IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 29, 2020, accepted July 11, 2020, date of publication July 14, 2020, date of current version July 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009318

A Systematic Review of Hyper-Heuristics on Combinatorial Optimization Problems

MELISSA SÁNCHEZ, JORGE M. CRUZ-DUARTE, (Member, IEEE), JOSÉ CARLOS ORTÍZ-BAYLISS, (Member, IEEE), HECTOR CEBALLOS, HUGO TERASHIMA-MARÍN, (Senior Member, IEEE), AND IVAN AMAYA, (Member, IEEE)
School of Engineering and Sciences, Tecnológico de Monterrey, Monterrey 64849, Mexico

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

This work was supported in part by the Consejo Nacional de Ciencia y Tecnología (CONACyT) Basic Science Project under Grant 287479, and in part by the ITESM Research Group with Strategic Focus in Intelligent Systems.

ABSTRACT Hyper-heuristics aim at interchanging different solvers while solving a problem. The idea is to determine the best approach for solving a problem at its current state. This way, every time we make a move it gets us closer to a solution. The problem changes; so does its state. As a consequence, for the next move, a different solver may be invoked. Hyper-heuristics have been around for almost 20 years. However, combinatorial optimization problems date from way back. Thus, it is paramount to determine whether the efforts revolving around hyper-heuristic research have been targeted at the problems of the highest interest for the combinatorial optimization community. In this work, we tackle such an endeavor. We begin by determining the most relevant combinatorial optimization problems, and then we analyze them in the context of hyper-heuristics. The idea is to verify whether they remain as relevant when considering exclusively works related to hyper-heuristics. We find that some of the most relevant problem domains have also been popular for hyper-heuristics research. Alas, others have not and few efforts have been directed towards solving them. We identify the following problem domains, which may help in furthering the impact of hyper-heuristics: Shortest Path, Set Cover, Longest Path, and Minimum Spanning Tree. We believe that focusing research on ways for solving them may lead to an increase in the relevance and impact that hyper-heuristics have on combinatorial optimization problems.

INDEX TERMS Combinatorial problems, hyper-heuristics, job-shop scheduling, longest path, NP-hard problems, optimization, set cover, shortest path, systematic review, vehicle routing.

I. INTRODUCTION

Optimization is present in all natural phenomena and living species, including modern human behavior. It is hard to think of a service or product from our daily routine that disregards optimization. For example, vehicles we use for moving around require parts that must be manufactured following a schedule that minimizes costs and improves efficiency. Moreover, such parts must be designed in a way that makes them safe but not cumbersome. Besides, if electronics are involved, a proper cooling scheme must be implemented for the device. And, of course, other examples abound: flight route planning, automatic movie recommendations, and examinations timetabling, just to name a few. Even a simple part of our routine can be subject to optimization. For example, coffee brewing can be optimized to get the amount of water and coffee that makes the most out of each bean, or simply the one that minimizes costs without hindering flavor.

The literature contains examples of how to tackle optimization problems through diverse methodologies; some of them, quite ingenious and others rather rudimentary. Besides, technology advances rather fast, which allows for more robust solutions to the same problems. However, for better or for worse, it also paves the way for new optimization problems. Still, solutions to these new problems can be achieved by proposing new methods or by adapting ones from the literature. Nevertheless, to do so, researchers and practitioners must have access to a global vision of what is currently available.

So, there must be a way to navigate the plethora of paths. Fig. 1 shows a simple organization of problems and solvers, where the colored paths indicate those we target on this work.

The associate editor coordinating the review of this manuscript and approving it for publication was Her-Terng Yau.

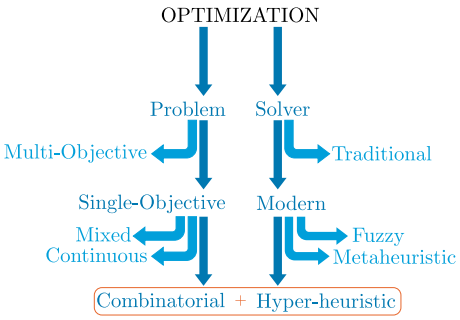


FIGURE 1. A simple tree representation of available optimization problems and solvers. Highlighted roots indicate the elements we target on this work.

On the one hand, an optimization problem deals with finding the best configuration of variables within a delimited feasible region, such that one or more goals are maximized (or minimized). When only a goal is considered, the optimization problem is considered to be single-objective; otherwise, it is multi-objective. These types of problems share similar characteristics, but we do not delve on the multi-objective problem type because it has been recently revised [1], [2]. Therefore, single-objective problems can be split, most simplistically, into two categories concerning the nature of their domains, i.e., continuous and discrete. The former are usually present in practical engineering scenarios, e.g., for finding the channel width that minimizes entropy generation rate in a heat sink for electronic thermal management applications [3], and for assessing the thermophysical properties of a sample from its electric field distribution [4]. Literature is prolific about these problems, and also about those that consider both kinds of variables, e.g., [5]. However, the interaction of this type of problems and hyper-heuristics has not been extensively studied yet [6]. An initial work on this direction is the research by Cruz-Duarte *et al.* [7], which presented a preliminary study to apply hyper-heuristic models to continuous functions. Authors aimed at creating a strategy based on a hyper-heuristic for tailoring population-based metaheuristics and including search operators from well-known techniques as building blocks to generate new ones. Their approach was tested through four benchmark continuous functions with varying number of dimensions. They obtained diverse configurations of metaheuristics that showed promising results. Because of this, we do not deepen on them. Instead, we center our attention on Combinatorial Optimization Problems (COPs). An instance of a COP usually requires finding a valid permutation of values for each variable. Hence, solving a COP may relate to finding a subset of items, such that one parameter is maximized while the other one remains below a given threshold [8]. The current literature describes plenty of COPs that are mainly linked to industrial production, resource management, and vehicle routing, among others. Therefore, the need to delve deeper into these problems is evident, as is their relevance in future research and applications.

We center our study in the relation between Combinatorial Optimization Problems (COPs) and hyper-heuristics. According to literature, this is a more mature field, as we will try to demonstrate it in later sections. Less explored topics regarding hyper-heuristics are, for example, multi-objective optimization [1], [2] and continuous optimization [7]. Examples of the former include the work of Gómez and Terashima-Marín [9], where authors proposed an approach for applying hyper-heuristics to a bi-objective two-dimensional bin packing problem, based on evolutionary-learning mechanism to produce sets of variable-length rules representing hyper-heuristics. The two conflicting objectives considered are the number of bins to accommodate the pieces and the total time required to complete the task. The proposed framework integrates three well-studied multi-objective algorithms such as the NSGA–II [10], the SPEA2 [11], and the GDEA 3 [12]. Maashi *et al.* [13] also reported a learning selection-choice-function-based hyper-heuristic to solve multi-objective optimization problems. Particularly, their work combines three well-known multi-objective optimization algorithms (NSGA–II, SPEA2 and MOGA [14]) using them as a low-level heuristics to solve benchmark multi-objective problems.

On the other hand, there is a wide variety of solvers that deal with the optimization problems described above. Nonetheless, those can be grouped within exact or approximate approaches—which are also known as traditional (hard) or modern (soft) methods [15]–[17]. Hard methods follow a mathematical procedure and provide the best solution for the problem. However, they become too complex or computationally costly for big problems, so they become unfeasible to scale up. Therefore, modern methods seek to solve the problem with a low computing burden. Unfortunately, the latter sacrifice optimality since this approach cannot guarantee it. Heuristics are a representative example of this approach. In general, a heuristic approximates a solution via loosely defined rules or based on trial-and-error. Heuristics are grouped into three categories according to their level of sophistication [6], as described next. Low-level ones (or just heuristics) are the simplest, which can be defined by single rules or direct actions over the search domain. Besides, higher-level methods, known as metaheuristics (MHs), are designed for finding a solution through one or more low-level heuristics that guide the search. Such mechanisms provide some degree of intelligence, even though the search usually contains one or more stochastic components. Traditionally, biological and natural processes have inspired MHs, such as the collective behavior of fish [18] and bees [19], or the annealing of metals [20]. In fact, many authors put them under the umbrella of Simulation Optimization methods because they carry out simulations (with creative analogies) to approach the solution [21]. Metaheuristics are problem-independent and have been employed to solve many hard computational problems (NP-hard) [22]–[26]. Even so, some work may be required for coding the problem into the metaheuristic scheme. For example, Particle Swarm Optimization [27] implements a real-valued codification for each design variable. So, the algorithm must be adapted for dealing

with discrete variables [28]. Nonetheless, MHs have a drawback: they explore the solution space for each problem, so the whole process must be repeated when solving a new problem instance. In other words, no information is preserved across problem instances.

Various approaches that seek to combine low-level heuristics have appeared to surmount such an issue [29]–[31]. One of such approaches is known as Hyper-heuristics (HHs). Their idea is to try and circumvent the No-Free-Lunch (NFL) theorem [32] by learning how to combine heuristics when solving a single instance of a problem. Hence, HHs seek to integrate simple and computationally cheap approaches for conquering a problem [33]. A basic difference between MHs and HHs is that the former explores the solution space of a problem, whereas the latter focuses on the solver space. So, a HH does not solve a problem directly. Instead, at each step of the solution process, it selects a heuristic for dealing with such a step. Even when this has proven fruitful [34]–[40], it leads to the algorithm selection problem [41]. Moreover, HHs have been traditionally classified into selection and generation ones. In the former, selecting the heuristic to apply is done directly. In the latter, selection takes place at a deeper level. In generation HHs, then, each one of the available heuristics is decomposed into building blocks which are then combined for creating new heuristics. Both models have already been implemented in many applications [6].

Despite all the contributions in the literature regarding HHs and COPs, we have not found any investigation that deepens into the relationship between the focus given to hyper-heuristics and the most researched applications on combinatorial optimization. Although there are significant efforts that strive to discuss the newest advancements in hyper-heuristics [42], they fail detail whether those relate to problem domains with the highest interest from the research community. So, we carry this review out to fill this knowledge gap. We do so by analyzing the behavior of the most relevant applications regarding COPs, and whether they remain the same when tackled through HHs.

II. FUNDAMENTALS

This section presents some fundamental concepts about combinatorial optimization problems and hyper-heuristics. Our goal is that the following text serves to guide researchers less familiar with hyper-heuristics into quickly grasping the idea behind this approach.

A. AN OVERVIEW OF SOME COPs

We start by presenting some concepts about a few combinatorial optimization problems. It is important to remark that it is unfeasible to summarize them all here. So, we select some exemplary ones, and mention their main highlights. Even so, the first problem, i.e., Balanced Partition, is discussed in more detail. The reason is that this domain will be used in II-C for showing the way in which hyper-heuristics can be used to solve combinatorial optimization problems. Also, and striving to enhance our scope, we provide references for other COPs so that the interested reader can easily find information about them. This will be shown in Table 2, which will be presented in Sect. IV-A.

1) BALANCED PARTITION (BP)

Imagine a problem where a set of items, S, needs to be split into two subsets, S_1 and S_2, in such a way that the sums of their respective items are as close as possible. This is known as the Balanced Partition (BP) problem. A straightforward application of such a problem is load balancing within a dual-core processor. Of course, this could be extended to the case of more than two subsets. But, for the sake of brevity we will refrain from doing so.

A solution to the BP problem is given by any item distribution across sets. Moreover, an optimal solution to the problem is one with minimum difference across sets. Since item ordering is unimportant, there are 2^n candidate solutions that must be evaluated, as only a few of them may yield the optimal solution. So, if we have an instance given by $S = [1, 1, 2]$, there are eight possible item combinations representing solutions. But, only two of them yield an optimal split, either by setting $S_1 = [1, 1], S_2 = [2]$ or by setting $S_1 = [2], S_2 = [1, 1]$.

One way of solving this kind of problems is to assign all items to a subset, e.g., S_1, and then move items to the other until 50% or more of the load rests on S_2. Such movements can be ruled by a low-level heuristic, e.g., by selecting the largest or the smallest items. These are known as the Max and Min heuristics, and will be used throughout the example given in Sect. II-C. This process, however, has a drawback: a stopping criterion must be included to avoid moving all items to S_2. For BP problems, it is evident that such a criterion relates to the relative load in S_2. In fact, this value can be defined as a 'feature' of the problem, which maps the current state of the instance to a single value, as shown in Eq. 1, where S_i^k represents item k from subset S_i. In doing so, a value of 0.5 and beyond would indicate that no more items should be moved, as the target subset contains 50% or more of the load from the whole system. Bear in mind that additional features can be used to improve the mapping of the problem. Some examples include normalized versions of the average item size at each subset, as well as their standard deviation or the median value. The intuition backing up these ideas rests on assessing the spread within the instance, as a way of mapping solution progress.

$$F_1 = \frac{\sum_{j=1}^{N_2} S_2^j}{\sum_{i=1}^{N_1} S_1^i + \sum_{j=1}^{N_2} S_2^j}. \tag{1}$$

Since there are many candidate solutions for a single instance, a way for comparing among them is required. So, a metric of solution quality, Q, can be defined as the absolute value of the difference among set totals, as shown in Eq. 2. This way, lower values of Q represent better solutions. Also, this metric allows for a straightforward extension to multiple problem instances, as the average Q can be calculated for

assessing the quality of a solver over a set of instances.

$$Q = \left| \sum_{i=1}^{N_1} S_1^i - \sum_{j=1}^{N_2} S_2^j \right| \tag{2}$$

2) CONSTRAINT SATISFACTION

A Constraint Satisfaction Problem (CSP) is defined by a set of variables, X, and a set of constraints, C. Each variable $x \in X$ has a non-empty domain, $D(x)$, of d_x possible values. Each constraint, $c \in C$, specifies the forbidden combinations of values for some subset of variables. The solution of a CSP can either be the solution itself (i.e., the values for each variable so that all constraints are satisfied), or evidence that no solution is possible. Many COPs, such as scheduling, timetabling, rostering, and cutting stock can be formulated as CSPs. Some examples of problems represented as CSPs can be found at [43]–[45]. Nonobe and Ibaraki present some ideas suggesting that if an algorithm for CSPs can solve all those problems, then it could be considered as a ''general solver'', and be used for various problems in real applications [46].

In CSPs, the ordering in which variables are considered for assignment affects the complexity of the search. For this reason, finding methods that efficiently order these variables is paramount and represents an opportunity for improving the search in CSPs. In this regard, hyper-heuristics have been used to control heuristic usage throughout the search [47]. Other recent works include dynamic algorithm portfolios such as CP-Hydra [48], [49] and ACE [50]; as well as hyper-heuristic approaches [51], [52]. There is also a growing interest in automated feature selection for learning purposes [53].

3) BIN PACKING

The Bin Packing Problem (BPP) consists of packing a set of items by minimizing the number of bins used [54], [55]. In general, the BPP is an exciting problem since many other optimization problems such as the cutting stock problem [56] and the knapsack problem [57] can be modeled as BPPs [58]. There is a wide variety of versions for this problem. Some cover aspects such as the number of dimensions of the bins and items. Others, address specific constraints on the type of items that can be packed together or in specific bins.

A mathematical formulation of the BPP [59] is shown in (3). In this formulation, $N = \bigcup_i^n i$ stands for the set of n pieces (or bins), while $y_i \in \{0, 1\}$ is a binary variable indicating whether the bin $i \in N$ has been used, and x_{ij} indicates whether the piece $j \in N$ is packed into bin i.

$$\begin{aligned} &\max \left\{ \sum_{i=1}^{n} y_i \right\} \\ &\text{s.t.} \sum_{j=1}^{n} w_j x_{ij} \leq y_i, \quad \forall\, i \in N \\ &\sum_{i=1}^{n} x_{ij} = 1, \quad \forall\, j \in N \end{aligned} \tag{3}$$

Concerning hyper-heuristics, the literature contains interesting applications for BPP. Burke et al. [60] explored generation hyper-heuristics for the BPP (in up to three dimensions) by using a genetic programming implementation. Their results proved that it is indeed possible to produce packing heuristics with a higher level of generality than that of human-made heuristics. Sim *et al.* explored how immune systems can be used to produce hyper-heuristics for the one-dimensional BPP [61]. Their results were comparable to the ones obtained by state-of-the-art methods and also proved that this type of hyper-heuristic is a reliable strategy for addressing dynamical datasets. Later on, López-Camacho *et al.* [62] proposed a unified hyper-heuristic framework for both, creating and testing hyper-heuristics on one and two-dimensional BPP. In their work, the authors used a genetic algorithm to map the instance state to one suitable heuristic, thus representing a hyper-heuristic.

4) KNAPSACK

The Knapsack Problem (KP), in its classical formulation, consists of a set $S = \bigcup_i^n i$ of n items. Each item i has two properties: profit $p_i \geq 0$ and weight $w_i > 0$. Moreover, there is a knapsack with a limited capacity $c > 0$. It is customary to express the KP as an integer programming problem, where the objective is to identify the selectivity of an item $x_i \in \{0, 1\}$. This implies finding whether an item must be placed into the knapsack ($x_i = 1$) or be left behind ($x_i = 0$). Solving the simplest version of the knapsack problem, when $w_i = p_i \; \forall\, i \in S$, implies finding a subset of items $S_1 = \{i \in S : x_i = 1\}$ such that their total weight remains within the capacity of the knapsack. This problem is described as follows,

$$\max \left\{ \sum_{i=1}^{n} w_i x_i \right\} \quad \text{s.t.} \sum_{i=1}^{n} w_i x_i \leq c. \tag{4}$$

Otherwise, when $w_i \neq p_i \; \forall\, i \in S$, an optimal solution requires that the profit given by S_1 is maximum. In doing so, the problem can be stated as

$$\max \left\{ \sum_{i=1}^{n} p_i x_i \right\} \quad \text{s.t.} \sum_{i=1}^{n} w_i x_i \leq c. \tag{5}$$

The KP is a well-studied COP. The chief reason is that it has a wide range of applications, which include [63]: cargo loading, cutting stock, allocation, and cryptography. Exact methods for solving the KP guarantee finding the optimal solution. However, they require that such a solution exists and that enough run time is given. Dynamic programming and branch and bound are included in this category. Unfortunately, exact methods can only solve small instances because of the exponential growth in the solution space (2^n). Aside from exact methods, literature also describes approximate ones. Among them, we can highlight those commonly known as heuristics. With them, finding the optimal solution is not guaranteed. But it is feasible to find one acceptable one, according to some specific performance metric. One way of

solving the KP with heuristics is to select items (one at a time) until the knapsack is filled. Some common heuristics include selecting the item with the maximum profit, or the one with the minimum weight. More complex heuristics are also feasible, such as the one with the maximum ratio between profit and weight.

