathcal{M} = \{m_1, \ldots, m_i\}$ of i parallel machines. Each k job has a known integer processing time p_k and a known loading time s_l associated with a machine i (also called a setup). Such setup is handled by a server, which may be one or more for all machines, each with different availability; in some cases this setup time is included in the processing time of each job. Additionally, there are three variants of parallel machines [5]:

1) Identical machines: They have the exact same capabilities.
2) Uniform machines: Machines have different capabilities, usually determined by a speed factor that affects all jobs.
3) Unrelated machines: Each machine-job pairing has its unique processing time depending on the nature of each machine.

The complexity of PMSP is NP-hard for all cases. However, Kravchenko proposed a pseudo-polynomial algorithm for the case of two machines where all setup times are equal to one [43].

This problem formulation is easily seen in different areas characterized by homogeneous tasks or uniform machine functionalities. In the healthcare sector, optimized scheduling of imaging equipment such as computed tomography (CT) and magnetic resonance imaging (MRI) machines is crucial for enhancing patient throughput and optimizing resource allocation [59]. In manufacturing, parallel scheduling — applied to identical machines performing similar assembly tasks— significantly improves production line efficiency by minimizing idle times and synchronizing output rates [60]. Similarly, effective scheduling algorithms distribute tasks across multiple processors in computational tasks within data centers, thereby enhancing computational output [61].

9) TIMETABLING PROBLEM

Timetabling Problem (TTP) is a combinatorial problem of high interest in recent years [62]. It has been proven to be NP-hard due to its many constraints and requirements [47]. There are several variants of timetabling, each one with its specific mathematical formulation. However, a general approach consists of a set of k events $\mathcal{E} = \{e_1, \ldots, e_k\}$, where each event has a duration d from the set $\mathcal{D} = \{d_1, \ldots, d_k\}$. Similarly, there is a set of i resources $\mathcal{R} = \{r_1, \ldots, r_i\}$. All events from $\mathcal{E}$ must be scheduled over a set of spaces $\mathcal{S} = \{s_1, \ldots, s_l\}$ and a set of time-slots $\mathcal{T} = \{t_1, \ldots, t_n\}$. The objective is to find an optimal allocation of the given set of events (*e.g.*, courses, exams, surgeries, and sports) and resources (*e.g.*, teachers, exam proctors, nurses, medical doctors, and referees) over spaces (*e.g.*, classrooms, operating rooms, and sports fields) and time [63]. An optimal allocation satisfies the following objectives:

- Resources attend only one event at a time.
- Events must be scheduled in spaces that meet its requirements.
- Spaces can host only one event at a time.
- Resources attend only one event at a time.

Other objectives may be specific to the variant, *e.g.*, schedule preferences for resources, and minimimum empty time slots. This challenge is expected to academic institutions such as schools, colleges, or universities [64]. One type of TTP is educational timetabling, from which three main variations focus on courses, examinations, and schools, respectively [65]. The course timetabling problem involves assigning a set of courses to finite time slots and featured rooms [66]. Similarly, in the examination timetabling problem, one must assign a set of exams to a set of time slots and available rooms [67]. Other examples include the school timetabling [68] and staff timetabling [69] problems. In 2010, Burke et al. conducted a survey of search methodologies and automated system development for examination timetabling [70]. Their work comprehensively surveys various methodologies applied to examination timetabling, including heuristic, MH, and HH approaches.

10) PROJECT SCHEDULING PROBLEM

The Project Scheduling Problem (PSP) involves the allocation of resources to tasks or activities that need to be completed within a project to optimize certain performance indicators such as minimizing the total project duration (makespan), adhering to deadlines, or minimizing costs [71]. In the PSP, a project is denoted as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $n = 0, 1, 2, \ldots, N + 1$ is the node set of the project, and there are N real activities [72]. Activity 0 and activity $n + 1$ are the start and end dummy activities, respectively. $\mathcal{A}$ is the set of arrows formed by the priority relationship between activities. Each activity needs $k = 1, 2, 3, \ldots, K$ renewable resources. The duration of each activity and its resource consumption are recorded as d_i and $r_{i,k}$, respectively. We can denote the project deadline as $\mathcal{D}$, and the difference between the project makespan and the deadline as $\mathcal{E}$. The objective is for $\mathcal{E}$ to be equal or less to zero so that the project occurs in time or early completion. Otherwise, it incurs delay damages.

11) RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

The Resource-constrained Project Scheduling Problem (RPSP) is a variant of PSP with limited resources. Thus, RPSP may be formulated as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, with the same definitions as for the PSP. Additionally, the RPSP has a set $\mathcal{R} = \{r_1, \ldots, r_K\}$ of K resources, where R_k is the available amount of resource k. The objective is to complete all activities, following all precedence and resource constraints, while minimizing the difference between the project makespan and the project deadline.

12) WORKFLOW SCHEDULING PROBLEM

The Workflow Scheduling Problem (WSP) involves arranging and managing tasks or activities within a workflow

system to optimize specific performance metrics such as throughput, execution time, or resource utilization [2]. A workflow is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of v nodes and $\mathcal{E}$ is the set of e edges that connect nodes [73]. A node $v_i \in \mathcal{V}$ represents a task t_i, and a node $e_{i,j} \in \mathcal{E}$ represents precedence dependency between task t_i and task t_j. Therefore, task t_j can be executed until task t_i is finished. The weight of edge $e_{i,j}$ represents the transfer time between t_i and t_j. The set P is the set of computing resources for executing the tasks. The WSP is a somewhat simplified version of project scheduling with no overall deadline. This makes it suitable for many applications where optimizing complex sequences of operations and enhancing efficiency is critical. For instance, each patient encounter in healthcare may be viewed as a distinct workflow, *i.e.*, from initial admission and diagnostic tests through surgeries and other procedures. Managing these multiple and concurrent patient workflows is essential for improving care quality and hospital efficiency [59]. In the media production industry, workflow scheduling orchestrates the logistics of film production, coordinating crew availability and post-production processes to streamline operations and reduce downtime [74]. Lastly, workflow scheduling is implemented in manufacturing to synchronize assembly lines and maintenance routines, significantly enhancing production throughput and reducing operational costs [75].

13) DYNAMIC WORKFLOW SCHEDULING PROBLEM

The Dynamic Workflow Scheduling Problem (DWSP) is a more complex variant of the WSP, where the system must adapt to changes, including new arrivals of finite or undetermined workflows and variations in resource capacity, *e.g.*, processors breaking down. This problem is highly present in computing environments, where workflows keep arriving, and a server must allocate them to the available processors. Similarly to the WSP, in the DWSP, a workflow is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of v nodes and $\mathcal{E}$ is the set of e edges that connect nodes. A node $v_i \in \mathcal{V}$ represents a task t_i, and a node $e_{i,j} \in \mathcal{E}$ represents the precedence dependency between task t_i and task t_j. However, new tasks arrive, and resources from P may vary over time [76].