Regarding hyper-heuristics, we can mention the work of Drake *et al.* [64]. They implemented a genetic-programming approach for the automatic generation of constructive heuristics for the multi-dimensional version of the KP, obtaining promising results. Also related to generation hyper-heuristics, Burke *et al.* [60] addressed heuristic generation for the KP by using a genetic programming implementation. Their results obtained outstanding results on unseen instances, what supports the idea that their approach produces heuristics that generalize better than their human-designed counterparts. The work by Duhart *et al.* [65] also stands out, where authors solved the KP using hyper-heuristics produced by Ant Colony Optimization.

5) JOB SHOP SCHEDULING

The Job Shop Scheduling Problem (JSSP) is paramount for industries in general, as it offers the ability to craft efficient production programs that can be tailored to customer demands. This problem is detailed as follows. Let $J = \bigcup_{i=1}^{n} j_i$ be a set of n jobs that must be processed on a set of m machines, i.e., $M = \bigcup_{k=1}^{m} m_k$. Likewise, let $a_{i,k} \in A_i$ be an activity (operation) of job j_i that needs to be processed uninterruptedly by machine m_k. Also, each job has to be processed sequentially and may require one or more machines, i.e., $a_{i,k} \rightarrow a_{i,r}\ \forall\ a_{i,k}, a_{i,r} \in A_i$. So, a job j_i comprises a set of at most m activities $A_i = \bigcup_{k=1}^{m} a_{i,k}$, where each activity has a processing time that depends on the machine, i.e., $p_{i,k}$, and a start time $t_{i,k} \geq 0$. The set of all activities is given by $A = \bigcup_{i=1}^{n} A_i$. Thence, a schedule s from the space of all possible schedules S comprises a set of starting times $t_{i,k}$ for all the activities, such that it satisfies all constraints and vouches for the completion of all jobs, i.e., $s = \bigcup_{i=1}^{n} \bigcup_{k=1}^{m} t_{i,k} \in S$. For the most simple conditions, there are $\#S = n!^m$ possible schedules [66]. Subsequently, let $C_s = \bigcup_{i=1}^{n} \bigcup_{k=1}^{m} c_{i,k}$ be a set of completion times for all the activities, since $c_{i,k} = t_{i,k} + p_{i,k}$ with $i, k = \arg\{a_{i,k}\},\ \forall a_{i,k} \in A$. Therefore, the goal in a JSSP can be stated as to find an optimum schedule s_*. For example, one that minimizes the time spent for completing all jobs, given by the makespan $C_{\max} = (\max\{C_s\} : \forall a_{i,j} \in A)$. So, a JSSP can be mathematically formulated as follows [67], [68]:

$$
\begin{aligned}
s_* = \underset{S}{\operatorname{argmin}}\ & \{C_{\max}\} \\
\text{s.t. } & t_{i,r} - t_{i,k} \geq p_{i,k}, \quad \text{if } a_{i,k} \rightarrow a_{i,r},\ \forall a_{i,k}, a_{i,r} \in A, \\
& (t_{i,k} - t_{l,k} \geq p_{l,k}) \vee (t_{l,k} - t_{i,k} \geq p_{i,k}), \\
& \qquad \forall a_{i,k}, a_{l,k} \in A, \\
& t_{i,k} \geq 0, \quad \forall a_{i,k} \in A.
\end{aligned} \tag{6}
$$

Naturally, more elaborated problems can be stated by changing the sources of production demand, type of machines, performance index, characteristics of a production environment, processing characteristic of the operation, kind of plan, and resource constraints [69], [70]. Despite seeming like an easy task, the standard version of the JSSP is considered to be NP-complete [71]. This means there is currently no algorithm that can optimally solve all JSSPs in polynomial time, albeit several solution techniques have been proposed. Some examples include Memetic Algorithm [72], Tabu Search [73], Genetic Programming [74], and Deep Reinforcement Learning [75]. Moreover, should such an algorithm be discovered, it could be used to solve all NP problems, as they can be transformed into the NP-complete domain.

Hyper-heuristics have also been used for tackling JSSPs [76]. For example, Chaurasia *et al.* implemented an Evolutionary Algorithm with guided mutation as HH for solving the JSSP, assuming no waiting time between activities for each job [77]. The authors claimed that this approach surmounted other well-known methodologies in 80% of their experiments. Another example rests on the work of Garza-Santisteban *et al.*, who presented a hyper-heuristic model powered by Simulated Annealing for dealing with JSSP instances of different sizes [39]. They reported that such a model was able to outperform a synthetic Oracle, obtained from several low-level heuristics, in almost 30%. Besides them, Lara-Cárdenas *et al.* utilized Neural Networks for boosting the performance of various HHs reported in the literature in JSSP applications [78].

B. AN OVERVIEW OF HYPER-HEURISTICS

Hyper-heuristics are a fairly recent approach at solving optimization problems. The idea behind a hyper-heuristic is to combine the nature of some available solvers into a more powerful one that outperforms them. Initially, hyper-heuristics (HHs) were classified from two perspectives: their nature and the nature of their available solvers. This process led to the definition of four categories, as perspectives are not mutually exclusive [6]. Based on its nature, a hyper-heuristic can pursue the 'selection' of low-level solvers (LLS), or the 'generation' of new LLS based on the building blocks of the available ones. Now, let us remember that LLS tackle combinatorial optimization problems (COPs). So, they can be used for 'building' the solution one step at a time, or to 'modify' existing ones. Therefore, they are known as constructive and perturbative heuristics, respectively. In this way, the four categories of hyper-heuristics become: selection constructive, selection perturbative, generation constructive, and generation perturbative. It is important to remark that Drake *et al.* recently expanded upon these categories, also providing perspectives related to the feedback, the number of objectives, and the way parameters are set, among others [79]. It is unfeasible to pretend covering all kinds of hyper-heuristics in this manuscript. But, the interested reader is referred to [6], [79].

When properly trained, hyper-heuristics can provide outstanding results. We show an example of how this is achieved in Sect. II-C. So, for now we will mention some relevant works from the literature where hyper-heuristics have been used. Most of them target COPs. For example, Zhong *et al.* used a hyper-heuristic approach for generating schedules of mobile heat sinks in wireless sensor networks [80]. The authors followed a genetic programming (GP) approach with access to five low-level heuristics and the four basic arithmetic operations to link them. They tested their hyper-heuristic approach in stationary and dynamic networks, finding that it was able to outperform human-proposed heuristics and reach values similar to the reference ones, but in shorter times. Another example worth mentioning is that of Toledo *et al.* [81]. The authors focused on the Orienteering Problem with Hotel Selection (OPHS). In this problem, a tour spanning several days must be built, in such a way that points of interests and hotels are selected efficiently. Toledo *et al.* laid out a hyper-heuristic model powered by Large Neighborhood Search. They found results of good quality in acceptable computational times. In fact, they were able to reach the best solution for about 55% of the instances they analyzed.

Notwithstanding, some efforts have been pursued towards extending hyper-heuristics to the continuous domain. For example, Miranda *et al.* developed an approach for fusing automatic algorithm selection and generation hyper-heuristics [82]. Their goal was to provide a hyper-heuristic model with lowered computational cost. To do so, the authors reused previously built algorithms in similar problems. They found that their proposed approach could drastically speedup the process since their selection method only required a fraction of the time it would take to tailor a new solver. Similarly, Cruz-Duarte *et al.* presented a work where they decomposed 10 well-known metaheuristics into 22 low-level search operators [7]. By merging this information, their hyper-heuristic model created 'new' metaheuristics with a different number of search operators (i.e., cardinality). The authors tested their approach under several scenarios with standard test functions. They found that their proposed approach allows finding good sequences of operators, promoting a better exploration and exploitation of the search space. Also, they detected an increased demand for solvers with higher cardinality as problems grew in dimensions.

C. AN EXAMPLE OF HYPER-HEURISTICS IN COPs

We now migrate to discussing a simple, but useful, illustrative example of how a hyper-heuristic works when solving a COP. To do so, we select the Balanced Partition (BP) domain, as the process is rather straightforward. Bear in mind that, despite this, hyper-heuristic models are broad in scope and, so, they can be applied with ease to other COPs. One example of such broadness is found in the different models that can be derived based on the feedback used by the hyper-heuristic, as recently stated by Drake *et al.* [79]. This way, hyper-heuristic models can improve as instances are received (online learning), or they can be trained on a set of instances and then used to solve new problems (offline learning). However, they can also pursue a mixed learning approach, or even have no learning at all. Nonetheless, the following examples are restricted to the case of offline hyper-heuristics, mainly for simplicity. So, online hyper-heuristics are beyond the scope of this section.

Let us assume, for the sake of clarity, that we need to solve the following three problem instances:

$$\left.\begin{array}{lcccccccccccc} \text{Instance 1:} & 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \text{Instance 2:} & 10 & 9 & 8 & 1 & 1 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \\ \text{Instance 3:} & 10 & 3 & 4 & 2 & 10 & 10 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right\} \text{Items} \quad (7)$$

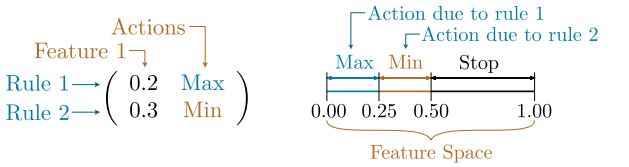
Recall the heuristics mentioned in Sect. II-A1. An instance is solved either by selecting the largest items (Max heuristic) or by selecting the smallest ones (Min heuristic). So, for the first instance both heuristics yield a perfect split, i.e., $Q = 0$, However, the former performs poorly in the second instance, yielding a solution with $Q = 15$. Conversely, the latter provides a high-quality solution ($Q = 1$). For the final instance, none of them are particularly good, though Max performs worse ($Q = 14$ vs. $Q = 6$). Therefore, Max yields an average quality of $Q = 9.67$ across all instances. Hence, Min represents a better alternative, rendering $Q = 2.33$. This means that there is room for improvement in some instances. So, a hyper-heuristic may represent a better approach, and we now focus on them.

1) A RULE-BASED HH MODEL

In this kind of hyper-heuristics, a set of rules guides the decisions of the solver that will be employed. One way to represent such rules is in the form of a selection matrix, where each row represents a different rule. So, the rule most similar to the current problem state is used. Such a similarity metric can be given by the Euclidean distance, with closest elements being more alike. Moreover, the problem state and the rules must be represented somehow. Hence, it is customary to utilize a feature vector that maps different properties of the problem domain. Notice that feature vectors constitute the columns of the selection matrix. For the sake of simplicity, this example considers a single feature (F_1), which was previously defined (see Eq. 1). Additionally, each rule must include an action that will be taken if such a rule is selected. So, a rule is given by $N_f + 1$ elements, where N_f is the number of features considered in the model. This way, in a hyper-heuristic with two rules the selection matrix (i.e., the selector) has two rows (each rule) and two columns (one feature plus the action to take).

Because of the rules-problem interaction, a ''zone of influence'' appears for each selector. Each region provides an overview of the feature values that will lead to each decision. In our case, the plot is given by a straight line, as we only consider a single feature. Different portions of this line indicate different actions to take should the feature take that value.

With this information, we can proceed to the example, which is more detailed version of the one previously commented in [35]. Figure 2 shows a selector (left) and its



(a) Rule-based HH model (b) Zones of influence

FIGURE 2. An example of (a) a rule-based selection hyper-heuristic and (b) its corresponding zone of influence.

corresponding zone of influence (right). Thus, should F_1 be below 0.25, Max will be used for selecting an item. Otherwise, Min will be employed. Bear in mind that this selector was calculated intuitively. But, real applications consider problem instances so complicated that the selector cannot be created in this fashion. So, a learning stage is required, e.g., by using metaheuristics to optimize the performance of the hyper-heuristic. Let us now retrace the path that the hyper-heuristic follows when solving the last instance. Since subset 2 begins empty, the initial feature value is $0/46 = 0$. So, the first item is selected with Max, and the value 10 is moved from subset 1 to subset 2. In the next step, $F_1 = 10/46 = 0.217$. Since $0.217 < 0.250$, Max is used once again. Hence, another item with a value of 10 is moved to subset 2. Now, F_1 becomes 0.435. Since the feature falls within the zone of influence of the Min heuristic, the smallest item (value of 1) is moved to subset 2. Because our selector only contains two rules, and considering that F_1 continually increases, all subsequent decisions will be made with the Min heuristic. In the end, our hyper-heuristic provides a solution where subset 1 contains the items $\{3, 4, 2, 10, 1, 1, 1, 1\}$ while subset 2 keeps $\{10, 10, 1, 1, 1\}$. In other words, it provides a perfectly balanced solution (i.e., $Q = 23 - 23 = 0$). By following a similar approach for solving the other two instances, solutions with $Q = 0$ and $Q = 1$ can be achieved. This yields an average performance metric of 0.33, whilst standalone heuristics yielded worse values (2.33 and 9.67). So, the hyper-heuristic approach provides a solution that is, overall, seven times better than the best available heuristic. But, the fascinating element is that such performance level is achieved without needing to create a new heuristic, nor by exploiting the solution domain. It is achieved by properly mixing available solvers that do not necessarily perform well over the whole set of problems. This is the charm of hyper-heuristics: they combine solvers to conquer a problem.

2) A SEQUENCE-BASED HH MODEL

In the prior case, it was necessary to identify a set of features for mapping the current problem state. So, hyper-heuristic performance is biased to the ability of such features for differentiating among problem states. The most straightforward example is to think of a feature, F_2, given by a constant, e.g., $F_2 = 0.5$. Including such a feature in the previous model may hinder performance, as this feature will map all states to the same value (i.e., 0.5), thus inhibiting any instance differentiation (from this perspective, at least).

Let us then migrate to another hyper-heuristic model, one that does not require features. In this model, we are interested in identifying the right combinations of heuristics instead of general rules. Therefore, this model may perform better under a more restricted set of instances, where they behave similarly. In other words, the idea is to find something akin to rules-of-thumb. This way, the model becomes rather simple, as it only needs to represent the actions that will be taken. Thus, it can be stated as an action vector. Here, each element represents an action that will be taken at a different step in the solution-building process.

Despite its apparent simplicity, this modeling leads to different versions with varying performance. One reason rests in the length of the action vector. In a general case, it is expected that a hyper-heuristic has fewer elements than the number of decisions it will make when solving an instance. Otherwise, it will be akin to tuning the model for a given instance, which may overfit the general behavior of the instance set. So, repetitions must be accounted for somehow. Figure 3 shows three different arrangements for such looping schema. The first one determines the number of times that the sequence will be repeated when solving the instance, and then expands each action in the sequence given by the model, said several times. Sánchez *et al.* followed such an approach [83]. To account for the fact that the sequences will seldom be repeated an integer number of times, the authors began increasing the number of repetitions of each action backward. So, the last actions in the sequence are repeated one more time than the others.

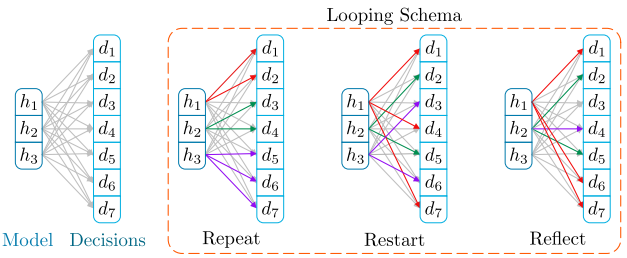


FIGURE 3. Overview of a sequence-based hyper-heuristic model and the effect of three different looping schema on the decisions taken when building a solution, assuming that the hyper-heuristic always requires seven steps to build the solution.

The previous approach requires knowing the number of steps needed for finding the solution beforehand. This assumption holds for some problem domains, such as the one-dimensional Bin Packing Problem, where all items must be packed. But, it does not apply to Balanced Partition, where the number of steps depends on the relative contribution of each selected item. Figure 3 shows other approaches that can be pursued, where each action is only used once (i.e., the ones on the rightmost part). Throughout the first steps of the solution, both strategies behave in the same way. However, after the sequence ends, they diverge. In the first case, the sequence is swept again from the beginning, i.e., it is restarted. In the second one, it is reflected and thus swept decreasingly. Each process is repeated until the instance is solved. In doing

so, different action sequences can be generated, which should yield varying performance, depending on the selected looping style. Nonetheless, both approaches can be used in cases where the number of steps required for finding a solution is unknown.

Let us now see how such approaches behave when solving the troublesome instance from our example, i.e., the third one. Also, assume that the hyper-heuristic will be given by the Min-Max sequence. In other words, the first item will be moved following the Min heuristic. So, an item with the value of one is selected. Then, the biggest item will be moved (Max heuristic). It signifies that an item valued at 10 is selected. Afterward, the selection process will change depending on the considered looping criterion. If the sequence begins anew (i.e., a restart scheme), the Min heuristic will be used, and another item with unitary value will be moved. Then, an item with a value of 10 will be selected (Max heuristic). After another renewal, Min is employed once more, and a solution with a perfect split is achieved. Should the other looping criterion be used (i.e., a reflect scheme), the solution process would be slightly different. Prior to making the third choice, the sequence is reflected, so the last action (i.e., Max) is repeated. So, an item with a value of 10 is selected. Subsequently, Min is used, and an item valued at one is moved. At this point, the sequence is reflected once again, thus repeating the selection with the Min heuristic. Hence, another item valued at one is chosen. Once again, this leads to a perfect split, thus ending the solution process. In summary, in using these sequence-based hyper-heuristics optimal solutions were achieved with a process quite more straightforward than with the rule-based model. Both approaches lead to the same solution, though in a slightly different order. Moreover, when solving the other instances, these models also perform correctly, yielding $Q = 0$ and $Q = 1$ for the first two instances, respectively. Despite this, and as with rule-based hyper-heuristics, the model does not perform properly under some conditions. For example, consider that the model is reversed, i.e., the sequence to use is Max-Min. Once again, both looping methods yield the same solution for the third instance. But, the quality worsens and becomes $Q = 18$, meaning that both heuristics outperformed hyper-heuristics (an undesired scenario). Even so, for the remaining instances, performance is better, yielding Q values for the first two instances of 0 and 3, respectively. Therefore, it is crucial to consider a proper training stage, where appropriate values can be found for the hyper-heuristic model. Nonetheless, it is once again evident that hyper-heuristics can outperform low-level heuristics simply by learning when to use them.

D. SOME HYPER-HEURISTIC FRAMEWORKS

In Section II-C1, we depicted an example of a selection hyper-heuristic through a selection matrix. However, we provided no details on how to produce such a matrix or any other similar hyper-heuristic. Hence, this section briefly describes three hyper-heuristic frameworks from the literature. For the sake of brevity, we only describe their most relevant aspects, as an introductory material. However, for the interested reader, we provide precise citations where such frameworks (or their variations) are detailed.

1) A GENETIC ALGORITHM FRAMEWORK FOR HEURISTIC SELECTION

A popular hyper-heuristic framework is based on genetic algorithms to produce selection hyper-heuristics [33], [84], [85]. From now on, we refer to it simply as GAHH.

In GAHH, a genetic algorithm evolves a population of selection hyper-heuristics. When the evolutionary process finishes, the best individual represents the resulting (best) hyper-heuristic. GAHH is an offline approach. Hence, the hyper-heuristic is ready for deployment, only after the evolutionary process ends. Also, GAHH encodes hyper-heuristics into the chromosomes. So, each chromosome represents a set of rules with the form *condition* $\rightarrow$ *action*. The condition within a rule defines the problem state (given by features that characterize the problem). The corresponding action is the heuristic to apply if such a condition is met. As with the example from Section II-C1, the rule which condition is closest to the current problem state is the one that fires (it determines the heuristic to apply). Closeness is usually calculated through the Euclidean distance.

The genetic algorithm maintains a population of chromosomes, where each chromosome contains two or more rules, as defined in the example from Section II-C1. However, these rules are not represented as a matrix, but as a single vector (the unrolled rules in the matrix). The representation of this chromosome can be either binary or real-valued, but the genetic operators must be updated accordingly. Since the features that characterize the problem domain should be normalized, their values must lie within a well-known interval, given by the minimum and maximum value of each feature considered for the process. Although the genetic operators should guarantee to produce values in such a range, sometimes allowing the values to lie outside the interval has proved beneficial for the hyper-heuristic process [85], [86]. As part of the implementation of the genetic algorithm, the genetic operators should also consider affecting the number of rules in the chromosome, since most of the applications of GAHH allow the chromosome to increase or decrease the rules encoded within them. The evolutionary process iteratively takes place until it meets a termination criterion (usually, the number of iterations). Although there is no restriction, literature shows that a recurring approach uses a steady-state genetic algorithm. Thus, a maximum of two chromosomes are replaced per iteration.

2) A GENETIC PROGRAMMING FRAMEWORK FOR HEURISTIC GENERATION

Hyper-heuristics initially gained popularity mainly because of their capability to switch heuristics throughout the search process. Nowadays, however, they are also used to produce new heuristics based on the 'components' of existing heuristics —the criteria such heuristics use to make their

decisions [64], [87]–[89]. In this regard, the vast majority of works have relied on genetic programming to implement generation hyper-heuristics. From this point on, we will refer to this framework as GPHH.

GPHH represents heuristics as functions, which are evolved by using tree-like structures (customary in genetic programming). Such an evolution is done through genetic operators adapted to tree-like structures. Moreover, functions represent interactions between terminals, and they can choose among a set of available operations. Then, GPHH requires two sets: terminals (that represent the features that characterize the problem state) and operations (that define the interactions between such features). On the one hand, features are usually problem-dependent, since they describe the problem state of a particular domain. On the other hand, operations vary from one implementation to another. The most common ones are the basic arithmetical operations: addition, subtraction, multiplication, and protected division (to avoid errors if the divisor is zero). Other popular choices include minimum and maximum, as well as conditionals such as the *IF-THEN-ELSE* statement. Although less frequently used, some implementations have also included the power and root as available operations. All the functions produced by GPHH must comply with a valid grammar—proposed for each particular implementation. Such a grammar guarantees that the operations receive values they can handle in order to produce a feasible result.

As described before, GPHH evolves functions (in a tree-like structure), and those functions are treated as heuristics. In practice, those heuristics are functions that combine features through the available operations. Then, the result of evaluating one heuristic is a real value, which can be used to determine the next step to take in the search. For example, let us imagine that GPHH is employed to create heuristics for the JSSP (as described in Section II-A5). In such a scenario, a function (that represents a heuristic) will produce a value for each available activity. Then, the activity with the smallest value should be added to the schedule. Please note that the way functions represent heuristics changes from one implementation to the next. Here, we only present one idea within the umbrella of possibilities that have been implemented in literature.

3) HYPER-HEURISTICS FLEXIBLE FRAMEWORK

Literature contains but a few works that attempt to reduce the effort of developing and testing hyper-heuristics, particularly for inexperienced users. The most significant contributions work as software libraries or modules [29], [90]. HyFlex is an example of such frameworks, which stands out due to its impact on the community [29]. It is an object-oriented framework for the implementation and comparison of different hyper-heuristics employing problem-independent operators. HyFlex has been used for both research and competitions—as in the two editions of the Cross-Domain Heuristic Search Challenge (CHeSC), in 2011 and 2013.

Contrary to what we discussed in the previous frameworks, HyFlex works as an iterative method that attempts to improve random initial solutions by using different operators classified as local-search, mutation, crossover, and ruin-and-recreate. Internally, it keeps a list of available heuristics for each of the implemented problem domains. Hence, calling heuristics from different domains becomes transparent for the user. When one problem domain is requested to use one heuristic that is not implemented, HyFlex ignores the call, and the hyper-heuristic moves on to the next decision. From a general perspective, HyFlex works as a perturbative solver. So, features decide among different heuristics without relating to the problem itself. Instead, they relate to the change in the value of their objective function, which is problem-dependent and always produces a real-valued number. Then, HyFlex incorporates both, operators and features, that are problem-independent. This allows the user to create hyper-heuristics that work for any of the supported problem domains with no additional changes. At the moment, HyFlex includes six problem domains: boolean satisfiability, one-dimensional bin packing, permutation flow shop, personnel scheduling, traveling salesman, and vehicle routing.

III. METHODOLOGY

This work aims at determining whether the Combinatorial Optimization Problems (COPs) of highest interest to the research community are the same as those within Hyper-heuristics (HHs) research. In doing so, we intend to provide waypoints for future research works on HHs, which align with relevant combinatorial optimization problems. Therefore, we investigate the literature about two broad research areas: COPs and HHs. Also, we search for sub-areas with the highest impact and those with the most significant research potential. We start by supplying an overview of the leading research topics in combinatorial optimization and gradually focus the search to better understand the kind of publications that impact the academic community. Then, we analyze the role that hyper-heuristics play within these topics. Moreover, we aim at identifying trends and gaps to generate a qualitative analysis of future paths for hyper-heuristic research. Therefore, our methodological approach is organized into the five stages shown in Fig. 4, which we describe below. This method was designed to adopt a reproducible, scientific, and transparent process for anyone who wants to follow it.

Our study begins with a quantitative data collection of previous publications to find relevant combinatorial optimization problems. Afterward, our approach shifts to a deductive data analysis, where we focus on finding meaningful parameters of the collected data. By using them, we filter the list to the ten most relevant problems. Later on, we measure their impact and influence with a citation overview spanning over the past four years. We then repeat the analysis, but restricting search results to hyper-heuristics striving to detect whether the combinatorial problems of higher interest prevail. Finally, we determin the five most relevant authors, journals,

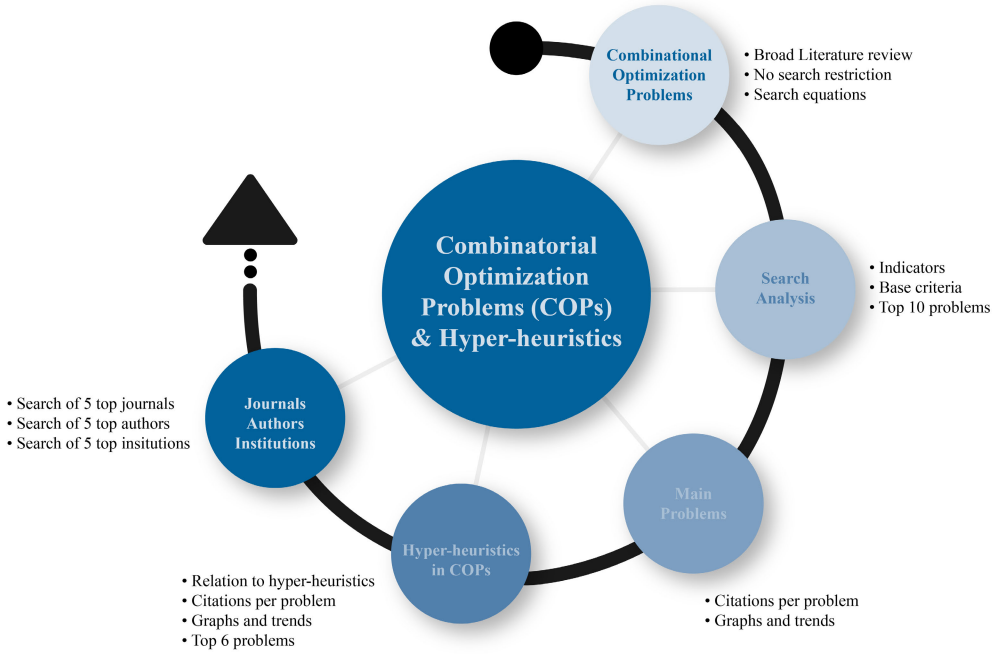


FIGURE 4. Five-stage methodology we follow throughout this work.

and institutions for each one of the top six COPs, within the context of HHs.

A. SEARCH ABOUT COPs

We carry out a broad literature review to identify several Combinatorial Optimization Problems (COPs). We acquire and store data about each problem for further analysis. To this end, we use Elvesier's Scopus database. The reasons are thrice-fold. First, Scopus compounds a comprehensive overview of scientific research from different sources around the world. Second, it allows identifying emerging trends thanks to its built-in analytical tools. Finally, it is accessible due to institutional agreements. The search query, at this point, is designed striving for generality, so time restrictions are disregarded. Hence, we use the expression: ''combinatorial optimization problem''. Afterward, we thoroughly read and analyze both the abstracts and general contents of the results. As an outcome, we find frequently mentioned keywords and expressions that narrow our study. Then, we systematically employ each one of them individually, to expand our data and consolidate the COPs and sub-problems from all the sources. Bear in mind that the same results are returned when using the American and British spelling of the word optimization. So, we disregard using both variants of the word, striving for simplicity.

B. PRELIMINARY DATA ANALYSIS AND SELECTION

This stage aims at performing an in-depth qualitative analysis of the most active fields, as to identify trends for future research. So, we scout each optimization problem and store two indicators that we deem relevant for the analysis: the number of total published documents and the year of its first publication. Using this information, we calculate the average number of manuscripts per year for each domain. Furthermore, we store raw data about the number of documents published for each year andof combinatorial problems to better determine any tendencies.

As the final approach on this stage, we filter our results to the top 10 problem domains. To do so, we keep those that comply with all of the following criteria:

1) The number of published manuscripts tends to increase with each year.
2) The average number of manuscripts per year since the first appearance is above 50.
3) There are over 2000 total published documents.
4) The domain is not inherent and directly related to another one, i.e., the selected problem is not a subdomain of another.

Since this may yield more than ten problem domains, we then sort the remaining ones increasingly, based on elements such as:

1) Average number of documents per year,
2) Total number of published manuscripts, and
3) Growth over the last two years.

Moreover, and to avoid that small differences bias our conclusions, we set the following assumptions for two or more problems:

1) They have the same average of documents per year if their difference is less than or equal to 13;
2) They are equal when they have a difference of less than or equal to 800 manuscripts; and
3) They have the same growth if, and only if, they have precisely the identical growth value.

It is essential to mention that we define each threshold as roughly 10% of the average of its corresponding parameter value. Also, whenever a criterion ties, the next one is analyzed. Even so, there is no case where two or more problem domains exhibit the same growth value.

For example, should Bin Packing Problem (BPP) and Boolean Satisfiability Problem (SAT) exhibit an average of 47 and 50 documents per year, respectively, they would be considered as tied. So, we would compare the total number of published manuscripts. Let us assume that it is 5853 for BPP and 2019 for SAT. Thus, the tie is broken, and BPP becomes more relevant than SAT.

C. MAIN PROBLEMS

This stage strives to analyze citation data to gather insights about research patterns and trends. It identifies recent research interest for each one of the filtered problem domains. Keeping this in mind, we determine the number of citations generated between 2015 and 2019, and related to the manuscripts published between 2014 and 2019. During this process, a new search query is made for each problem, including the word *problem*, to have complete and relevant data. Afterward, we filter the results to those from 2014 to 2019. To better analyze these values, we divide them by the number of available documents, thus considering a ratio (cf. Sect. III-B).

D. HYPER-HEURISTICS IN COPs

This stage focuses on finding whether the results hold when analyzing the output generated by hyper-heuristic research. In other words, we want to see how the most relevant combinatorial optimization problems change when migrating from a global observation to one restricted to what has been tackled with hyper-heuristics. Therefore, we replicate the previous citation overview analysis while restricting the search with the expression: *"hyper-heuristic*" OR "hyper heuristic*"*. The reason: results differ when using each term. Hence, we search for each problem individually and then restrict the corresponding results to the keywords, as mentioned earlier. Afterward, we filter the data to the time window between 2014 and 2019.

Moreover, and once again seeking to avoid bias, we calculate the ratio between the number of citations belonging to works restricted to hyper-heuristics and the corresponding number of unrestricted citations. This way, we can more clearly see the relative impact of each problem.

As a complement, we analyze the number of published manuscripts per year between 2010 and 2019, which relate to hyper-heuristics. So, we use these limits as a new time window. To detect whether there has been a growing interest for each problem, we calculate the number of citations per document, by dividing the citation information into the values yielded by the previous step. Finally, we seek to filter the list of problems to the six most relevant ones. To do so, we include a problem domain if and only if it complies with all the following criteria:

1) With each year, more documents related to hyper-heuristics are cited. To this end, we calculate the average yearly growth of the number of citations for the problem;
2) With respect to its first mention, the problem has accumulated an average of over 50 citations per year, restricted to documents related to hyper-heuristics;
3) Its average number of documents per year, related to hyper-heuristics, is higher than 10; and
4) It includes over 100 citations of documents related to hyper-heuristics.

E. ANALYSIS ABOUT JOURNALS, AUTHORS, AND INSTITUTIONS

As the final approach, we delve deeper into our analysis of the top six problem domains. For each one of these domains, we identify the top five journals, authors and institutions. In this regard, we part from the whole search results restricted to hyper-heuristics, i.e., disregarding time windows. Then, we sort such data per author affiliation, as well as by author name and by source title. In case of ties, we differentiate among them by the number of citations. To support this idea, we use the software VOSviewer to create co-authorship diagrams in terms of authors and organizations for each domain. We look at it as a global authorship network and then simplify it to the top five authors. Based on these indicators, we determine whether the problems of higher interest in combinatorial optimization have also been of high importance in hyper-heuristic research.

Moreover, this allow us to identify the main actors for each one of the top problems. It also enables the analysis to determine how diverse the research community working on hyper-heuristics is for a given problem. Additionally, and to avoid misjudging the relevance of journals, authors, or institutions for each problem, we normalize the resulting data.

IV. RESULTS

A. SEARCH ABOUT COPs

We obtain almost 30000 results when employing the search query: *combinatorial optimization problem*. After inspecting the most relevant results, we detect the common keywords from Fig. 5. As expected, the terms *combinatorial*, *optimization*, and *problem* are present. Nonetheless, related terms, such as *discrete*, are also quite common. Moreover, and since most combinatorial problems of interest are NP-hard, such a term is also present. Besides them, popular terms relate to the kind of solution being sought (i.e., maximization/minimization), as well as to some particular problems (e.g., Traveling Salesman) and solvers (e.g., metaheuristics).

Our expanded search includes each relevant keyword. Table 1 shows the resulting data, where we summarize the number of results per search query, as well as the number of problem domains that can be derived from them. We detect an overlap in the latter. So, we synthesize the data and

FIGURE 5. Most common keywords from the initial search query. The larger the font size, the more occurrences of the term.

TABLE 1. Number of results and types of combinatorial problems from each search query. Some problems overlap across expressions.

Search query	Results	Types
Combinatorial Optimization Problem	29773	26
Discrete Optimization Problem	26575	25
NP-hard Problem	31616	22
Maximization Problem	25614	5
Minimization Problem	58534	12
Heuristics Optimization Problem	46044	18
Metaheuristics Optimization	16065	9
Simulation Optimization Problem	102768	11
Traveling Salesman Problem	10757	11
Knapsack Problem	6040	6
Evolutionary Algorithm Optimization Problem	35070	10

obtain 40 different problems. These can be clustered into 21 different groups, as Table 2 evidences.

B. PRELIMINARY DATA ANALYSIS AND SELECTION

A selection of the problems from Table 2 that comply with the criteria from Sect. III-B leads to 13 candidate problem domains. It means that all of them fulfill the given requirements (i.e., tendency of the number of publications, the average number of manuscripts per year, the total number of published documents, and domain relationship). So, we sort them.