III. METHODOLOGY

This work examines recent advancements in hyper-heuristics within the context of scheduling problems, identifying emerging research opportunities at their intersection. We analyze the current state of the art, extract critical features, and align these findings with high-impact applications. Our analysis begins with an overview of leading research topics in scheduling problems and hyper-heuristics, focusing on publications where hyper-heuristics is the primary strategy. We systematically categorize all selected works to address six fundamental research questions, identifying prevailing trends and gaps and providing a qualitative analysis of potential future research directions for hyper-heuristic methodologies. Therefore, our methodology comprises three phases:

1) Inclusion, search, and selection
2) Systematic classification
3) Analysis about trends and relations

This study is designed with a reproducible process, beginning with a quantitative data collection and shifting towards a deductive data analysis. Finally, our process led to 215 documents covering a 10-year window from 2012 to 2022.

A. INCLUSION, SEARCH, AND SELECTION

The initial phase of our research involves selecting publications. We start by conducting a broad literature search on scheduling problems and Hyper-Heuristics (HHs), utilizing Elsevier's Scopus database. Our choice of Scopus is motivated by several factors:

1) **Comprehensive Coverage:** Scopus provides a broad overview of scientific research from different sources worldwide, which is crucial for building the current state of the art.
2) **Analytical Capabilities:** The platform offers built-in analytical tools that facilitate the identification of emerging trends.
3) **Accessibility:** We have access to the database thanks to institutional subscriptions.

The query begins with several search equations involving words like 'scheduling,' 'hyper-heuristic,' and 'heuristic algorithms.' However, all iterations yielded a similar number of results, and so we opted for a simple equation:

```
hyper-heuristics AND scheduling    (1)
```

Such an equation returned 378 results, for which we downloaded the corresponding metadata. Our inclusion criteria were based on two primary parameters: publication year and hyper-heuristics relevance. We limited our selection to publications within ten years, from 2012 to 2022, to ensure contemporary relevance. Subsequently, we meticulously examined the titles and abstracts to assess the centrality of hyper-heuristics in each study, keeping only those works where hyper-heuristics were the primary focus. This rigorous selection process yielded a dataset of 215 documents.

B. SYSTEMATIC CLASSIFICATION

The second phase of our research involves systematically classifying the 215 selected works to address six specific research questions. We employed a structured approach to review each document, following this sequential order:

1) Abstract: Initial screening for relevance and key findings.
2) Figures and Tables: Detailed examination of visual data representations.
3) Results: In-depth analysis of the findings.
4) Methodology: Review of the research methods and procedures.

5) Conclusions: Summary of the results and implications of the study.

We meticulously reviewed each section to ensure comprehensive coverage of the research questions. In some cases, abstracts alone provided all the necessary information. However, other documents required a thorough examination in multiple sections. We continued this detailed review process until all pertinent information was extracted or until it was determined that the necessary data were unavailable, in which case we noted the information as missing. The following list contains the research questions that guided this phase:

1) According to the classification proposed by Burke et al. [21], what kind of HHs have been implemented to solve SPs?
2) What strategies have been used to refine HHs in the context of SPs?
3) What types of SPs are currently being addressed using hyper-heuristics?
4) Are single- or multiple-objective HHs more prevalent in solving SPs?
5) Within the context of SPs: what sub-areas implement hyper-heuristics?
6) What are the primary optimization objectives commonly targeted in solving SPs using HHs?

Each paper was analyzed and categorized into a six-fold framework based on the answers to these questions. Note that whenever a work tackled more than one type of data, *e.g.*, when tackling two or more scheduling problems, we counted it multiple times. The first research question leverages the well-established classification system by [21], which categorizes hyper-heuristics based on learning approach, heuristic set, and heuristic type (detailed in Section II-C).

The second question addresses complementary strategies used alongside hyper-heuristics, such as metaheuristics, systematic search algorithms, and learning algorithms. These algorithms can be used for various purposes, such as the training phase of a hyper-heuristic, creating new heuristics, or modifying existing solutions, to name a few. We further divided these into five sub-categories: evolutionary computation (genetic programming, genetic algorithms, and cooperative co-evolution algorithms), metaheuristics (excluding those related to evolutionary computation), systematic search algorithms (including Depth-First Search, Breadth-First Search, and Greedy Search but excluding metaheuristics like Simulated Annealing and Tabu Search), learning algorithms, and other strategies, which are outside of the scope of the previous ones, such as graph theory and Markov models.

The third question identifies the specific scheduling problem each study addresses, such as job shop, flow shop, workflow, or timetabling. Note that we categorized variations of scheduling problems, such as workflow in cloud computing and dynamic flexible job shop, according to their primary components, *i.e.*, as 'workflow' and 'job shop,' respectively. The fourth question determines whether the studies focus on single or multiple objective functions. The fifth question explores each study's application areas, including sectors like manufacturing, healthcare, and cloud computing. If the paper does not show an apparent field of application, we assume it focuses on basic scientific research. Lastly, the sixth question examines the optimization objectives targeted by each work, such as makespan or tardiness.

C. RELATIONS AND TRENDS

In this phase, we aim to detect underlying patterns and relationships within our dataset. We employed a methodical approach to analyze the data pairwise. This involved examining how different variables interact and their application to various scheduling problems, such as specific types of hyper-heuristics, *e.g.*, selection, or generative. For example, we assessed the frequency of combining selection hyper-heuristics with constructive heuristics across different studies.

Our analysis extended to evaluating how often certain combinations appear, which provided insights into the most used strategies currently employed in the field. Moreover, we selected the most salient trends and applied statistical tools and visualization techniques to understand these relationships better. This structured analytical approach enabled us to understand the current landscape in hyper-heuristic research as applied to scheduling problems. Our findings are poised to inform future research directions, highlighting effective strategies and areas that require further investigation.

IV. RESULTS

This section presents our findings. For readability, we preserve the same structure from Section III.

A. INCLUSION, SEARCH, AND SELECTION

Using the search query `hyper-heuristics AND scheduling`, we initially obtained nearly 400 results. After we applied our inclusion criteria, we refined the dataset to 215 relevant entries. Our first data analysis focuses on the number of publications per year, as shown in Figure 2. This figure illustrates a growing trend in the volume of research, peaking in 2019 with 35 manuscripts. This highlights the research community's interest in utilizing hyper-heuristics as a tool for solving scheduling problems. Given their intrinsic compatibility, we believe the growing interest in hyper-heuristics (HHs) and scheduling problems (SPs) is well-founded. Scheduling problems demand quick, practical solutions, and hyper-heuristics are uniquely suited to deliver these efficiently.