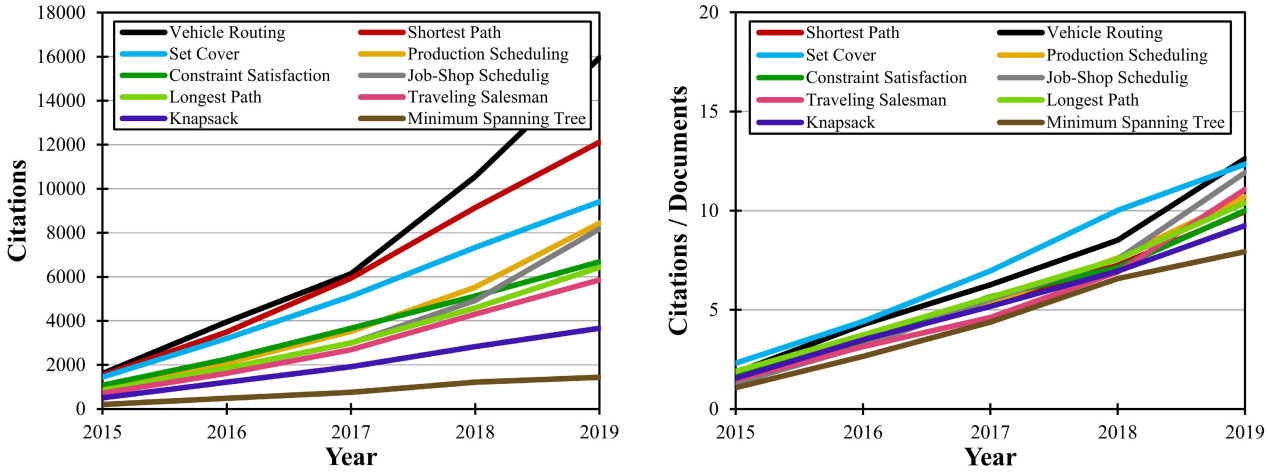
It is interesting to note that only the first two candidates do not tie based on the first criterion. In other words, only they exhibit a difference in average documents per year beyond 13. The leading entry is Vehicle Routing (VR), which has 18 more documents per year (on average) than Constraint Satisfaction (CS). At this point, there is quite a gap in the data, with Traveling Salesman (TS) having 55 fewer documents per year. However, there is a tie between TS, Job-Shop Scheduling (JSS), and Shortest Path (SP). However, since SP has more documents per year, it ends up being third in the ranking. This increases the previously mentioned gap to 59. A similar effect occurs for Production Scheduling, Set Cover (SC), and Knapsack (KP): They are tied, but the former has more documents than the other two, so it ends up ranking higher. It is important to note that such values belong to filtered data, which allows us to better analyze results.

In the case of CS, there is a document that was published in 1959. This entry is discarded since it is not sufficiently related to the subject. So, we consider 1970 as its first publication year. The same happens with the SC and KP problems and their oldest publications. Therefore, we consider the first mention of SC as 1934 and of KP as 1969. Also, bear in mind that the first criterion profoundly affects the number of ties. For example, in our case, 11 problem domains tie based on the first criterion. Nevertheless, if we instead use the second criterion, such a number reduces to five. However, we go for the first one since, this way, we can measure interest from the research community.

The second criterion is the most common tiebreaker, i.e., the number of total document results. Out of the 11 remaining problems, we can set nine of them apart with it. Most cases exhibit a difference between 2000 and 2700 documents. Even so, the highest one is 8898 documents, which occurs between JSS and SP. As a whole, the average difference is 3687 publications.

TABLE 2. Initial list of combinatorial problems, including all queries from Table 1.

Type	Problem
Traveling Salesman [91]	Traveling Salesman [92]
	Multiple Traveling Salesman [93]
	Bottleneck Traveling Salesman [94]
Cutting Stock [95]	Cutting Stock [96]
	2D Cutting [97]
Packing [98]	Packing [99]
	2D Packing [100]
	Bin Packing [101]
Knapsack [102]	Knapsack [103]
	Subset Sum [104]
	Unbounded Knapsack [104]
	Bounded Knapsack [105]
	Multiple Knapsack [106]
	Quadratic Knapsack [107]
Scheduling [108]	Scheduling [109]
	Production Scheduling [110]
	Workforce Scheduling [111]
	Job-Shop Scheduling [112]
	Precedence Constrained Scheduling [113]
Educational Timetabling [114]	Educational Timetabling [115]
Facility Location [116]	Assignment [117]
	Quadratic Assignment [118]
Spanning Tree [119]	Maximum Leaf Spanning Tree [120]
	Degree Constrained Spanning Tree [121]
	Minimum Spanning Tree [122]
Boolean Satisfiability [123]	Boolean Satisfiability [124]
Covering [125]	Minimum Vertex Cover [126]
	Set Cover [127]
	Exact Cover [128]
	Minimum Edge Cover [129]
Vehicle Routing [130]	Vehicle Routing [131]
Constraint Satisfaction [132]	Constraint Satisfaction [133]
Steiner Tree [134]	Steiner Tree [135]
Hamiltonian Path [136]	Hamiltonian Path [137]
	Hamiltonian Cycle
Path [138]	Longest Path [137]
	Shortest Path [139]
Subgraph Isomorphism [140]	Subgraph Isomorphism [141]
Clique [142]	Clique [143]



(a) Total citations (b) Average citations per manuscript

FIGURE 6. Citation overview of top 10 combinatorial optimization problems from a general perspective.

At this point, only two problems need sorting. They are Steiner Tree (ST) and Boolean Satisfiability (BS). So, we use the last criterion, i.e., the growth over the last three years. The average growth of ST (equal to 4.33) is higher than that of BS (equal to 2.33). In any case, these problems are the lowest-ranked ones, so they are ultimately discarded.

Table 3 shows the resultant list of problems, including the number of yearly and total manuscripts, for each problem. We can split the problems depicted in this table into four isolated groups. The top section contains the first two problems, which are quite separated from the others. The second and third ones are close together, and each group contains three problem domains. One has around 172 average results per year; the other, around 129. The lowest section has up to 84 manuscripts per year, and it includes those that do not make it into the top 10. In this regard, Boolean Satisfiability ranks at the bottom, with an average of 50 documents per year. Again, ranking depends on the ordering of sorting criteria. So, the first problem (Vehicle Routing) does not necessarily have the highest number of total manuscripts. It only has 12092 documents. However, other problem domains render higher values. This is the case for Shortest Path and Production Scheduling (PS), which present about 5000 and 1000 more documents, respectively.

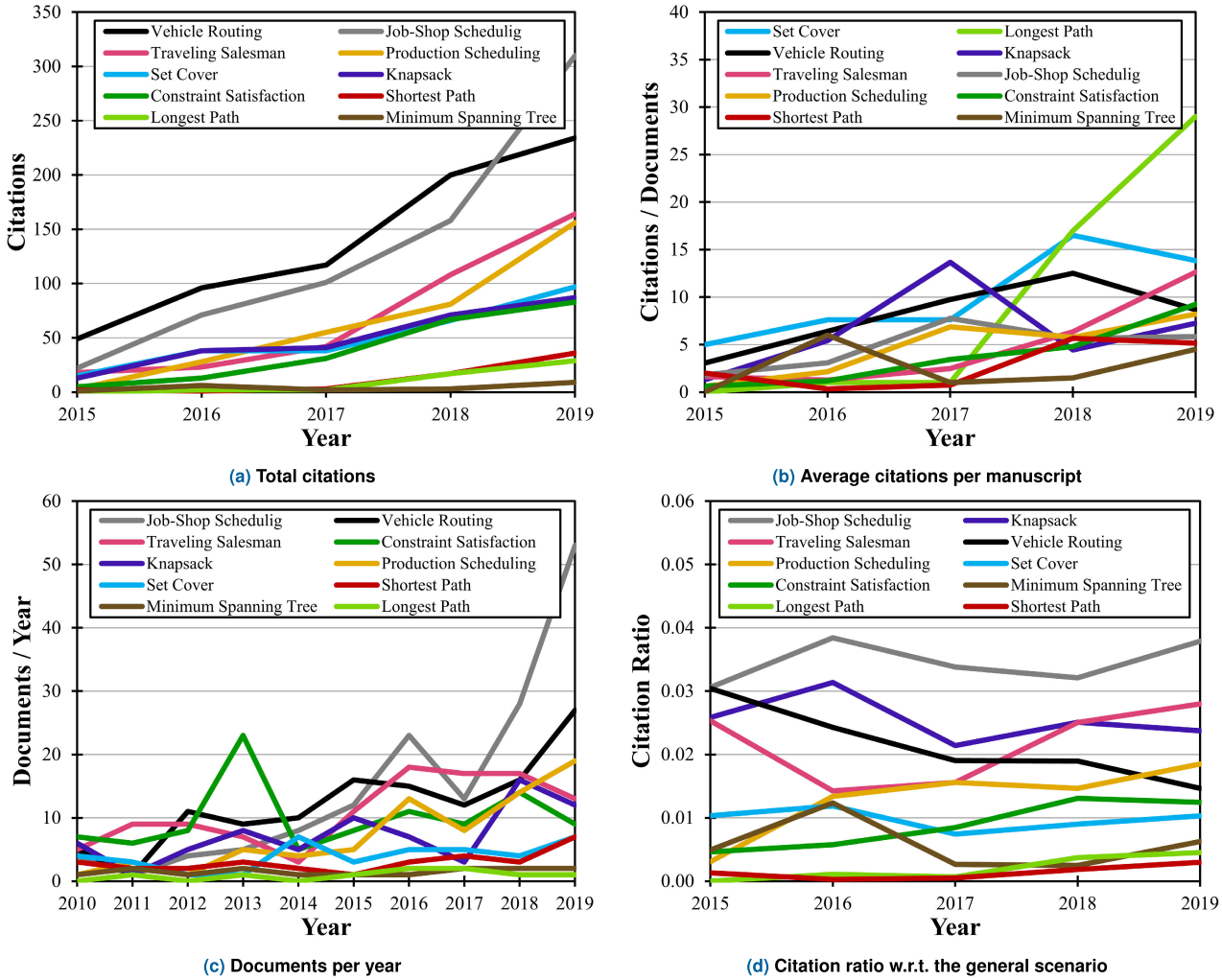
TABLE 3. Top 10 of combinatorial problems when sorting with the criteria from Sect. III-B. Values in bold represent ties between two or more problems. Data shown are given for the general scenario (i.e., without restricting to hyper-heuristics).

Rank	Problem	Results / year	Results
1	Vehicle Routing (VR)	247	12092
2	Constraint Satisfaction (CS)	229	11232
3	Shortest Path (SP)	**170**	17655
4	Traveling Salesman (TS)	**174**	10938
5	Job-Shop Scheduling (JSS)	**172**	8757
6	Production Scheduling (PS)	**133**	13315
7	Set Cover (SC)	**134**	11430
8	Knapsack (KP)	**121**	6026
9	Longest Path (LP)	**84**	8684
10	Minimum Spanning Tree (MST)	**74**	3558

C. MAIN PROBLEMS

As we mention in Sect. III-C, this section considers data of the top 10 problems from Table 3. The total citation count for these problems is 201093, and its evolution is presented in Fig. 6a. Citations have increased across the years and for all problem domains. Most problems follow a mostly linear tendency (R^2 about 0.97), with average growths between 1390 and 1836 citations per year. These data implies that the scientific community is noticing the manuscripts, and also that interest in them is growing. Total citations increased almost nine-fold throughout the time window, shifting from 9626 in 2015 to 78174 in 2019. Even so, it is interesting to remark that all problems seem to increase their number of citations linearly between 2015 and 2017. However, from that point onward, only some of the problem domains exhibit a higher citation growth rate. Besides, Set Cover (SC) started at the same level as VR and SP. Nevertheless, SC grows at a different rate than the others, so it gets separated and thus ends up in the second group discussed in the previous section. Moreover, and even though Minimum Spanning Tree (MST) has exhibited a linear growth rate throughout the years, it ranks last. It was the only one which beganto stagnate towards 2019.

Despite the gap in the number of documents for each problem, it is interesting to see that the average number of citations per article has been similar for all problems (Fig. 6b). In 2015, manuscripts of all problem domains averaged between 1 and 2.4 citations, and most of them grow linearly, with an average increase between 1.75 and 2.6 citations per year and a $R^2 \approx 0.98$. Hence, and even though the gap between citations can be huge (Fig. 6a), all the selected problems increase the number of manuscripts accordingly. Hence, the interest in them has been growing at similar rates. Even so, a noticeable difference among them appears in 2018. For example, MST



(a) Total citations

(b) Average citations per manuscript

(c) Documents per year

(d) Citation ratio w.r.t. the general scenario

FIGURE 7. Citation overview of top 10 combinatorial optimization problems restricted to hyper-heuristics. A value of zero indicates that there are neither documents nor citations for the corresponding year.

begins to slow down, while three other problems stand out: SC, VR, and JSS. The first two exhibit the highest growth rates for most of the observation window. Actually, SC has an advantage over VR of almost two cites per document per year in 2018, but in 2019 VR increases about twice as usual and even surpasses SC. Nonetheless, albeit JSS does not stand out in previous years, in 2019, it increases in more than four citations per document per year, almost reaching the levels of SC and VR. We believe that such behavior reflects a growing interest from the research community as works are being cited more frequently. Should the behavior hold, we expect that VR and JSS become the most relevant combinatorial problems by the end of 2020.

D. HYPER-HEURISTICS IN COPs

We now study the influence of restricting search results to those about hyper-heuristics. Remember that this is achieved by replicating each search and by adding a "limit to" restriction with the expression: *"hyper-heuristic*" OR "hyper heuristic*"*. This leads to a total of 551 publications and 2869 citations, which represents 198224 fewer citations. Hence, manuscripts related to hyper-heuristics only accrue for 1.45% of total citations. Fig. 7a shows the evolution of citation data across the top 10 problems. It is noteworthy that some behaviors hold. For example, total citations also increased nine-fold throughout the time window, rising from 128 in 2015 to 1205 in 2019. Alas, other elements are lost, e.g., tendencies are not as clear, though they still seem linear.

Fig. 7a shows several facts worth highlighting. First, there are three clear groups until 2018, when the middle one splits into two. Once again, the top section includes two problems, one of which matches the global scenario, i.e., VR. The other one is JSS, which previously ranks at a middle position. Subsequently, the next section of the plot contains the two groups with medium interest. It seems as if TS and PS have received more interest in the last two years so that they now have a clear difference w.r.t. SC, KP, and CS. The lowest section includes the last three problems, which never reach more than 40 citations in a single year.

Another interesting fact from Fig. 7a is that the two problems of highest interest in hyper-heuristic research are also the most recent ones (1970 for VR and 1968 for JSS). As an exception, MST is also recent (1971) but represents the problem of the least interest. Consequently, and despite ranking third in the general scenario, the oldest problem (i.e., Shortest Path from 1915) ranks almost at the bottom. Such erratic behavior may be due to hyper-heuristics being recent. Because of this, work with hyper-heuristics has been restricted to some applications. As a consequence, knowledge about their benefits has not been widespread. Hence, being the oldest problem domain does not necessarily imply that it has been explored with hyper-heuristics.

To address this issue, interest from the research community should be generated through pioneering works that illustrate the benefits derived from using them. Hereabouts, researchers working on the same problem domain but through a different approach would eventually notice hyper-heuristics. For example, SP now ranks in the group of lowest interest. Working on hyper-heuristics for dealing with SP problems should allow tapping into its research community. Since this represents the third problem of the highest activity in the global scenario, a meaningful impact could be generated. Nonetheless, the low number of works on SP may be due to some specific reason, e.g., to hyper-heuristics performing poorly. Such a fact would undoubtedly hinder its impact. Therefore, an in-depth study should be developed, which focuses on analyzing the currently available literature and determining the real reason for the low number of manuscripts. However, this goes beyond the scope of our work.

Withal, let us now analyze how citations behave w.r.t. the number of manuscripts. Fig. 7b displays the evolution of the ratio between citations and documents. Summon into mind that we fix the value of zero for cases where there are neither publications nor citations. Opposite to the general case, data do not exhibit a clear tendency, though all problem domains seem to increase. The increase of Longest Path since 2017 is noticeable. Thus, in 2019 LP achieves almost twice the ratio of the next problem, i.e., Set Cover. Even so, VR, SC, and SP lost interest in 2019. This contrasts heavily against the general case, where no problem domain declines over the years. Also, it could mean that some other approach appeared, which has diverted attention from hyper-heuristics. For example, total citations of VR has increased over the years, but its growth for 2019 is smaller thanfor the previous year. A reason for this could rest on the phenomenon mentioned above since the total citations for VR increases more in this period than for the 2017-2018 range (cf. Fig.6a). A similar effect occurs with SC and SP. However, these two do not experience a reduction in their citation growth rate. So, it is unlikely that interest in solving them with hyper-heuristics has diverted to other approaches. Instead, the number of published manuscripts almost doubled from 2018 to 2019, which could mean that interest is growing (see Fig. 7c). Such an effect is also present for VR. Hence, even if interest has diverted due to the appearance of other approaches, research is still being carried out increasingly. Thus, interest in these problems may grow in the upcoming years. Since the rise in documents belongs to 2019, it is also possible that they have not been cited yet. Consequently, we might expect a boost in the number of citations in forthcoming years. In any case, it is evident that interest in these problems is increasing, as more people are working on them.

TABLE 4. Relation of total citations per document in the 2015-2019 range, sorted decreasingly. Cit.: Citations 2015-2019. Doc.: Documents 2015-2019.