Following, we present key points about the more than 400 authors responsible for these publications. First, Figure 3 displays the top 12 authors by number of publications. Notably, the leading author, Zhang M., has contributed nearly 30 publications, averaging about three publications

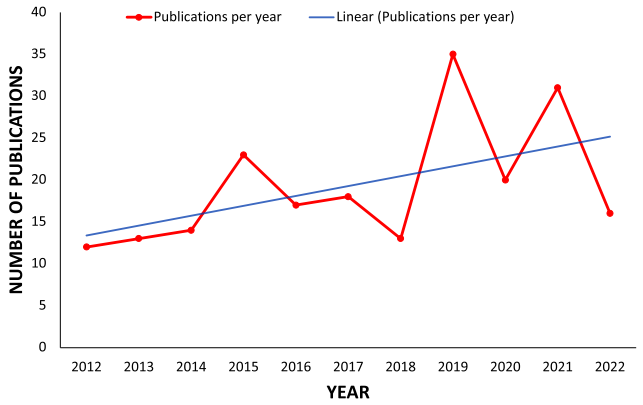

FIGURE 2. Behavior of yearly publications about hyper-heuristics involved in solving scheduling problems over the last ten years.Red and blue strokes are for the acquired data and their linear trend, respectively.

per year. In comparison, the second-ranked author, Mei Y., has 19 publications, approximately one-third fewer than Zhang M. Both these authors have collaborated in some of the papers, and both primarily focused on Genetic Programming-based Hyper-heuristics (GPHH) to address various scheduling problems. For instance, in collaboration with Yang et al. [77] they employed a GPHH methodology to tackle the workflow problem, aiming to minimize overall costs and makespan while adhering to Service Level Agreement (SLA) requirements. Conversely, together with Zhang F. and Nguyen S., they adopted an evolutionary multi-tasking approach to solving multiple dynamic, flexible job shop scheduling problems using a GPHH method [53]. Moreover, they worked in a knowledge transfer approach for implementing GPHHs in multiple problem domains, including scheduling problems [78].

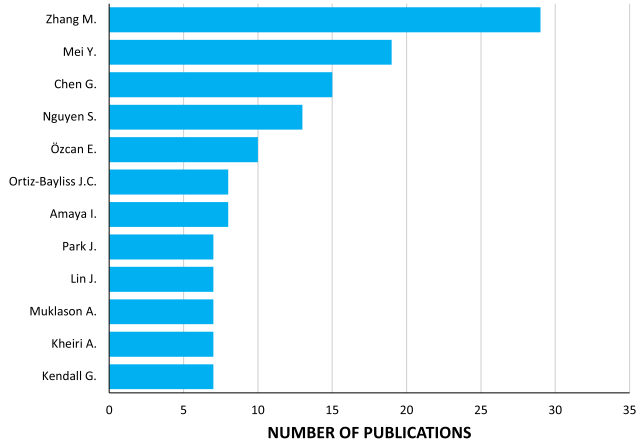

FIGURE 3. Top 12 authors in the number of published works from 2012-2022 in Scopus.

The third and fourth authors (Chen G. and Nguyen S.) have 15 and 13 publications, respectively. Similarly to the previous authors, they have extensively utilized Genetic Programming-based Hyper-heuristics (GPHH). Additionally, the following three authors, Özcan E., Ortiz-Bayliss J.C., and Amaya I., published between eight and ten articles focusing on selection hyper-heuristics for various scheduling problems. The remaining authors in the top 12 brandish seven publications each [21], [36]. Looking at the affiliations of these authors, half of them are affiliated with institutions in New Zealand, including the top four.

Considering all authors, Figure 4 illustrates their distribution based on their publication output. We can easily see that most authors have published only one paper. Notably, those with between one and three publications account for over 90% of the total data.

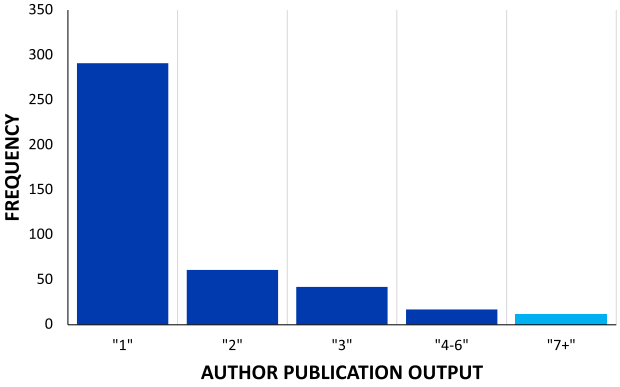

FIGURE 4. Number of authors grouped by different publication outputs. The x-axis represents different categories of publication output, while the y-axis indicates the number of authors falling into each category.Note that the last category groups the data shown in Figure 3.

The number of citations of each author is just as important as their number of publications. Figure 5 provides the top 11 authors ranked by their citation numbers. There is a notorious correlation between the number of publications and the number of citations, as over 60% of the names from this figure also appeared in Figure 3. Exceptions to this trend, *i.e.*, authors among the top in published articles but not in citations, include Chen G., Park J., Ortiz-Bayliss J.C., and Amaya I. Conversely, authors such as Qu R., Sabar N.R., Ayob M., and Hildebrandt T., despite publishing fewer than seven papers during the observation period, are among the most cited. Additionally, we would like to highlight a couple of notable contributors. Zhang M. is the leading author in terms of both the number of publications and citations. Kendall G., in contrast, has achieved nearly 350 citations with just seven publications.

Finally, a closer look at these publications reveals that most publications involve Victoria University in New Zealand, which makes it the leading entity on the topic. It is followed by the University of Nottingham in England and Tecnológico de Monterrey in Mexico.

B. SYSTEMATIC CLASSIFICATION

We now jump to the data extracted from the papers, aiming to answer the research questions laid out in Section III-B.

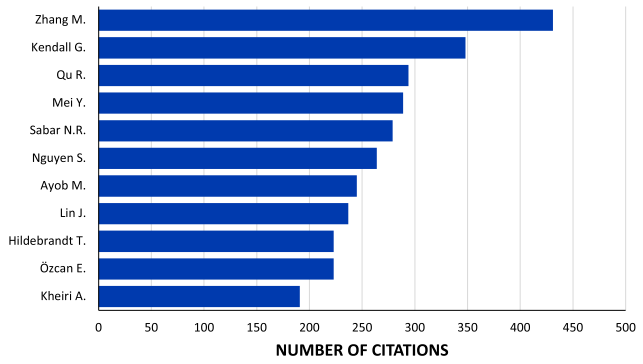

FIGURE 5. Top 11 authors regarding the number of citations from 2012-2022 in Scopus.