Problem domain	Cit.	Doc.	Cit. / Doc.
Set Cover (SC)	254	24	10.60
Vehicle Routing (VR)	696	86	8.09
Longest Path (LP)	50	7	7.14
Production Scheduling (PS)	323	59	5.47
Knapsack (K)	250	48	5.21
Job-Shop Schedulig (JSS)	662	129	5.13
Traveling Salesman (TS)	355	76	4.67
Constraint Satisfaction (CS)	199	51	3.90
Shortest Path (SP)	59	18	3.28
Minimum Spanning Tree (MST)	21	8	2.63

Since the number of papers related to hyper-heuristics is rather small, we run a complementary analysis of the accumulated data. Table 4 shows the total number of documents related to hyper-heuristics generated in the 2015-2019 time window, as well as the citations such documents have generated. We also provide the corresponding citation ratio, and we use such values for sorting the table. Despite some changes in the ordering (w.r.t. Fig. 7b), SC, VR, and LP remain as the three main problems. Similarly, the three problems of lowest interest remain as CS, SP, and MST. However, the middle region of the table exhibits some changes in the ordering. Among them, we can highlight PS rising from the seventh to the fourth place. The reason for this is that PS has exhibited a more stable behavior across the years. Another change includes a swap between positions of JSS and TS.

Fig. 7c shows a broader overview of the number of manuscripts about hyper-heuristics. Overall, the number of documents has increased, going from 36 in 2010 to 150 in 2019, and totaling 814. Notwithstanding, all problem domains have behaved irregularly. For example, even though CS has yielded more records over the years, the peak of over 20 documents achieved in 2013 has notbeen repeated. In turn, it dipped to about five in 2014 and now hovers around the ten documents mark. However, although the order is slightly altered, the top three problems are the same as when analyzed from the perspective of the number of citations (Fig. 7a): Job-Shop Scheduling, Vehicle Routing, and Traveling Salesman. The first two present a dramatic growth since 2017, whereas Traveling Salesman actually began to decline. Similarly, the lowest-ranked problems remain as Minimum Spanning Tree and Longest Path, but the order is altered, contrary to what occurs with the top three. Despite this, they both remain around the two documents per year mark.

TABLE 5. Top six combinatorial optimization problems related to hyper-heuristics. Data shown in the first three columns are average values. Values in bold represent the best result for each criterion.

Rank	Problem	Citation growth	Citations /year	Results /year	Total citations	Total results	Citations /results	Citation ratio (%)
1	Job-Shop Scheduling	**72**	165.5	**23.7**	**694**	**213**	3.3	**3.7**
2	Vehicle Routing	46.3	**174**	17.6	434	158	2.7	1.1
3	Production Scheduling	38.3	80.8	18.4	651	166	3.9	3.2
4	Traveling Salesman	36.5	88.8	14.3	255	129	2.0	1.7
5	Set Cover	20.5	63.5	4.4	254	46	**5.5**	1.0
6	Constraint Satisfaction	19.5	49.8	13.2	263	119	2.2	1.4

Finally, Fig. 7d exhibits the liaison between the number of citations restricted to hyper-heuristics and the global scenario. In other words, it shows the ratio of citations that belong to hyper-heuristics, and so we can observe citation growth. Keep in mind that the figure displays growth as a ratio, which differs from the real number of citations over the periods. In this case, and contrary to the previous ones, most problems do not follow a clear pattern. All problems have experienced increments and decrements throughout the years. However, the only ones that follow a clear tendency are Vehicle Routing (VR), Production Scheduling (PS), and Constraint Satisfaction (CS). The former losses track, whilst the other two gain it. Research of hyper-heuristics for solving VR problems represented 3% of the global citations in 2015, but it fell to 1.5% by 2019. Alternatively, PS and CS increased six-fold and three-fold in the same time period, respectively. Even so, there is a couple of interesting facts regarding JSS problems. The first one concerns the existence of a gap between JSS and the other problem domains, although it only represents 3.8% of all citations from 2019. The second fact is that it seems to exhibit a small rising slope, which means that the percentage could improve in the coming years. In this sense, top problems change w.r.t. Fig. 7a: JSS steps up to the first place, whilst Knapsack rises to the second one, and TS remains third. It is unexpected that Knapsack reaches such a ranking, since until now it had not appeared among the top three domains in any category. Besides, the last problems prevail.

Using the aforementioned data, as well as the metrics from Sec. III-D, we select the six most relevant combinatorial optimization problems for hyper-heuristics, and sort them accordingly, yielding Table 5. A higher average citation growth indicates that the problem has a higher tendency to grow between 2015 and 2019. As previously discussed, the impact of research related to JSS and VR is higher by far than for the remaining problems. JSS seems to be gaining interest quicker. The average citation growth for JSS is about 50% higher than for VR, and it almost doubles that of Production Scheduling, which sits in third place. It is also interesting to notice that JSS ranked first on almost all criteria. The only two exceptions are for the number of citations per year, where VR wins and JSS places second, and for the average number of citations per document, where SC proves best and JSS places third. Thus, it is evident that JSS leads in terms of hyper-heuristic related research.

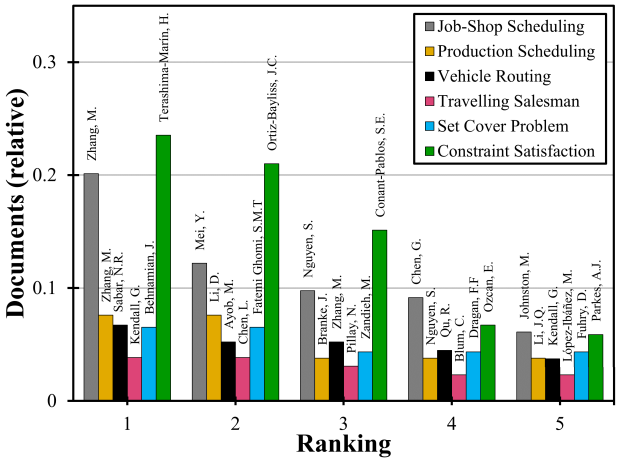


FIGURE 8. Relative contribution per combinatorial optimization problem of the top five authors when restricted to hyper-heuristics.

E. ANALYSIS ABOUT JOURNALS, AUTHORS, AND INSTITUTIONS

Our final approach sought to analyze the most relevant elements of each selected problem domain. We strove to detect the most relevant venues researchers have selected for publishing their work, as well as the leading authors for each problem and their corresponding affiliations. We believe these work as metrics that explain changes in the citation impact and may provide more details about research interest. These insights may also facilitate collaboration among researchers.

Fig. 8 presents the relative contribution of the top five authors per problem. At first glance, we noticed that authors are quite diverse. Nonetheless, it is interesting to see that Zhang, M. is the only author who appears among the top three for half the problem domains. Particularly, Zhang, M. leads the research in JSS and PS problems. In fact, he provided 20% of the documents in JSS. Similarly, CS is the problem domain with the highest concentration, where the first author, Terashima-Marín H. accrues 23% of all works. Still, it represents quite a difference concerning the other problem domains, where the highest-ranking author (also Zhang, M.) accrued less than 8% of HH related works. These problems also seem to have a better distribution of manuscripts, with the top five authors possessing similar contribution ratios. Actually, TS appears to be the most varied problem domain, with a single author contributing a maximum of 2.5% of all

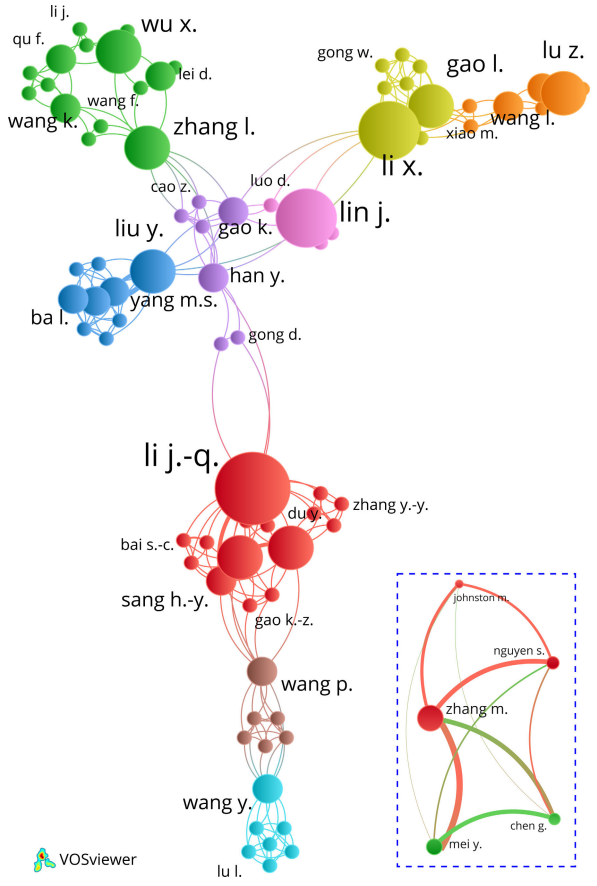


FIGURE 9. Co-authorship network for the Job-Shop Scheduling problem, generated with the VOSviewer software and based on the Scopus data. The box contains the network with the top five authors.

works. It is also worth mentioning that JSS and CS are the problem domains where authors have the highest number of manuscripts. The former leads the ranking, but the latter ranks at the bottom, as Table 5 shows. In both cases the top three authors have a similar number of publications, which is several (three or more) times that of the top three authors of the remaining problems. Moreover, the number of records related to CS is even lower than those for VR, PS, and TS.

Furthermore, we observed that for these two problems (i.e., JSS and CS), research has skewed towards a few authors. Even though this has the benefit of having a strong researcher for that domain, it risks biasing research towards his/her ideas.

When taking a look at author connections for the JSS domain, we found that its 279 authors are spread over 69 clusters. Alas, they are not fully connected. The largest network is composed of 79 researchers distributed throughout nine clusters (see Fig. 9), which represents about 28% of the research community. Such a network does not contain the authors with the highest number of works. So, Fig. 9 shows , in the blue dashed line box, the connections among the top five authors for this problem. Since they are all connected, they all published at least one manuscript with the others. In fact, the top author (Zhang, M.) has published 33 manuscripts

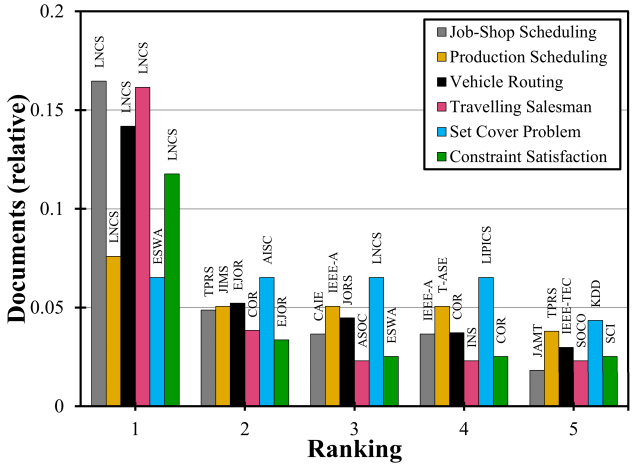


FIGURE 10. Relative contribution per combinatorial optimization problem of the top five journals when restricted to hyper-heuristics.

and virtually all the remaining authors share all of theirs with him. The only exception is Nguyen, S., who shares about 94% representing about 20% of all works for this problem domain. It evidences that a big chunk of the research has been carried out by the same research team. Therefore, the Job Shop Scheduling problem domain could benefit from a higher diversity in the number of research teams that actively contribute towards its development.

A similar issue arises when analyzing the co-authorship network for CS, where the quantity of clusters is 69, covering 231 authors. The largest network is composed of 30 researchers (about 13%) distributed in four clusters. In this case, the top author, Terashima-Marín, H., has published 28 manuscripts. The remaining ones share at least 50% of the publications with him. As mentioned, despite having less manuscripts, the contribution of this research team represents a bigger chunk than for the JSS case. In contrast, Traveling Salesman is the most spread out problem domain. There are 92 clusters and the largest network is conformed by 20 out of 323 authors (about 6%). Besides, only two of the top five researchers share publications between them. A similar effect occurs for Set Cover, where 9 out of 133 authors (about 7%) compose the most extensive network, where only two of the top authors are linked (i.e., Zhang, M. and Nguyen, S.). In addition, there are some mixed cases where only one or two of the top five authors are not connected to the others. It is the case for PS and VR, where their largest networks are composed of 13% (28/220) and 28% (74/296) of the authors, respectively. Since these values are similar to those of JSS and CS, it seems to exist a relationship between the largest network size and the research distribution.

We now focus on the venues researchers have chosen over the years. Fig. 10 shows the top five journals per problem, as well as their normalized number of publications. In this figure, we use the acronyms given in Table 6 for the sake of clarity. There are a few remarks to discuss. First of all, data are not as diverse as the one for authors (cf. Fig. 8). Lecture Notes in Computer Science (LNCS) is the most

TABLE 6. Journal abbreviations (Abbr.) used in this work.

Abb.	Journal
AISC	Advances in Intelligent Systems and Computing
ASOC	Applied Soft Computing Computation
CAIE	Computers and Industrial Engineering
COR	Computers and Operations Research
ECJ	Evolutionary Computation
EJOR	European Journal of Operational Research
ESWA	Expert Systems with Applications
IEEE-A	IEEE Access
INS	Information Sciences
JAMT	International Journal of Advanced Manufacturing Technology
JIMS	Journal of Intelligent Manufacturing
JORS	Journal of the Operational Research Society
KDD	Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
LIPICS	Leibniz International Proceedings in Informatics
LNCS	Lecture Notes in Computer Science Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics
SCI	Studies in Computational Intelligence
SOCO	Soft Computing
TEVC	IEEE Transactions on Evolutionary Computation
TPRS	International Journal of Production Research
T-ASE	IEEE Transactions on Automation Science and Engineering

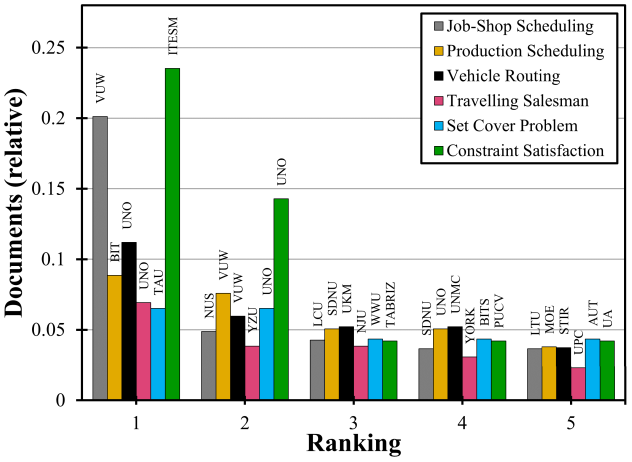


FIGURE 11. Relative contribution per combinatorial optimization problem of the top five institutions when restricted to hyper-heuristics.

common output, ranking first in five out of the six problem domains. Also, for JSS, VR, TS, and CS, the contributions of LNCS are at least thrice the others, ranging between 12% and 17% of all publications made for each problem domain. It differs from the case of authors where only JSS and CS displayed an advantage of such magnitude, although all rankings for PS and SC have a similar contribution. It is noticeable that this time CS does not lead the number of contributions. Moreover, LNCS also appears as a suitable venue for works related to SC, ranking third among all candidates. Albeit the relative contribution of SC is higher than it would seem from the raw data, it remains quite stable, with each of the first four positions contributing about 6% of the manuscripts. Similarly, PS also exhibits an even distribution. This time, however, the top venue (LNCS) constitutes about 7.5% of all research, whilst the next three (JIMS, IEEE-A, and T-ASE) provide approximately 5% each. Besides LNCS, there are other recurring journals worth mentioning, such as Computers and Operations Research (COR), which appears in three domains: second in TS, and fourth in CS and VR. Four other journals appear in two domains: International Journal of Production Research (TPRS), IEEE Access (IEEE-A), Expert Systems with Applications (ESWA), and the European Journal of Operational Research (EJOR). Beyond them, all venues only appear once.

To perform an akin approach, we gathered the top five institutions per problem, taking into account the normalized data and the acronyms from Fig. 11 and Table 6, respectively. It is interesting to remark a recurring institution, University of Nottingham (UNO), which appears in almost all problem domains, i.e., twice as the top institution, twice as the second one, and once as the fourth one. Likewise, Victoria University of Wellington (VUW) is the second most common institution, showing up once at the top and twice at the second place. The problem domain where VUW ranks at the top is JSS, where it contains 33 publications, representing about four times more documents than any other institution, and accruing 20% of all records. Furthermore, considering CS that has five fewer publications in the top institution, Tecnólogico de Monterrey (ITESM) accrues about 23% of all works liaised with it. CS is also the only domain where the top two institutions have several times the number of documents than the rest. For the remaining problems, the behavior is rather even with the first two places representing between 5% and 10%, while the other ones hover between 3% and 5%.

By analyzing author connections at the affiliation level, we identified that authors have tended to mostly collaborate with people from their same institution, as natural. For example, in the case of CS, all top three researchers belong to ITESM. Moreover, the largest cluster contains ten items, where nine of them are affiliated to the same institution (Universidad Autónoma de Chile, UA). Similarly, 41% of all clusters are made up of members from a single institution. In other words, authors have tended to not collaborate with other institutions. In the case of JSS problems, there are 225 different affiliations clustered into 108 different groups. But, the largest one is conformed by 12 items, where five of them are affiliated to the same institution (VUW). As an illustrative example, consider School of Engineering and Computer Science, Evolutionary Computation Research Group, School of Mathematics and Statistics, and Operations Research, which belong to VUW. Similarly, 52% of all clusters are conformed by one organization and 18.5% by two. For the remaining problem domains, the number of clusters is about half the amount of records, and they comprise between 6 and 12 organizations. Although, we noticed that, once again, several of them belong to the same institution. In summary, collaboration across institutions has been scarce–at least for works dealing with hyper-heuristics. We believe that such a fact also may, and will, bias research if there is not a timely collaboration across institutions.

TABLE 7. Institution abbreviations (Abbr.) used in this work.

Abbr.	Institution
AUT	Amirkabir University of Technology
BIT	Beijing Institute of Technology
BITS	Birla Institute of Technology and Science, Pilani
ITESM	Tecnologico de Monterrey
LCU	Liaocheng University
LTU	La Trobe University
MOE	Ministry of Education China
NJU	Nanjing University
NUS	National University of Singapore
PUCV	Pontificia Universidad Católica de Valparaíso
SDNU	Shandong Normal University
STIR	University of Stirling
TABRIZ	University of Tabriz
TAU	Tel Aviv University
UA	Universidad Autónoma de Chile
UKM	University Kebangsaan Malaysia
UNMC	The University of Nottingham Malaysia Campus
UNO	University of Nottingham
UPC	Universitat Politecnica de Catalunya
VUW	Victoria University of Wellington
WWU	Westfalische Wilhelms-Universitat Munster
YORK	University of York
YZU	Yangzhou University

V. DISCUSSION

Throughout this work we set out to answer the following questions:

- **Q1:** What are the Combinatorial Optimization Problems (COPs) of highest interest for the overall research community and how have they evolved since 2015?
- **Q2:** Have these COPs been tackled with hyper-heuristics (HHs)?
- **Q3:** How has interest in these COPs evolved regarding HH research?
- **Q4:** Out of these COPs, which ones have been the most relevant for HH research?
- **Q5:** What have been the most influential authors and institutions on these top problems?
- **Q6:** Where have these researchers chosen to publish their work?

So, this section now discusses such answers. For **Q1**, it becomes evident that, in a general sense, interest in combinatorial optimization problems (COPs) has increased steadily over the years. Such a fact is reflected in a growing number of manuscripts and citations throughout the time window we selected (2015-2019), as Table 3 and Fig. 6 show. For **Q2**, the answer is straightforward: yes, the most relevant COPs have been tackled with hyper-heuristics, though at different scales. Interest has also grown for the case of hyper-heuristics (**Q3**). But its behavior is not as evident. The main reason is the lower number of citations and documents, which can be due to hyper-heuristics being recent. Because of this lowered productivity values, the behavior becomes more sensitive to small changes, such as a reduction of a few articles for a given year, as indicated by Fig. 7.

We analyze the interest of the research community on hyper-heuristics (**Q4**) by observing citations and their corresponding ratio w.r.t. the overall perspective (Table 5). Based on this information, the six most relevant problems are: Job-Shop Scheduling (JSS), Vehicle Routing (VR), Production Scheduling (PS), Traveling Salesman (TS), Set Cover (SC), and Constraint Satisfaction (CS). The citation ratio is stable for some problem domains, e.g. TS and SC. Similarly, other problem domains have been gaining ground by increasing the citation ratio that their manuscripts represent, e.g. PS. Alas, hyper-heuristics have fallen behind in some other problem domains, such as for VR problems. Scientific reports with hyper-heuristics rendered about 3% of all citations related to VR in 2015, although it had diminished to approximately 1.5% by 2019. A remedy rests in improving the attractiveness of hyper-heuristics. In a similar fashion, the number of documents for the Minimum Spanning Tree (MST) problem has been minimal. So, it is a good target for testing new developments and for generating new model ideas. For VR problems, ways must be sought for combining hyper-heuristic models with popular ideas from the overall research community. This way, researchers working on the same problem domain, but through a different approach, would eventually notice hyper-heuristics. In doing so, a novel approach for improving hyper-heuristic performance may be revealed.

Regarding **Q5**, this work also discloses interesting insights about the top authors and their institutions, restricted to hyper-heuristics. For starters, it is clear that some authors work on different problem domains and that they significantly contribute to them (Fig. 8). Moreover, contributions of a single author are quite elevated. Even if this provides a definite referent for a given problem domain, it risks biasing research towards the preferences of a single author. A closer look at co-authorship networks reveals that, in some cases, the top authors of a single problem domain belong to the same research team, as Fig. 9 shows. Nonetheless, there were other problem domains where the opposite happened, e.g. TS problems. This indicates a more mature problem domain, as it has a higher author diversity that provides a broader scope. Regarding institutions (Fig. 11), University of Nottingham (UNO) has a strong presence in hyper-heuristics, as it ranks twice at the top (for VR and TS problems), while also being within the top five for three other problems (SC, CS, and PS). Victoria University of Wellington follows, ranking first for JSS problems and second for PS and VR problems. In the case of JSS and CS problems, the top institutions contribute with 20% or more of the manuscripts. This indicates a strong dominance of one or a few research teams for a given problem domain, which also risks biasing research. By analyzing the co-authorship networks at the institution level, it becomes evident that for most problem domains, the majority of collaborations are bounded by the same institution. For all the domains, the quantity of different clusters is half the quantity of organizations related to the domain. And from this quantity, approximately half of it is conformed by only one document. We believe this further increases the risk of biasing research. For the final question (**Q6**), Fig. 10 reveals that authors commonly choose Lecture Notes in Computer

Science (LNCS) for publishing their work. This venue is the most popular one for all but one problem domain. The only exception is SC, where it ranks third.

In spite of our best efforts, there are undoubtedly some drawbacks to our work. For example, we do not search for all combinatorial problems that have been tackled with hyper-heuristics. Perhaps, this would have allowed identifying some other problem domains where a lot of work has been done with hyper-heuristics.

A quick Scopus search reveals that timetabling problems generated a peak of between 40 and 50 manuscripts in the observation window. Hence, its behavior resembles that of the JSS domain. Nonetheless, in this study, we analyze how much research has been carried out with hyper-heuristics in the most relevant COPs. So, it may be fruitful to expand this analysis in future work. This could be done by reversing the cycle: including problem domains of high interest for the research community working on hyper-heuristics, and determining whether those problems are of high interest for the overall research community. Another drawback of our approach is that the ranking we consider can be affected by the first mention of the problems. If such a problem is really old but interest on it only recently skyrocketed, it will probably be qualified as unimportant. So, our work could be complemented by considering multiple points of view (i.e. rankings). Nonetheless, this goes beyond the scope of this work, though we will consider it for a future one.

Just as a final remark, we want to highlight some insights about the top six COPs. Table 8 shows relevant data for the five most influential manuscripts of each COP. The first element that pops up is that a survey gives the main contribution to the top three problems. These COPs have two or more surveys within their top works. It evidences the high impact that this kind of works can deliver. Nonetheless, there has not been a survey for the remaining three COPs with an impact high enough to make it into the top manuscripts. So, this stands out as an opportunity. Based on the information from JSS, PS, and VR problems, an important focus for the review is to tackle the different models and variations of the main problem, as well as their applications and solvers. For the reader interested in pursuing this path, we recommend reading [144]–[146], as their process may prove valuable for such an endeavor.

A second insight is that most COPs include works over two decades old. Since the metric we use considers the average citations per year, this means that they represent manuscripts that have remained valid and relevant throughout time. The only exception is for PS, where the oldest work dates from 2004 [145]. Since three of the documents within this COP are reviews, this may imply that this field evolves rapidly. Even so, top works still manage to achieve an average of 30-50 citations per year, which matches the ranges of most COPs. Another exciting pattern from Table 8 is that the average citability of TS problems is quite higher than for the other problems. This goes in hand with the fact that most top manuscripts belong to seminal works where methods with high popularity were proposed, such as Ant Colony Systems and Harmony Search. So, all the works date from 2001 or earlier. Such a pattern suggests that TS problems stand as a good choice for testing new solvers and that they can lead to a high impact (an average citability per year of over 180). A final remark from the data is that in the case of CS problems, the dichotomy conjecture has played quite a significant role. If this is combined with the first insight, an exciting research path arises: to deeply review and categorize works dealing with the dichotomy conjecture as to conglomerate its progress and analyze the variants where it has been successfully proven.

Finally, it is important to highlight that selection hyper-heuristics have focused on creating models that select among low-level heuristics (as was shown in the examples). Nonetheless, this does not imply that such models are restricted to low-level heuristics. In fact, any kind of solver could be used here, as long as it allows interactions across different solving methods. So, one could think of a second-order hyper-heuristic where the hyper-heuristic itself is able to select among a pool of low-level heuristics and, let us say, first-order hyper-heuristics. We are currently studying this idea, as we consider that it may provide an additional degree of freedom to the model, which may improve its generalization capabilities. In fact, one may go as far as proposing 'deep' selection hyper-heuristics, where different layers of solvers can be used in conjunction. This way, solvers at the lowest-levels can be highly specialized and generalization could be achieved by progressively abstracting the problem domain with each additional layer. Such a model may also facilitate research about transfer learning in hyper-heuristics, as one topology could be used as a lower-level solver in another. Besides, this permits using other kind of solvers, such as those based on neural-networks. Of course, computational cost is something that may limit this approach, so it may be worthwhile to consider processes that can be paralleled. Nonetheless, in the aforementioned idea training can be done per layer, in such a way that higher layers only use an already trained hyper-heuristic at a lower-layer. Conversely, the whole system could be trained simultaneously, which will skyrocket computational cost but which may also achieve better generalization.

As a summary of this discussion, we encourage researchers and practitioners interested in the joint HH + COP field to:

- Strengthen contributions of HHs in all COPs, but with special attention on JSS, VR, and TS problems.
- Diversify and expand collaboration networks to nourish the field and to avoid biased contributions.
- Develop surveys on all COP domains that focus on problem variants, as well as on the applications and solvers of each domain. This would allow renewing existing surveys and proposing new relevant ones.
- Explore the feasibility of using HHs in relevant COPs that have been left unexplored, such as SP, SC, LP, and MST.

TABLE 8. Summary of the five most influential manuscripts for the top six problems we identified. C: Total number of citations for the manuscript. C/Y: Average citations per year.

	Author(s)	Manuscript title	Year	C	C/Y	Description
Job-Shop Scheduling Problem (JSSP)	Allahverdi et al.	A survey of scheduling problems with setup times or costs [144]	2008	881	73.4	Review covering over 300 papers about scheduling models with setup times (costs). A classification is given according to parameters such as batching, sequence-dependence, and shop environments, among others.
	Taillard E.	Benchmarks for basic scheduling problems [147]	1993	1493	55.3	Proposal of 260 randomly generated scheduling problems which size corresponds to real dimensions of industrial problems.
	Nouiri et al.	An effective and distributed particle swarm optimization algorithm for flexible Job-Shop Scheduling Problem [148]	2018	103	51.5	Solution of the Flexible Job-Shop Scheduling problem (FJSP) with Particle Swarm Optimization (PSO). Results showed that PSO is effective and efficient when solving the FJSP.
	Pezzella et al.	A genetic algorithm for the Flexible Job-Shop Scheduling Problem [149]	2008	524	43.7	A previous study that also focused on the FJSP. But, the authors used a genetic algorithm (GA). They also obtained a good performance when solving this problem.
	Zhang et al.	Review of job shop scheduling research and its new perspectives under Industry 4.0 [69]	2019	39	39.0	Review of over 120 papers that discusses new techniques and frameworks for future developments of JSSP models. Authors state that Industry 4.0 requires that scheduling deals with a smart and distributed manufacturing system, supported by novel and emerging manufacturing technologies.
Production Scheduling Problem (PSP)	Sahinidis N.V.	Optimization under uncertainty: State-of-the-art and opportunities [145]	2004	772	48.3	Review about theory and methodology developed to cope with the complexity of optimization problems under uncertainty. It includes a discussion of several paths for future development.
	Harjunkoski et al.	Scope for industrial applications of production scheduling models and solution methods [150]	2014	278	46.3	Review on existing scheduling methodologies associated to process industries. It also provides an overview of models and methods. Authors include general guidelines and examples about modeling and solving industrial problems.
	Hax et al.	Hierarchical integration of production planning and scheduling [151]	2017	134	44.7	Proposal of a hierarchical planning and scheduling system for a situation of seasonal demand with multiple plants and multiple products. It discusses implementation problems and difficulties for estimating costs and benefits.
	Shrouf et al.	Optimizing the production scheduling of a single machine to minimize total energy consumption costs [152]	2014	252	42.0	Proposal of a mathematical model for minimizing energy consumption costs in a single machine. Authors found that costs can be significantly reduced by avoiding high-energy price periods.
	Branke et al.	Automated design of production scheduling heuristics: a review [153]	2016	135	33.8	Review that suggests a taxonomy of heuristics. It provides guidelines for designing hyper-heuristics for the production scheduling problem. Authors also identify challenges, open questions, and directions for future work.
Vehicle Routing Problem (VRP)	Pillac et al.	A review of Dynamic Vehicle Routing Problems [146]	2013	500	71.4	Survey that classifies routing problems from the perspective of information quality and evolution. Authors also review applications and solution methods for dynamic VRPs.
	Alonso-Mora et al.	On-demand high-capacity ride-sharing via Dynamic Trip-Vehicle Assignment [154]	2017	214	71.3	Generalizes a mathematical model for real-time high-capacity ride-sharing that scales to large numbers of passengers and trips. Authors also dynamically generate optimal routes with respect to online demand and vehicle locations.
	Lin et al.	Survey of Green Vehicle Routing Problem: Past and future trends [155]	2014	416	69.3	Extensive review of Green VRPs (GVRPs). Authors discuss interactions between traditional VRP variants and the GVRP. They also offer an insight into future research regarding GVRPs.
	Solomon Marius M.	Algorithms for the vehicle routing and scheduling problems with time window constraints [156]	1987	2150	65.2	Early work that presents designs and analyzes algorithms for vehicle routing and scheduling problems with time window constraints. Insertion-type heuristics performed particularly well.
	Dorling et al.	Vehicle routing problems for drone delivery [157]	2017	188	62.7	Recent work dealing with two multi-trip VRPs for drone delivery. Authors consider multiple trips to the depot and the effect of battery and payload weight on energy consumption. Results confirm the importance of reusing drones and optimizing battery size.
Traveling Salesman Problem (TSP)	Dorigo et al.	Ant System: optimization by a colony of cooperating agents [158]	1996	7872	328.0	Seminal work about ant systems (AS), where it was used to solve TSPs. The model considers positive feedback, distributed computation, and the use of a constructive greedy heuristic.
	Dorigo et al.	Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem [159]	1997	5536	240.7	Pioneering work on ant colony systems (ACS). As with the previous one, authors applied to TSPs and found that ACS outperforms other nature-inspired algorithms, such as simulated annealing and evolutionary computation.
	Geem et al.	A new heuristic optimization algorithm: Harmony Search [160]	2001	3520	185.3	Seminal work of Harmony Search (HS). Authors claim that it is based on the music improvisation process and test it on TSPs.
	Hopfield and Tank	"Neural" computation of decisions in optimization problems [161]	1985	3740	106.9	Early work dealing with the usage of highly-interconnected networks of nonlinear analog neurons for solving the TSP. Authors claim that effectiveness derives from a nonlinear response of neurons and from their connectivity.
	Stützle and Hoos	MAX-MIN Ant System [162]	2000	2132	106.6	Presentation of a MAX-MIN Ant System to solve the TSP and the quadratic assignment problem (QAP). Results show that, back then, this algorithm provided the best performance for both problems.

TABLE 8. ***(Continued.)*** **Summary of the five most influential manuscripts for the top six problems we identified. C: Total number of citations for the manuscript. C/Y: Average citations per year.**

	Author(s)	Manuscript title	Year	C	C/Y	Description
Set Cover Problem (SCP)	Feige U.	A threshold of $\ln(n)$ for approximating Set Cover [163]	1998	1686	76.6	Early work with the proof that $(1-\mathcal{O}(1))\ln(n)$ is a threshold below which set cover cannot be approximated efficiently, unless NP has slightly superpolynomial time algorithms.
	Hastad J.	Some optimal inapproximability results [164]	2001	801	42.2	Provide NP-hardness of approximation problems by using Probabilistically Checkable Proofs (PCPs). Authors provide improved lower bounds for the efficient approximability of some optimization problems.
	Cardei et al.	Energy–efficient target coverage in wireless sensor networks [165]	2005	780	52.0	Proposal of an efficient method for extending lifetime of sensor networks. Authors model the problem as the maximum set cover problem and design two heuristics for efficiently computing the sets, using linear programming and a greedy approach.
	Impagliazzo et al.	Which problems have strongly exponential complexity? [166]	2001	721	37.9	Manuscript that addresses the relative likelihood of sub–exponential algorithms for NP–complete problems. Authors introduce a generalized reduction called Sub–exponential Reduction Familiy (SERF) that preserves sub-exponential complexity.
	Baker B.S.	Approximation algorithms for NP–complete problems on planar graphs [167]	1994	536	20.6	Description of a general technique for obtaining approximation schemes for various NP–complete problems on planar graphs. The strategy depends on decomposing a planar graph into subgraphs of a form known as k-outerplanar.
Constraint Satisfaction Problem (CSP)	Dechter et al.	Temporal constraint networks [168]	1991	1222	42.1	Extend network-based methods to include continuous variables. Authors provide a framework for processing temporal constraints. They study the applicability of path consistency algorithms as a preprocessing step in temporal problems and prove their termination. Besides, they define boundaries for the complexities.
	Mackworth A.K.	Consistency in networks of relations [169]	1977	1505	35.0	Early work on a general paradigm for tackling the issues that arise when solving CSPs, such as local inconsistencies. It proposes the alternating constraint manipulation with case analyses for producing an OR problem graph that may be searched more easily.
	Bulatov A.A.	A dichotomy theorem for nonuniform CSPs [170]	2017	102	34.0	Confirmation of the Dichotomy Conjecture, proposed by Feder and Vardi in 1993 [171]: given a set of constraints in a CSP, the problem is either solvable in polynomial time or is NP-complete.
	Feder and Vardi	The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory [172]	1998	690	31.4	Early work attempting to find a large subclass of NP-hard problems that exhibits the dichotomy. Authors fail at achieving this, but instead isolated a class known as "Monotone Monadic SNP without inequality" that can be reduced to a CSP.
	Zhuk D.	A proof of CSP dichotomy conjecture [173]	2017	90	30.0	Algorithm proposal for solving CSPs in polynomial time for constraint languages having a weak near unanimity polymorphism, thus contributing to prove the dichotomy conjecture.