The first research question addresses the classification of hyper-heuristics. We observe how scientific reports distribute over the three-fold classification of hyper-heuristics by Burke et al. [21]. Figure 6 shows that most works do not provide information about the training process of their hyper-heuristics. However, from those that do, there has been a slight preference for offline training methods. This preference is reasonable, given that offline methods are typically simpler and more stable. Notwithstanding, research should aim towards online methods due to their greater applicability to real-world scenarios. Similarly, the majority of publications favor the selection of pre-defined heuristics over the generation of new ones. Additionally, there is a notable preference for perturbative heuristics, which are featured in over 65% of the analyzed works. This trend is logical given that many of these studies are grounded in evolutionary computation, which incorporates perturbation strategies to enhance solution quality.

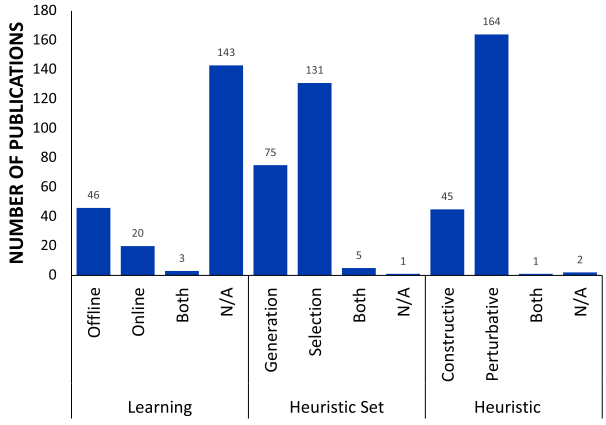

FIGURE 6. Classification of the hyper-heuristics used across all the 215 reviewed papers, following a three-fold classification as proposed by [21]. Learning refers to the learning process used to enhance the hyper-heuristic.A heuristic set refers to how the hyper-heuristic uses existing heuristics, and a heuristic refers to the nature of the available heuristics.

The second research question aims to examine the strategies used to power hyper-heuristics. Figure 7 depicts that evolutionary computation, metaheuristics, search algorithms, and learning algorithms are the most common complementary strategies, representing nearly 90% of publications. Most publications employed these methods for generating new heuristics or selecting the most effective heuristics for specific scheduling problems, particularly evolutionary computation and metaheuristics. The remaining 10% of publications, classified as `Other`, include less common strategies that appear in less than 1% of the works, such as simple heuristics, graph theory, and Markov models.

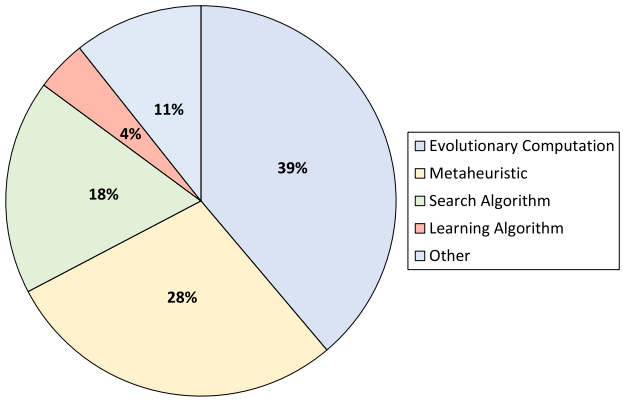

FIGURE 7. Distribution of employed strategies for powering hyper-heuristics to solve scheduling problems across 215 reviewed papers from the last 10 years.

The third research question focuses on the distribution of scheduling problems. Our systematic process identified over 100 distinct scheduling configurations. This many SPs come from adapting such a scheduling scheme to a specific scenario, for example, hospitals, schools, industries, etc. For the sake of simplicity, we grouped variations of the same fundamental problem —like flexible job shop and dynamic job shop— under a single category. Figure 8 shows that `job shop` and `timetabling` are currently the first and second most popular SPs targeted by HHs. Following these are `flow shop`, `project scheduling`, and `workflow`. The `Other` category encompasses SPs below ten publications, including specialized areas like nurse rostering, movie scene scheduling, and online scheduling. There is no surprise that job shop, a standard within scheduling problems, continues to be a primary focus of research.

The fourth research question examines the contrast between single- and multiple-objective-oriented publications. Our analysis reveals that single-objective functions predominate, representing 67% of the publications, as illustrated in Figure 9. The predominance of single-objective publications in scheduling problem research can be attributed to several factors. First, single-objective functions offer simplicity and clarity, facilitating easier problem formulation. Specifically, objectives like makespan are straightforward and provide a clear metric to validate the feasibility of a schedule. Additionally, multi-objective optimization exhibits

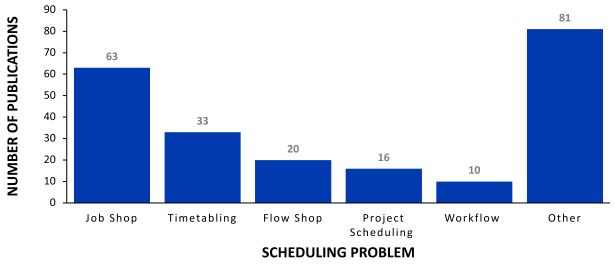

FIGURE 8. Number of publications per type of scheduling problem in the past 10 years.

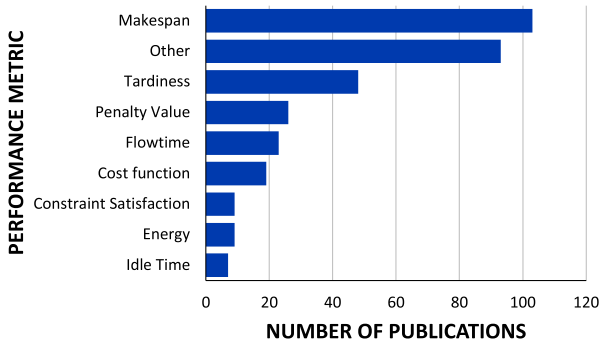

FIGURE 10. Most used performance metrics across all the 215 reviewed papers.

methodological challenges, especially when developing new heuristic methods, which may favor the less complex single-objective approach.

The next question strives to find the most relevant objectives. Figure 10 highlights the ones our system found across all entries. In particular, makespan emerges as the most preferred objective, appearing in nearly 50% of the publications. This preference is well-founded, as makespan provides a direct measure of a schedule's total completion time, a critical factor in many operational contexts. The `Other` category encompasses less frequently used objectives, such as idle time, decision time, and error percentage, each below seven publications.

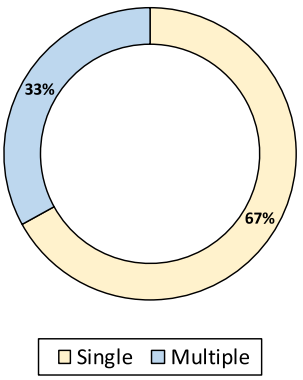

FIGURE 9. Distribution of documents in terms of the number of objectives associated with the objective function used during the training phase of the respective hyper-heuristic models.

Some studies on HH methods extend beyond theoretical research, demonstrating their application in real-world scheduling challenges. The final research question aims to identify the application areas targeted by each publication. Figure 11 shows a visual summary of our findings, where the size and frequency of words indicate the prevalence of each area within the publications. While a significant portion of the research remains theoretical, notable practical applications are seen in manufacturing and cloud computing, with 27 and 14 documents, respectively. Manufacturing emerges as the predominant field for applying scheduling solutions, largely due to the prevalence of JSS, a common scenario in this sector. Healthcare, fog computing, and aviation follow the list in terms of publication. Still, each of them only provide two entries. Thus, these applications lag significantly behind manufacturing and cloud computing. Additionally, fields like cloud computing, healthcare, and aviation demand exceptionally precise scheduling to ensure efficient operations. This underscores the critical need for robust scheduling approaches that can handle the complex requirements of these industries.

FIGURE 11. Most relevant applications across the analyzed papers.

C. RELATIONS AND TRENDS

This subsection highlights the most significant relationships and trends identified in our analysis outside the research questions. As stated in Section III-C, we examined data points in pairs to discern patterns in the application of hyper-heuristics. Our first standout involved the overlap of two classifications of hyper-heuristics: heuristic set (selection or generation) and heuristic classification (constructive or perturbative). According to Figure 12 the predominant combination involves selection hyper-heuristics with perturbative heuristics, with the next most common being generation hyper-heuristics paired with perturbative heuristics. The dominance of the perturbative attribute across both configurations can be largely attributed to the widespread use of evolutionary computation methodologies, which rely on perturbative strategies to evolve solutions progressively. Similarly, in Figure 13, we show the relation between HH classification and the type of SPs. A clear preference emerges for perturbative over generative approaches,

particularly noted in job shop and other scheduling domains. This indicates that perturbative hyper-heuristics, which refine solutions through iterative modifications, are highly valued for enhancing solution quality within scheduling environments. Furthermore, selection-based hyper-heuristics dominate generation-based strategies, suggesting that when an extensive pool of established heuristics is available, selecting the optimal heuristic is more effective and computationally efficient than generating new ones. These trends are consistent with the remaining scheduling problems: flow shop, workflow, and project scheduling. However, these specialized problems show fewer instances, hinting at potential gaps in the literature regarding applying current hyper-heuristic frameworks to these areas. Moreover, this analysis hints that expanding the application of generative and innovative heuristic strategies could address existing deficiencies and foster advancements in the field, ultimately leading to more robust and versatile scheduling solutions across a broader spectrum of real-world applications.

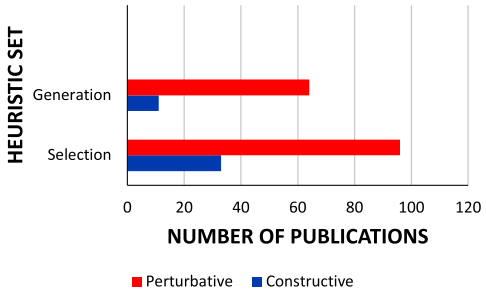

FIGURE 12. Relation between two of the hyper-heuristic classifications proposed by Burke et al. [21] over the examined documents.Selection or generation determines whether a hyper-heuristic selects heuristics or generates new ones. Perturbative or constructive dictates the nature of the heuristic: a constructive heuristic builds a solution from scratch, whereas a perturbative heuristic modifies an existing one.

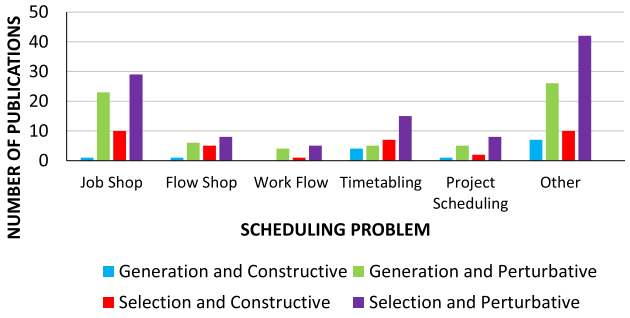

FIGURE 13. Distribution of manuscripts based on the classical grouping [21] and per scheduling problem.

As a second approach, we analyzed which algorithms are often used in scheduling problems. Figure 14 shows that for job shop (the most common scheduling problem), the preferred algorithms powering HHs are those within the scope of

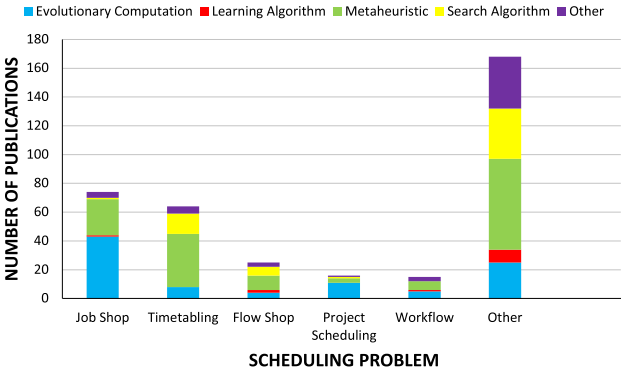

FIGURE 14. Distribution of the techniques used for powering hyper-heuristics, per scheduling problem, across all documents. The total exceeds 215 because certain documents targeted more than one scheduling problem.

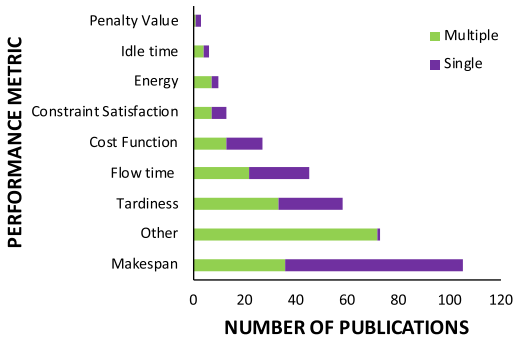

FIGURE 15. Overlapping the performance metrics and the objectives across all 215 papers. Totals do not add up to 215 because certain works used more than one performance metric.

evolutionary computation. On the other hand, metaheuristics and search algorithms are the preferred algorithms for timetabling (the second most common scheduling problem). For other scheduling categories, such as flow shop, project scheduling, and workflow, the choice of complementary strategies shows a more uniform distribution. Notably, project scheduling predominantly utilizes evolutionary computation, highlighting its suitability for the complex and dynamic nature of project management tasks. Meanwhile, metaheuristics dominate in the `Other` category, which encompasses less frequently studied scheduling problems, indicating their versatility and robustness in tackling a diverse array of less conventional scheduling challenges. These results suggest that algorithms within evolutionary computation and metaheuristics pair appropriately with HHs, reflecting both the adaptability and effectiveness of these computational strategies in optimizing scheduling processes.