- Propose new HH models that fuse ideas from relevant approaches in COPs. One example of such an approach would be to merge HH with metaheuristics. A worthwhile domain for testing the resulting method is TSP, as the most substantial contributions have stemmed from metaheuristics.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we presented an analysis of combinatorial optimization problems (COPs) and their relation with hyper-heuristics (HHs). We focused on identifying problems of the highest interest for the research community, and on whether those are the same when restricted to HHs. We identified 40 COPs. Based on growth data since 2014, we reduced the list to the top 10 problems from a general perspective, and the standpoint of hyper-heuristics. Afterward, we determined the five most relevant authors, journals, and institutions, for the top six problems.

Even though there is some common ground, COPs concerned with the overall research community are not the same as those relevant to the community working on HHs. For example, when considering citation growth, the most prevalent problems in an overall scenario are Vehicle Routing (VR) and Shortest Path (SP). When restricted to hyper-heuristic research, VR still rules, but SP falls to the eighth place. From this perspective, Job-Shop Scheduling (JSS) becomes the second most relevant problem domain, though it ranks sixth in the overall scenario.

Hyper-heuristics are potent tools that can be used to improve the performance of solvers. They have been mainly tested in COPs, though some works on continuous problems have also been explored. Their benefits include the ability to combine different solvers into a more robust approach. Moreover, the idea is flexible enough that it can be tailored to different needs and approaches. One of the most basic approaches is to learn when to select a given solver, from a pool of available ones. This is known as a selection hyper-heuristic and represents a common strategy. Even if this HH seems akin to an algorithm portfolio, the main difference is that a hyper-heuristic chooses at every step of the solution. Instead, an algorithm portfolio uses a single solver from start to end. Because of this, the latter is bounded by the performance of a synthetic Oracle, whilst a hyper-heuristic may even overcome it. Another approach is to decompose

available solvers into building blocks, and then combine such blocks to generate a new solver with improved performance. Even so, their main drawback is that they require a training stage, where the model is refined so that excellent performance can be achieved. Nonetheless, there is work dealing with models that are not tied by this restriction [38]. One of the main achievements of hyper-heuristics in COPs is their broad scope. All the COPs we analyzed had, at least, a few works with hyper-heuristics. It speaks about the diversity inherent to the research community in hyper-heuristics, which nurtures the discussion of ideas. Moreover, such works have been representative, as is evidenced by the average number of citations per document. For the most relevant COP, i.e., SC, each document was cited, on average, 10.6 times, meaning it sparked further developments. Even if the number is not as impressive, in the least relevant COP (Minimum Spanning Tree), each document generated an average of 2.63 citations, which is still valuable.

Hyper-heuristics have gained popularity in some problem domains, but they have lost it in others. For example, the citation ratio (ratio between citations of such works and the overall ones) increased sixfold for Production Scheduling (PS) in the 2015-2019 time window. Alas, it diminished by around 50% for Vehicle Routing (VR). Nonetheless, the number of citations and documents increase throughout these years. So, interest may simply not grow as fast as the allure of the overall problem domain. To address this issue, interest from the research community should be generated through pioneering works that illustrate the benefits derived from using hyper-heuristics. Some ideas include testing their models in the Minimum Spanning Tree (MST) problem and combining them with novel solutions for the VR problem. In the first case, benefits will be two-fold. First, more knowledge about the performance of each model in this COP will be generated. Second, researchers working on MST will notice the benefits of hyper-heuristics. In the second case, better hyper-heuristic models may be derived by incorporating novel ideas from a rapidly evolving COP.

Our data reveal that some authors working on hyper-heuristics have significantly contributed to more than one COP. Moreover, there are some problem domains where the contributions of a single author (or research team) represent a high percentage of all recent works. Constraint Satisfaction is one of such examples, where the leading author has almost 25% of all recent works, and where the top three authors belong to the same research team. Despite providing a bright leading researcher, this phenomenon risks biasing research towards individual ideas. This danger is exacerbated by the small level of collaboration across institutions. Notwithstanding, there is a clear tendency for communicating results through conferences. A proof of this is the notorious dominance of Lecture Notes in Computer Science, which represented an average document contribution of over 10% across the six most relevant COPs.

Efforts should be continued for the following problems: JSS, VR, and TS. The reason: they rank at the top when considering the number of manuscripts, the number of citations, or the citation ratio; thus, they are highly relevant. Moreover, their stats have continued to grow in all three sorting criteria. Nonetheless, allocating resources to these problem domains should further the impact of hyper-heuristics: Shortest Path, Set Cover, Longest Path, and Minimum Spanning Tree. These are optimization problems of interest in an overall scenario, but they have been cast aside when dealing with hyper-heuristics. Particular attention should be given to the first two, as they represent the second and third problems of the highest interest in the overall scenario, respectively.

Information contained within this manuscript paves the road for multiple research paths. For starters, an analysis flowing in the opposite direction should be executed. It would begin by analyzing the COPs that have been most relevant for the overall hyper-heuristics research community, and then explain how they map into the whole scenario. A different path for extending this literature analysis rests on including several rankings for the COPs, so that they are less prone to the effect of long periods of inactivity. Another natural avenue leads to the implementation of HH models in the aforementioned problem domains, and especially in SP and SC as they represent problems of high interest for the overall community. Our analysis could also be expanded to continuous optimization problems and to approaches beyond hyper-heuristics. And, of course, this work should be furthered by keeping a close eye on the behavior of combinatorial optimization problems for the following years, as well as on the evolution of research interest. This way, efforts on hyper-heuristic research could be focused on targeting the most relevant problems. Finally, and even though hyper-heuristics strive at circumventing the No-Free-Lunch theorem, they are still plagued by the algorithm selection problem. Hence, there is currently no model that properly generalizes all scenarios. So, this represents an interesting path to pursue in future works. We are actively working on this by exploring two-phase hyper-heuristic models, where different levels of generalization can be achieved.

REFERENCES

[1] B. Xin, L. Chen, J. Chen, H. Ishibuchi, K. Hirota, and B. Liu, "Interactive multiobjective optimization: A review of the state-of-the-art," *IEEE Access*, vol. 6, pp. 41256–41279, 2018.

[2] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Eng.*, vol. 5, no. 1, Jul. 2018, Art. no. 1502242.

[3] J. M. Cruz-Duarte, A. Garcia-Perez, I. M. Amaya-Contreras, C. R. Correa-Cely, R. J. Romero-Troncoso, and J. G. Avina-Cervantes, "Design of microelectronic cooling systems using a thermodynamic optimization strategy based on cuckoo search," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 7, no. 11, pp. 1804–1812, Nov. 2017.

[4] I. Amaya and R. Correa, "Reconstructing design parameters of a rectangular resonator via peak signal-to-noise ratio and global optimization algorithms," *Inverse Problems Sci. Eng.*, vol. 25, no. 6, pp. 864–886, Jun. 2017.

[5] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury, "Simulation optimization: A review of algorithms and applications," *4OR*, vol. 12, no. 4, pp. 301–333, Dec. 2014.

[6] N. Pillay and R. Qu, *Hyper-Heuristics: Theory and Applications* (Natural Computing Series). Cham, Switzerland: Springer, 2018.

[7] J. M. Cruz-Duarte, A. Ivan, J. C. Ortiz-Bayliss, S. E. Conant-Pablos, and H. Terashima-Marín, "A primary study on hyper-heuristics to customise metaheuristics for continuous optimisation," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.

[8] L. F. Plata-González, I. Amaya, J. C. Ortiz-Bayliss, S. E. Conant-Pablos, H. Terashima-Marín, and C. A. Coello Coello, "Evolutionary-based tailoring of synthetic instances for the knapsack problem," *Soft Comput.*, vol. 23, no. 23, pp. 12711–12728, Dec. 2019.

[9] J. C. Gomez and H. Terashima-Marín, "Evolutionary hyper-heuristics for tackling bi-objective 2D bin packing problems," *Genetic Program. Evolvable Mach.*, vol. 19, nos. 1–2, pp. 151–181, Jun. 2018.

[10] 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.

[11] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. EUROGEN Conf. Evol. Methods Design, Optim. Control Appl. Ind. Problems*, vol. 1, 2001, pp. 95–100.

[12] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, vol. 1, Sep. 2005, pp. 443–450.

[13] M. Maashi and E. Özcan, and G. Kendall, "A multi-objective hyper-heuristic based on choice function," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4475–4493, 2014.

[14] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 1. Hoboken, NJ, USA: Wiley, 2001.

[15] A. Kumar and A. Jaiswal, "Systematic literature review of sentiment analysis on Twitter using soft computing techniques," *Concurrency Comput. Pract. Exper.*, vol. 32, no. 1, p. e5107, Jan. 2020.

[16] D. Pradeepkumar and V. Ravi, "Soft computing hybrids for FOREX rate prediction: A comprehensive review," *Comput. Oper. Res.*, vol. 99, pp. 262–284, Nov. 2018.

[17] P. O. Omolaye, "A holistic review of soft computing techniques," *Appl. Comput. Math.*, vol. 6, no. 2, p. 93, 2017.

[18] K. E. Parsopoulos and M. N. Vrahatis, *Particle Swarm Optimization and Intelligence*. Hershey, PA, USA: IGI Global, 2010.

[19] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, Jun. 2014.

[20] T. Graß and M. Lewenstein, "Hybrid annealing: Coupling a quantum simulator to a classical computer," *Phys. Rev. A, Gen. Phys.*, vol. 95, no. 5, May 2017, Art. no. 052309.

[21] A. Attar, S. Raissi, and K. Khalili-Damghani, "Simulation–optimization approach for a continuous-review, base-stock inventory model with general compound demands, random lead times, and lost sales," *Simulation*, vol. 92, no. 6, pp. 547–564, Jun. 2016.

[22] H. R. Boveiri and M. Elhoseny, "A-COA: An adaptive cuckoo optimization algorithm for continuous and combinatorial optimization," *Neural Comput. Appl.*, vol. 32, no. 3, pp. 681–705, Feb. 2020.

[23] A. A. D. M. Meneses, P. V. da Silva, F. N. Nast, L. M. Araujo, and R. Schirru, "Application of cuckoo search algorithm to loading pattern optimization problems," *Ann. Nucl. Energy*, vol. 139, May 2020, Art. no. 107214.

[24] O. Ramos-Figueroa, M. Quiroz-Castellanos, E. Mezura-Montes, and O. Schütze, "Metaheuristics to solve grouping problems: A review and a case study," *Swarm Evol. Comput.*, vol. 53, Mar. 2020, Art. no. 100643.

[25] R. Xue and Z. Wu, "A survey of application and classification on teaching-learning-based optimization algorithm," *IEEE Access*, vol. 8, pp. 1062–1079, 2020.

[26] H. N. N. Al-Sammarraie and D. N. A. Jawawi, "Multiple black hole inspired meta-heuristic searching optimization for combinatorial testing," *IEEE Access*, vol. 8, pp. 33406–33418, 2020.

[27] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci. (MHS)*, 1995, pp. 39–43.

[28] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Appl. Math. Comput.*, vol. 195, no. 1, pp. 299–308, Jan. 2008.

[29] G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, and E. Burke, "HyFlex: A benchmark framework for cross-domain heuristic search," in *Proc. 12th Eur. Conf. Evol. Comput. Combinat. Optim.*, Berlin, Germany, 2012, pp. 136–147.

[30] C. Ansótegui, J. Gabàs, Y. Malitsky, and M. Sellmann, "MaxSAT by improved instance-specific algorithm configuration," *Artif. Intell.*, vol. 235, pp. 26–39, Jun. 2016.

[31] Y. Malitsky and M. Sellmann, "Instance-specific algorithm configuration as a method for non-model-based portfolio generation," in *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems—CPAIOR* (Lecture Notes in Computer Science), N. Beldiceanu, N. Jussien, and E. Pinson, Eds. Berlin, Germany: Springer, 2012, pp. 244–259.

[32] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[33] J. C. Ortiz-Bayliss, H. Terashima-Marín, and S. E. Conant-Pablos, "Combine and conquer: An evolutionary hyper-heuristic approach for solving constraint satisfaction problems," *Artif. Intell. Rev.*, vol. 46, pp. 327–349, Feb. 2016.

[34] C. Zhang, Y. Zhao, and L. Leng, "A hyper-heuristic algorithm for time-dependent green location routing problem with time windows," *IEEE Access*, vol. 8, pp. 83092–83104, 2020.

[35] I. Amaya, J. C. Ortiz-Bayliss, A. Rosales-Pérez, A. E. Gutiérrez-Rodríguez, S. E. Conant-Pablos, H. Terashima-Marín, and C. A. Coello, "Enhancing selection hyper-heuristics via feature transformations," *IEEE Comput. Intell. Mag.*, vol. 13, no. 2, pp. 30–41, May 2018.

[36] Y. Zhou, J.-J. Yang, and L.-Y. Zheng, "Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling," *IEEE Access*, vol. 7, pp. 68–88, 2019.

[37] J. Lin, "Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 186–196, Jan. 2019.

[38] I. Amaya, J. C. Ortiz-Bayliss, S. Conant-Pablos, and H. Terashima-Marin, "Hyper-heuristics reversed: Learning to combine solvers by evolving instances," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 1790–1797.

[39] F. Garza-Santisteban, R. Sanchez-Pamanes, L. A. Puente-Rodriguez, I. Amaya, J. C. Ortiz-Bayliss, S. Conant-Pablos, and H. Terashima-Marin, "A simulated annealing hyper-heuristic for job shop scheduling problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 57–64.

[40] E. Kieffer, G. Danoy, M. R. Brust, P. Bouvry, and A. Nagih, "Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 44–56, Feb. 2020.

[41] J. R. Rice, "The algorithm selection problem," *Adv. Comput.*, vol. 15, pp. 65–118, Jan. 1976.

[42] 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.

[43] J. A. Berlier and J. M. McCollum, "A constraint satisfaction algorithm for microcontroller selection and pin assignment," in *Proc. IEEE (SoutheastCon)*, Mar. 2010, pp. 348–351.

[44] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," *Eur. J. Oper. Res.*, vol. 119, no. 3, pp. 557–581, 1999.

[45] P. Hell and J. Nešetřil, "Colouring, constraint satisfaction, and complexity," *Comput. Sci. Rev.*, vol. 2, no. 3, pp. 143–163, Dec. 2008.

[46] K. Nonobe and T. Ibaraki, "A tabu search approach to the constraint satisfaction problem as a general problem solver," *Eur. J. Oper. Res.*, vol. 106, nos. 2–3, pp. 599–623, Apr. 1998.

[47] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Metaheuristics*. Norwell, MA, USA: Kluwer, 2003, pp. 457–474.

[48] E. O'Mahony, E. Hebrard, A. Holland, C. Nugent, and B. O'Sullivan, "Using case-based reasoning in an algorithm portfolio for constraint solving," in *Proc. 19th Irish Conf. Artif. Intell. Cogn. Sci.*, 2008, pp. 210–216.

[49] S. Petrovic and R. Qu, "Case-based reasoning as a heuristic selector in a hyper-heuristic for course timetabling problems," in *Proc. 6th Int. Conf. Knowl.-Based Intell. Inf. Eng. Syst. Appl. Technol. (KES)*, vol. 82, pp. 336–340, Sep. 2002.

[50] S. L. Epstein, E. C. Freuder, R. Wallace, A. Morozov, and B. Samuels, "The adaptive constraint engine," in *Proc. 8th Int. Conf. Princ. Pract. Constraint Program. (CP)*. London, U.K.: Springer-Verlag, 2002, pp. 525–542.