Finally, we analyzed the correlation between the optimization objectives and their application within single or multiple-objective environments. We identified whether specific objectives were predominantly used in isolation or as part of a broader, multi-objective optimization strategy. Figure 15 reveals that researchers apply single and multiple approaches

at comparable rates for most objectives. However, makespan is an exception, predominantly employed in single-objective contexts. This preference likely stems from its widespread acceptance as a comprehensive measure of efficiency in scheduling problems, effectively demonstrating the success of solutions when applied within hyper-heuristic frameworks. Conversely, less common metrics such as performance score, decision time, and earliness are rarely utilized independently, indicating their limited applicability or specificity to certain scheduling challenges.

V. CONCLUSION

This work has meticulously evaluated 215 publications from 2012 to 2022, focusing on employing hyper-heuristics to scheduling problems. Our analysis involved a detailed evaluation of various aspects, including hyper-heuristic model classification, the complementary strategies employed, the types of scheduling problems addressed, and their performance metrics. Our findings not only underline the prevalence of job shop scheduling and timetabling as dominant research areas, which represent 45% of the documents, but also highlight the typical use of selection hyper-heuristics combined with perturbative heuristics. Evolutionary computation has emerged as a leading strategy, reinforcing its effectiveness in developing robust scheduling solutions.

The analysis revealed a pronounced concentration of research output from New Zealand, with significant contributions from researchers like Zhang M., emphasizing regional leadership in this domain. However, our review identified notable gaps, particularly in the diversity of hyper-heuristic configurations and the under-utilization of specific performance metrics like energy consumption. These areas offer fertile ground for future research, potentially driving advancements toward environmentally sustainable scheduling practices.

Furthermore, the predominance of traditional performance metrics, such as makespan and tardiness, alongside the limited number of documents for industries like aviation, education, and healthcare, suggests a critical avenue for expanding the applicability of these methodologies. By bridging the gap between theoretical research and practical applications, future studies could foster the development of innovative hyper-heuristic approaches that cater to the dynamic nature of modern scheduling needs.

In conclusion, while this field has seen considerable advancements, the evolving complexity of scheduling problems across various industries calls for continued innovation in hyper-heuristic strategies. Future research should aim to explore underrepresented areas, employ a more comprehensive array of performance metrics, and strengthen the connection between hyper-heuristic research and its real-world applications. This direction promises to enhance the efficiency and effectiveness of scheduling solutions and broaden hyper-heuristics' impact in solving real-world problems.

DATA AVAILABILITY

Data generated throughout this manuscript is available upon request.

ACKNOWLEDGMENT

The authors would like to acknowledge the assistance of OpenAI's ChatGPT and Grammarly tools for supporting grammar and orthography in preparing this manuscript [1].

REFERENCES

[1] OpenAI. (2023). *ChatGPT: Optimizing Language Models for Dialogue*. Accessed: May 9, 2023. [Online]. Available: https://www.openai.com/chatgpt

[2] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 5th ed., Cham, Switzerland: Springer, 2016.

[3] D.-Z. Chen, C. Wei, G.-L. Jia, and Z.-H. Hu, "Shortest-path optimization of ship diesel engine disassembly and assembly based on AND/OR network," *Complexity*, vol. 2020, pp. 1–15, Feb. 2020.

[4] W. Luo, R. Chin, A. Cai, G. Lin, B. Su, and A. Zhang, "A tardiness-augmented approximation scheme for rejection-allowed multiprocessor rescheduling," *J. Combinat. Optim.*, vol. 44, no. 1, pp. 690–722, Aug. 2022.

[5] J. K. Lenstra, D. B. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, vol. 46, nos. 1–3, pp. 259–271, Jan. 1990.

[6] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA, USA: Addison-Wesley, 1984.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.

[8] J. Kleinberg and E. Tardos, *Algorithm Design*. London, U.K.: Pearson, 2006.

[9] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[10] V. V. Vazirani, *The Design of Approximation Algorithms*. Cham, Switzerland: Springer, 2001.

[11] J. Dorn, M. Girsch, G. Skele, and W. Slany, "Comparison of iterative improvement techniques for schedule optimization," *Eur. J. Oper. Res.*, vol. 94, no. 2, pp. 349–361, Oct. 1996.

[12] E. H. L. Aarts, P. J. M. van Laarhoven, J. K. Lenstra, and N. L. J. Ulder, "A computational study of local search algorithms for job shop scheduling," *ORSA J. Comput.*, vol. 6, no. 2, pp. 118–125, May 1994.

[13] S. Z. A. Bacha, K. Benatchba, and F. Benbouzid-Si Tayeb, "Adaptive search space to generate a per-instance genetic algorithm for the permutation flow shop problem," *Appl. Soft Comput.*, vol. 124, Jul. 2022, Art. no. 109079.

[14] J. Denzinger, M. Fuchs, and M. Fuchs, "High performance ATP systems by combining several AI methods," in *Proc. 15th Int. Joint Conf. Artif. Intell. (IJCAI)*, 1997, p. 102.

[15] H. Fan, H. Xiong, and M. Goh, "Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints," *Comput. Oper. Res.*, vol. 134, Oct. 2021, Art. no. 105401.

[16] A. Vela, J. M. Cruz-Duarte, J. C. Ortiz-Bayliss, and I. Amaya, "Beyond hyper-heuristics: A squared hyper-heuristic model for solving job shop scheduling problems," *IEEE Access*, vol. 10, pp. 43981–44007,2022.

[17] H.-B. Song, Y.-H. Yang, J. Lin, and J.-X. Ye, "An effective hyper heuristic-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *Appl. Soft Comput.*, vol. 135, Mar. 2023, Art. no. 110022.

[18] N. Bagheri Rad and J. Behnamian, "Recent trends in distributed production network scheduling problem," *Artif. Intell. Rev.*, vol. 55, no. 4, pp. 2945–2995, Apr. 2022.

[19] M. Sánchez, J. M. Cruz-Duarte, J. C. Ortíz-Bayliss, H. Ceballos, H. Terashima-Marin, and I. Amaya, "A systematic review of hyper-heuristics on combinatorial optimization problems," *IEEE Access*, vol. 8, pp. 128068–128095, 2020.

[20] E. O. Abiodun, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, and R. S. Alkhawaldeh, "A systematic review of emerging feature selection optimization methods for optimal text classification: The present state and prospective opportunities," *Neural Comput. Appl.*, vol. 33, no. 22, pp. 15091–15118, Nov. 2021.

[21] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: Revisited," *Int. Ser. Oper. Res. Manage. Sci.*, vol. 272, pp. 453–477, Sep. 2019.

[22] M. Naghibzadeh, *New Generation Computer Algorithms*. London, U.K.: Independent, 2021.

[23] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, U.K.: Pearson, 2003.

[24] L. Li, S. Zijin, N. Jiacheng, and Q. Fei, "Data-based scheduling framework and adaptive dispatching rule of complex manufacturing systems," *Int. J. Adv. Manuf. Technol.*, vol. 66, nos. 9–12, pp. 1891–1905, Jun. 2013.

[25] J. J. van Hoorn, "The current state of bounds on benchmark instances of the job-shop scheduling problem," *J. Scheduling*, vol. 21, no. 1, pp. 127–128, Feb. 2018.

[26] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *Int. J. Prod. Res.*, vol. 20, no. 1, pp. 27–45, Jan. 1982.

[27] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 110–124, Feb. 2016.

[28] B. Akay, D. Karaboga, and R. Akay, "A comprehensive survey on optimizing deep learning models by metaheuristics," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 829–894, Feb. 2022.

[29] A. E. Ezugwu, A. K. Shukla, R. Nath, A. A. Akinyelu, J. O. Agushaka, H. Chiroma, and P. K. Muhuri, "Metaheuristics: A comprehensive overview and classification along with bibliometric analysis," *Artif. Intell. Rev.*, vol. 54, no. 6, pp. 4237–4316, Aug. 2021.

[30] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Hoboken, NJ, USA: Wiley, 2001.

[31] C. A. C. Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 28–36, Feb. 2006.

[32] T. Murata and H. Ishibuchi, "MOGA: Multi-objective genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, vol. 1, Nov. 1995, p. 289.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[34] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," Comput. Eng. Netw. Lab., ETH Zürich, Zürich, Switzerland, TIK Rep. 103, 2001. [Online]. Available: https://www.research-collection.ethz.ch/handle/20.500.11850/145755?show=full

[35] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[36] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," *Eur. J. Oper. Res.*, vol. 285, no. 2, pp. 405–428, Sep. 2020.

[37] K. Sim and E. Hart, "An improved immune inspired hyper-heuristic for combinatorial optimisation problems," in *Proc. Annu. Conf. Genetic Evol. Comput.*, New York, NY, USA, Jul. 2014, pp. 121–128.

[38] S. Nguyen, M. Zhang, and M. Johnston, "A genetic programming based hyper-heuristic approach for combinatorial optimisation," in *Proc. 13th Annu. Conf. Genetic Evol. Comput.*, New York, NY, USA, Jul. 2011, pp. 1299–1306.

[39] N. R. Sabar and G. Kendall, "Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems," *Inf. Sci.*, vol. 314, pp. 225–239, Sep. 2015.

[40] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, Dec. 2013.

[41] E. Levner, J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt, and J. Węglarz, *Handbook on Scheduling: From Theory To Applications*, vol. 12. Cham, Switzerland: Springer, 2009.

[42] N. Babou, D. Rebaine, and M. Boudhar, "Two-machine open shop problem with a single server and set-up time considerations," *Theor. Comput. Sci.*, vol. 867, pp. 13–29, May 2021.

[43] S. A. Kravchenko and F. Werner, "Parallel machine scheduling problems with a single server," *Math. Comput. Model.*, vol. 26, no. 12, pp. 1–11, Dec. 1997.

[44] F. Habibi, F. Barzinpour, and S. J. Sadjadi, "Resource-constrained project scheduling problem: Review of past and recent developments," *J. Project Manage.*, vol. 2018, pp. 55–88, Sep. 2018.

[45] K. R. Escott, H. Ma, and G. Chen, "Transfer learning assisted GPHH for dynamic multi-workflow scheduling in cloud computing," in *Proc. Australas. Joint Conf. Artif. Intell.*, vol. 13151. Cham, Switzerland: Springer, 2022, pp. 440–451.

[46] S. Nahmias, *Production and Operations Analysis*, 7th ed., Long Grove, IL, USA: Waveland Press, 2015.

[47] F. De La Rosa-Rivera, J. I. Nunez-Varela, J. C. Ortiz-Bayliss, and H. Terashima-Marín, "Algorithm selection for solving educational timetabling problems," *Expert Syst. Appl.*, vol. 174, Jul. 2021, Art. no. 114694.

[48] P. Brucker, *Scheduling Algorithms*, 5th ed., Cham, Switzerland: Springer, 2007.

[49] P. Taylor, "Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling," *Manuf. Eng.*, vol. 31, no. 786636650, pp. 37–41, 1993.

[50] Z. Huang, Y. Mei, and M. Zhang, "Investigation of linear genetic programming for dynamic job shop scheduling," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2021, pp. 1–8.

[51] B. Xu, Y. Mei, Y. Wang, Z. Ji, and M. Zhang, "Genetic programming with delayed routing for multiobjective dynamic flexible job shop scheduling," *Evol. Comput.*, vol. 29, no. 1, pp. 75–105, 2020.

[52] S. Daužère-Pérès and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Ann. Oper. Res.*, vol. 70, pp. 281–306, Apr. 1997.

[53] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming," in *Proc. Genetic Evol. Comput. Conf. Companion*, Jul. 2020, pp. 107–108.

[54] W. Shi, X. Song, C. Yu, and J. Sun, "An asynchronous reinforcement learning hyper-heuristic algorithm for flow shop problem," in *Proc. IASTED Int. Conf. Artif. Intell. Appl.*, 2013, pp. 175–180.

[55] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Nav. Res. Logistics Quart.*, vol. 1, no. 1, pp. 61–68, Mar. 1954.

[56] G. M. Komaki, S. Sheikh, and B. Malakooti, "Flow shop scheduling problems with assembly operations: A review and new trends," *Int. J. Prod. Res.*, vol. 57, no. 10, pp. 2926–2955, May 2019.

[57] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.

[58] J. Y.-T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Boca Raton, FL, USA: CRC Press, 2004.

[59] J. Smith, A. Taylor, and H. Lee, "Optimizing patient scheduling in radiology services," *J. Healthcare Manage.*, vol. 63, no. 2, pp. 113–130, 2018.

[60] D. Jones and T. Roberts, "Parallel machine scheduling to maximize production efficiency," *J. Manuf. Syst.*, vol. 54, pp. 65–75, Oct. 2019.

[61] S. Doe and R. Clark, "Advanced scheduling algorithms for data centers," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 832–844, Mar. 2020.

[62] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A survey of university course timetabling problem: Perspectives, trends and opportunities," *IEEE Access*, vol. 9, pp. 106515–106529, 2021.

[63] D. Zhang, Y. Liu, R. M'Hallah, and S. C. H. Leung, "A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems," *Eur. J. Oper. Res.*, vol. 203, no. 3, pp. 550–558, Jun. 2010.

[64] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar, "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113943.

[65] N. Pillay, "A review of hyper-heuristics for educational timetabling," *Ann. Oper. Res.*, vol. 239, no. 1, pp. 3–38, Apr. 2016.

[66] A. Rezaeipanah, S. S. Matoori, and G. Ahmadi, "A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search," *Appl. Intell.*, vol. 51, no. 1, pp. 467–492, Jan. 2021.

[67] A. K. Mandal, M. N. M. Kahar, and G. Kendall, "Addressing examination timetabling problem using a partial exams approach in constructive and improvement," *Computation*, vol. 8, no. 2, p. 46, May 2020.

[68] O. Olufemi and C. Oyeleye, "School timetabling: Solution methodologiesand applications," *IEEE-SEM*, vol. 8, no. 3, pp. 35–62, Mar. 2020.

[69] A. W. Siddiqui and S. A. Raza, "A general ontological timetabling-model driven metaheuristics approach based on elite solutions," *Expert Syst. Appl.*, vol. 170, May 2021, Art. no. 114268.
[70] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A survey of search methodologies and automated system development for examination timetabling," *J. Scheduling*, vol. 13, no. 2, pp. 129–156, 2010.
[71] E. L. Demeulemeester and W. S. Herroelen, *Project Scheduling: A Research Handbook*. New York, NY, USA: Academic, 2002.
[72] O. Icmeli and W. O. Rom, "Solving the resource constrained project scheduling problem with optimization subroutine library," *Comput. Oper. Res.*, vol. 23, no. 8, pp. 801–817, Aug. 1996.
[73] Q.-Z. Xiao, J. Zhong, L. Feng, L. Luo, and J. Lv, "A cooperative coevolution hyper-heuristic framework for workflow scheduling problem," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 150–163, Jan. 2022.
[74] G. Johnson, *Efficiency in Film Production: The Role of Workflow Scheduling*. Bethlehem, PA, USA: Media Studies Press, 2019.
[75] C. Lopez and S. Kim, "Workflow scheduling in manufacturing: Techniques and benefits," *J. Ind. Optim.*, vol. 5, no. 1, pp. 88–102, 2022.
[76] B. K. R. Escott, H. Ma, and G. Chen, *Heuristic Approach for Dynamic Workflow Scheduling in the Cloud*. Cham, Switzerland: Springer, 2020.
[77] Y. Yang, G. Chen, H. Ma, M. Zhang, and V. Huang, "Budget and SLA aware dynamic workflow scheduling in cloud computing with heterogeneous resources," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2021, pp. 2141–2148.
[78] Y. Mei, M. A. Ardeh, and M. Zhang, "Knowledge transfer in genetic programming hyper-heuristics," in *Automated Design of Machine Learning and Search*. Cham, Switzerland: Springer, 2021.

ALONSO VELA was born in Piedras Negras, Coahuila, Mexico, in 1996. He received the B.Sc. degree in biomedical engineering and the M.Sc. degree in computer science from Tecnológico de Monterrey, in 2019 and 2021, respectively.

In 2018, he was with O-I Packaging Solutions as a Systems Engineer, where he was in charge of the quality laboratory. His research interests include machine learning, artificial intelligence, engineering design, and combinatorial optimization problems that have been solved through hyper-heuristics.

GERARDO HUMBERTO VALENCIA-RIVERA was born in Barrancabermeja, Santander, Colombia, in 1991. He received the B.Sc. degree in mechatronics from Universidad Santo Tomás, Bucaramanga, in 2017, and the M.Sc. degree in electrical engineering from Universidad de Guanajuato, Mexico, in 2019. He is currently pursuing the Ph.D. degree in computer science with Tecnológico de Monterrey, Mexico. His research interests include optimal control, microgrids, power quality, metaheuristics, and hyper-heuristics.

JORGE M. CRUZ-DUARTE (Senior Member, IEEE) is currently a Postdoctoral Researcher with Équipe de Recherche BONUS, CNRS, Inria, Centrale Lille, University of Lille, France. From 2021 to 2024, he was a Research Professor with the Research Group on Advanced Artificial Intelligence, Tecnológico de Monterrey. From 2019 to 2021, he was a Postdoctoral Fellow with the Research Group on Intelligent Systems, Tecnológico de Monterrey, Chinese Academy of Sciences. His research interests include neuromorphic computing, automated design and configuration of heuristics, fractional calculus, applied thermodynamics, data science, and artificial intelligence.

He is a Level 1 Member of CONACYT-SNII and an Active Member of IEEE, ACM, and AMEXCOMP. He is the Chair of the IEEE Computational Intelligence Society's Task Force on Automated Algorithm Design, Configuration, and Selection.

JOSÉ CARLOS ORTIZ-BAYLISS (Member, IEEE) was born in Culiacan, Sinaloa, Mexico, in 1981. He received the B.Sc. degree in computer engineering from Universidad Tecnológica de la Mixteca, in 2005, the M.Sc. degree in computer sciences from Tecnológico de Monterrey, in 2008, the Ph.D. degree from Tecnológico de Monterrey, in 2011, the M.Ed. degree from Universidad del Valle de Mexico, in 2017, the B.Sc. degree in project management from Universidad Virtual del Estado de Guanajuato, in 2019, and the M.Ed.A. degree from Instituto de Estudios Universitarios, in 2019.

He is currently an Assistant Research Professor with the School of Engineering and Sciences, Tecnológico de Monterrey. His research interests include computational intelligence, machine learning, heuristics, metaheuristics, and hyper-heuristics for solving combinatorial optimization problems. He is a member of the Mexican National System of Researchers, the Mexican Academy of Computing, and the Association for Computing Machinery.

IVAN AMAYA (Senior Member, IEEE) was born in Bucaramanga, Santander, Colombia, in 1986. He received the B.Sc. degree in mechatronics engineering from Universidad Autónoma de Bucaramanga, in 2008, and the Ph.D. degree in engineering from Universidad Industrial de Santander, in 2015.

From 2016 to 2018, he was a Postdoctoral Fellow with the Research Group with Strategic Focus on Intelligent Systems, Tecnológico de Monterrey. Since 2018, he has been a Research Professor with the School of Engineering and Sciences, Tecnológico de Monterrey. His research interests include numerical optimization of both, continuous and discrete problems, through the application of heuristics, metaheuristics, and hyper-heuristics. He is also interested in finding new ways to improve hyper-heuristic performance. He is a member of the Mexican National System of Researchers, the Mexican Academy of Computing, and the Association for Computing Machinery, and a Senior Member of the Institute of Electrical and Electronics Engineers.