[51] B. Crawford, R. Soto, C. Castro, and E. Monfroy, "A hyperheuristic approach for dynamic enumeration strategy selection in constraint satisfaction," in *Proc. 4th Int. Conf. Interplay Between Natural Artif. Comput. New Challenges Bioinspired Appl. (IWINAC)*. Berlin, Germany: Springer-Verlag, 2011, pp. 295–304.
[52] R. Soto, B. Crawford, E. Monfroy, and V. Bustos, "Using autonomous search for generating good enumeration strategy blends in constraint programming," in *Proc. ICCSA*, 2012, pp. 607–617.
[53] R. Diao and Q. Shen, "Nature inspired feature selection meta-heuristics," *Artif. Intell. Rev.*, vol. 44, no. 3, pp. 311–340, Oct. 2015.
[54] J. H. Drake, J. Swan, G. Neumann, and E. Özcan, "Sparse, continuous policy representations for uniform online bin packing via regression of interpolants," in *Evolutionary Computation in Combinatorial Optimization*. (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2017, pp. 189–200.
[55] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu, "Solving a new 3D bin packing problem with deep reinforcement learning method," 2017, *arXiv:1708.05930*. [Online]. Available: http://arxiv.org/abs/1708.05930
[56] M. Delorme, M. Iori, and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *Eur. J. Oper. Res.*, vol. 255, no. 1, pp. 1–20, Nov. 2016.
[57] U. Eliiyi and D. T. Eliiyi, "Applications of bin packing models through the supply chain," *Int. J. Bus. Manage.*, vol. 1, pp. 11–19, Jun. 2009.
[58] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter, "MIPLIB 2010," *Math. Program. Comput.*, vol. 3, no. 2, pp. 103–163, Jun. 2011.
[59] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Hoboken, NJ, USA: Wiley, 1990.
[60] E. K. Burke, M. R. Hyde, G. Kendall, and J. Woodward, "Automating the packing heuristic design process with genetic programming," *Evol. Comput.*, vol. 20, pp. 63–89, Mar. 2012.
[61] K. Sim, E. Hart, and B. Paechter, "Learning to solve bin packing problems with an immune inspired hyper-heuristic," in *Proc. ECAL*, 2013.
[62] E. López-Camacho, H. Terashima-Marin, P. Ross, and G. Ochoa, "A unified hyper-heuristic framework for solving bin packing problems," *Expert Syst. Appl.*, vol. 41, no. 15, pp. 6876–6889, Nov. 2014.
[63] C. Wilbaut, S. Hanafi, and S. Salhi, "A survey of effective heuristics and their application to a variety of knapsack problems," *IMA J. Manage. Math.*, vol. 19, no. 3, p. 227, 2008.
[64] J. H. Drake, M. Hyde, K. Ibrahim, and E. Özcan, "A genetic programming hyper-heuristic for the multidimensional knapsack problem," *Kybernetes*, vol. 43, no. 9/10, pp. 1500–1511, Nov. 2014.
[65] B. Duhart, F. Camarena, J. C. Ortiz-Bayliss, I. Amaya, and H. Terashima-Marín, "An experimental study on ant colony optimization hyper-heuristics for solving the knapsack problem," in *Pattern Recognition*, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, J. A. Olvera-López, and S. Sarkar, Eds. Cham, Switzerland: Springer, 2018, pp. 62–71.
[66] A. Muluk, H. Akpolat, and J. Xu, "Scheduling problems—An overview," *J. Syst. Sci. Syst. Eng.*, vol. 12, pp. 481–492, Dec. 2003.
[67] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *Eur. J. Oper. Res.*, vol. 113, no. 2, pp. 390–434, Mar. 1999.
[68] P. J. Van Laarhoven, E. H. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Oper. Res.*, vol. 40, no. 1, pp. 113–125, 1992.
[69] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under industry 4.0," *J. Intell. Manuf.*, vol. 30, no. 4, pp. 1809–1830, Apr. 2019.
[70] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 904–916, Jul. 2019.
[71] J. Mohan, K. Lanka, and A. N. Rao, "A review of dynamic job shop scheduling techniques," *Procedia Manuf.*, vol. 30, pp. 34–39, 2019.
[72] G. Gong, Q. Deng, R. Chiong, X. Gong, and H. Huang, "An effective memetic algorithm for multi-objective job-shop scheduling," *Knowl.-Based Syst.*, vol. 182, Oct. 2019, Art. no. 104840.
[73] P. Fattahi, M. M. Bidgoli, and P. Samouei, "An improved tabu search algorithm for job shop scheduling problem trough hybrid solution representations," *J. Qual. Eng. Prod. Optim.*, vol. 3, no. 1, pp. 13–26, 2018.
[74] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Genetic programming for job shop scheduling," in *Evolutionary and Swarm Intelligence Algorithms*. Cham, Switzerland: Springer, 2019, pp. 143–167.
[75] D. Coelho, "Deep reinforcement learning as a job shop scheduling solver: A literature," in *Proc. Hybrid Intell. Syst. 18th Int. Conf. Hybrid Intell. Syst. (HIS)*, vol. 923. Porto, Portugal: Springer, Dec. 2018, p. 350.
[76] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: Revisited," in *Handbook of Metaheuristics*. Cham, Switzerland: Springer, 2019, pp. 453–477.
[77] S. N. Chaurasia, S. Sundar, D. Jung, H. M. Lee, and J. H. Kim, "An evolutionary algorithm based hyper-heuristic for the job-shop scheduling problem with no-wait constraint," in *Harmony Search and Nature Inspired Optimization Algorithms*, vol. 741. Singapore: Springer, 2019, pp. 249–257.
[78] E. Lara-Cárdenas, X. Sánchez-Díaz, I. Amaya, and J. C. Ortiz-Bayliss, "Improving hyper-heuristic performance for job shop scheduling problems using neural networks," in *Advances in Soft Computing* L. Martínez-Villaseñor, I. Batyrshin, and A. Marín-Hernández, Eds. Cham, Switzerland: Springer, 2019, pp. 150–161.
[79] 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.
[80] J. Zhong, Z. Huang, L. Feng, W. Du, and Y. Li, "A hyper-heuristic framework for lifetime maximization in wireless sensor networks with a mobile sink," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 1, pp. 223–236, Jan. 2020.
[81] A. Toledo, M.-C. Riff, and B. Neveu, "A hyper-heuristic for the orienteering problem with hotel selection," *IEEE Access*, vol. 8, pp. 1303–1313, 2020.
[82] P. B. C. Miranda, R. B. C. Prudêncio, and G. L. Pappa, "H3AD: A hybrid hyper-heuristic for algorithm design," *Inf. Sci.*, vol. 414, pp. 340–354, Nov. 2017.
[83] X. Sánchez-Díaz, J. C. Ortiz-Bayliss, I. Amaya, J. M. Cruz-Duarte, S. E. Conant-Pablos, and H. Terashima-Marin, "A preliminary study on feature-independent hyper-heuristics for the 0/1 knapsack problem," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.
[84] H. Terashima-Marín, J. C. Ortiz-Bayliss, P. Ross, and M. Valenzuela-Rendón, "Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problems," in *Proc. 10th Annu. Conf. Genetic Evol. Comput. (GECCO)*, 2008, pp. 571–578.
[85] P. Ross, J. G. Marín-Blázquez, S. Schulenburg, and E. Hart, "Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, 2003, pp. 1295–1306.
[86] H. Terashima-Marín, P. Ross, C. J. Farías-Zárate, E. López-Camacho, and M. Valenzuela-Rendón, "Generalized hyper-heuristics for solving 2D regular and irregular packing problems," *Ann. Oper. Res.*, vol. 179, no. 1, pp. 369–392, Sep. 2010.
[87] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, *Exploring Hyper-heuristic Methodologies With Genetic Programming*. Berlin, Germany: Springer, 2009, pp. 177–201.
[88] R. Hunt, K. Neshatian, and M. Zhang, "A genetic programming approach to hyper-heuristic feature selection," in *Simulated Evolution and Learning*, L. T. Bui, Y. S. Ong, N. X. Hoai, H. Ishibuchi, and P. N. Suganthan, Eds. Berlin, Germany: Springer, 2012, pp. 320–330.
[89] E. Sopov, "Genetic programming hyper-heuristic for the automated synthesis of selection operators in genetic algorithms," in *Proc. 9th Int. Joint Conf. Comput. Intell. (IJCCI)*, C. Sabourin, J. J. M. Guervos, U.-M. O'Reilly, K. Madani, and K. Warwick, Eds. Madeira, Portugal: SciTePress, Nov. 2017, pp. 231–238.
[90] J. Swan, E. Özcan, and G. Kendall, "Hyperion—A recursive hyper-heuristic framework," in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed. Berlin, Germany: Springer, 2011, pp. 616–630.
[91] N. Sathya and A. Muthukumaravel, "A review of the optimization algorithms on traveling salesman problem," *Indian J. Sci. Technol.*, vol. 8, no. 29, pp. 1–4, Nov. 2015.
[92] P. Zhao and D. Xu, "Hybrid algorithm for solving traveling salesman problem," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 646, no. 1, pp. 012032-1–012032-6, 2019.
[93] Y. Shuai, S. Yunfeng, and Z. Kai, "An effective method for solving multiple travelling salesman problem based on NSGA-II," *Syst. Sci. Control Eng.*, vol. 7, no. 2, pp. 121–129, 2019.
[94] M. Patterson and D. Friesen, "Variants of the traveling salesman problem," *Stud. Bus. Econ.*, vol. 14, no. 1, pp. 208–220, 2019.

[95] K. B. Parmar, H. B. Prajapati, and V. K. Dabhi, "Cutting stock problem: A survey of evolutionary computing based solution," in *Proc. IEEE Int. Conf. Green Comput., Commun. Electr. Eng. (ICGCCEE)*, Dec. 2014, pp. 1–6.

[96] K. C. Ágoston, "The effect of welding on the one-dimensional cutting-stock problem: The case of fixed firefighting systems in the construction industry," *Adv. Oper. Res.*, vol. 2019, pp. 1–12, Mar. 2019.

[97] P. B. Bangun, S. Octarina, and A. P. Pertama, "Implementation of branch and cut method on n-sheet model in solving two dimensional cutting stock problem," *J. Phys. Conf. Ser.*, vol. 1282, Jul. 2019, Art. no. 012012.

[98] N. Chernov, Y. Stoyan, and T. Romanova, "Mathematical model and efficient algorithms for object packing problem," *Comput. Geometry*, vol. 43, no. 5, pp. 535–553, Jul. 2010.

[99] T. Romanova, A. Pankratov, I. Litvinchev, Y. Pankratova, and I. Urniaieva, "Optimized packing clusters of objects in a rectangular container," *Math. Problems Eng.*, vol. 2019, pp. 1–12, Feb. 2019.

[100] İ. Erozan and E. Çalşkan, "A multi-objective genetic algorithm for a special type of 2D orthogonal packing problems," *Appl. Math. Model.*, vol. 77, pp. 66–81, Jan. 2020.

[101] N. M. Cid-Garcia and Y. A. Rios-Solis, "Positions and covering: A two-stage methodology to obtain optimal solutions for the 2D-bin packing problem," *PLoS ONE*, vol. 15, no. 4, pp. 1–22, 2020.

[102] M. Assi and R. A. Haraty, "A survey of the knapsack problem," in *Proc. Int. Arab Conf. Inf. Technol.*, Nov. 2018, pp. 1–6.

[103] R. I. E. Saragih, N. F. Saragih, and M. Aritonang, "Improving performance genetic algorithm on knapsack problem by setting parameter," *J. Phys. Conf. Ser.*, vol. 1361, Nov. 2019, Art. no. 012034.

[104] V. Poirriez, N. Yanev, and R. Andonov, "A hybrid algorithm for the unbounded knapsack problem," *Discrete Optim.*, vol. 6, no. 1, pp. 110–124, Feb. 2009.

[105] L. A. McLay and S. H. Jacobson, "Algorithms for the bounded set-up knapsack problem," *Discrete Optim.*, vol. 4, no. 2, pp. 206–212, Jun. 2007.

[106] S. Martello and M. Monaci, "Algorithmic approaches to the multiple knapsack assignment problem," *Omega*, vol. 90, Jan. 2020, Art. no. 102004.

[107] B. Schulze, M. Stiglmayr, L. Paquete, C. M. Fonseca, D. Willems, and S. Ruzika, "On the rectangular knapsack problem: Approximation of a specific quadratic knapsack problem," *Math. Methods Oper. Res.*, pp. 1–26, Feb. 2020, doi: 10.1007/s00186-020-00702-0.

[108] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *Eur. J. Oper. Res.*, vol. 246, no. 2, pp. 345–378, Oct. 2015.

[109] C. Luo and G. Zhang, "Single machine scheduling problem with controllable setup and job processing times and position-dependent workloads," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 790, no. 1, p. 6, 2020.

[110] G. P. Georgiadis, B. Mariño Pampín, D. Adrián Cabo, and M. C. Georgiadis, "Optimal production scheduling of food process industries," *Comput. Chem. Eng.*, vol. 134, Mar. 2020, Art. no. 106682.

[111] M. Liu and X. Liu, "Satisfaction-driven bi-objective multi-skill workforce scheduling problem," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 229–234, 2019.

[112] H. Li, H. Zhu, and T. Jiang, "Modified migrating birds optimization for energy-aware flexible job shop scheduling problem," *Algorithms*, vol. 13, no. 2, pp. 1–16, 2020.

[113] J. A. S. Araujo, H. G. Santos, B. Gendron, S. D. Jena, S. S. Brito, and D. S. Souza, "Strong bounds for resource constrained project scheduling: Preprocessing and cutting planes," *Comput. Oper. Res.*, vol. 113, Jan. 2020, Art. no. 104782.

[114] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for University course timetabling problem," *Comput. Ind. Eng.*, vol. 86, pp. 43–59, Aug. 2015.

[115] B. Wang, Y. Geng, and Z. Zhang, "Applying genetic algorithm to University classroom arrangement problem," *J. Phys. Conf. Ser.*, vol. 1325, Oct. 2019, Art. no. 012157.

[116] A. Boloori Arabani and R. Z. Farahani, "Facility location dynamics: An overview of classifications and applications," *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 408–420, Feb. 2012.

[117] J. B. Zuñiga, J. A. Saucedo Martínez, T. E. Salais Fierro, and J. A. M. Saucedo, "Optimization of the storage location assignment and the picker-routing problem by using mathematical programming," *Appl. Sci.*, vol. 10, no. 2, p. 534, Jan. 2020.

[118] A. S. Hameed, B. M. Aboobaider, M. L. Mutar, and N. H. Choon, "A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem," *Int. J. Ind. Eng. Comput.*, vol. 11, no. 1, pp. 51–72, 2020.

[119] K. Ozeki and T. Yamashita, "Spanning trees: A survey," *Graphs Combinatorics*, vol. 27, no. 1, pp. 1–26, Jan. 2011.

[120] S. Kamei, H. Kakugawa, S. Devismes, and S. Tixeuil, "A self-stabilizing 3-approximation for the maximum leaf spanning tree problem in arbitrary networks," *J. Combinat. Optim.*, vol. 25, no. 3, pp. 430–459, Apr. 2013.

[121] H. Li, K. Shi, H. Li, S. Lin, G. Xu, and S. Li, "A new crossover algebra of GA for solving the degree constrained minimum spanning tree problems," *J. Phys. Conf. Ser.*, vol. 1187, no. 4, Apr. 2019, Art. no. 042085.

[122] A. F. Marpaung, "Comparative of prim's and Boruvka's algorithm to solve minimum spanning tree problems," *J. Phys. Conf. Ser.*, vol. 1462, Feb. 2020, Art. no. 012043.

[123] W. Gong and X. Zhou, "A survey of SAT solver," in *Proc. AIP Conf.*, vol. 1836, Jun. 2017, Art. no. 020059.

[124] H. Yamashita, K. Aihara, and H. Suzuki, "Timescales of Boolean satisfiability solver using continuous-time dynamical system," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 84, May 2020, Art. no. 105183.

[125] R. Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia, and M. Goh, "Covering problems in facility location: A review," *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 368–407, Feb. 2012.

[126] R. Hasudungan, D. M. Pangestuty, A. J. Latifah, and Rudiman, "Solving minimum vertex cover problem using DNA computing," *J. Phys. Conf. Ser.*, vol. 1361, Nov. 2019, Art. no. 012038.

[127] Y. Li, Y. Liu, D. Juedes, F. Drews, R. Bunescu, and L. Welch, "Set cover-based methods for motif selection," *Bioinformatics*, vol. 36, no. 4, pp. 1044–1051, Sep. 2019.

[128] H. Wang and L.-A. Wu, "Ultrafast adiabatic quantum algorithm for the NP-complete exact cover problem," *Sci. Rep.*, vol. 6, no. 1, pp. 4–10, Apr. 2016.

[129] N. Fröhlich, A. Maier, and H. W. Hamacher, "Covering edges in networks," *Networks*, vol. 75, no. 3, pp. 278–290, Apr. 2020.

[130] U. Ritzinger, J. Puchinger, and R. F. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *Int. J. Prod. Res.*, vol. 54, no. 1, pp. 215–231, Jan. 2016.

[131] M. Granada-Echeverri, E. M. Toro, and J. J. Santa, "A mixed integer linear programming formulation for the vehicle routing problem with backhauls," *Int. J. Ind. Eng. Comput.*, vol. 10, no. 2, pp. 295–308, 2019.

[132] A. A. Bulatov, "Constraint satisfaction problems: Complexity and algorithms," in *Language and Automata Theory and Applications. LATA* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10792, no. 4. New York, NY, USA: Springer, 2018, pp. 1–25.

[133] J. Bulín, A. Krokhin, and J. Opršal, "Algebraic approach to promise constraint satisfaction," in *Proc. Annu. ACM Symp. Theory Comput.*, 2019, pp. 602–613.

[134] M. Bezenšek and B. Robič, "A survey of parallel and distributed algorithms for the Steiner tree problem," *Int. J. Parallel Program.*, vol. 42, no. 2, pp. 287–319, Apr. 2014.

[135] F. V. Fomin, P. Kaski, D. Lokshtanov, F. Panolan, and S. Saurabh, "Parameterized single-exponential time polynomial space algorithm for Steiner tree," *SIAM J. Discrete Math.*, vol. 33, no. 1, pp. 327–345, Jan. 2019.

[136] R. J. Gould, "Recent advances on the Hamiltonian problem: Survey III," *Graphs Combinatorics*, vol. 30, no. 1, pp. 1–46, Jan. 2014.

[137] F. Keshavarz-Kohjerdi and A. Bagheri, "Linear-time algorithms for finding Hamiltonian and longest (s,t)-paths in C-shaped grid graphs," *Discrete Optim.*, vol. 35, Feb. 2020, Art. no. 100554.

[138] A. Madkour, W. G. Aref, F. Ur Rehman, M. A. Rahman, and S. Basalamah, "A survey of shortest-path algorithms," 2017, *arXiv:1705.02044*. [Online]. Available: http://arxiv.org/abs/1705.02044

[139] Y. Qiu, A. Zou, P. Chen, and L. Xu, "GPU-accelerated fast implementation of shortest path algorithm in the noise simulation analysis system," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 768, Mar. 2020, Art. no. 072035.

[140] E. Duesbury, J. D. Holliday, and P. Willett, "Maximum common subgraph isomorphism algorithms," *Match*, vol. 77, no. 2, pp. 213–232, 2017.

[141] J. Malík, O. Suchý, and T. Valla, "Efficient implementation of color coding algorithm for subgraph isomorphism problem," in *Analysis of Experimental Algorithms* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11544. New York, NY, USA: Springer, 2019, pp. 283–299.

[142] K. K. Singh and D. A. K. Pandey, "Survey of algorithms on maximum clique problem," *Int. Adv. Res. J. Sci., Eng. Technol.*, vol. 2, no. 2, pp. 15–20, Feb. 2015.
[143] H. Dau, O. Milenkovic, and G. J. Puleo, "On the triangle clique cover and Kt clique cover problems," *Discrete Math.*, vol. 343, no. 1, Jan. 2020, Art. no. 111627.
[144] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, Jun. 2008.
[145] N. V. Sahinidis, "Optimization under uncertainty: State-of-the-art and opportunities," *Comput. Chem. Eng.*, vol. 28, nos. 6–7, pp. 971–983, Jun. 2004.
[146] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *Eur. J. Oper. Res.*, vol. 225, no. 1, pp. 1–11, 2013.
[147] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, Jan. 1993.
[148] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *J. Intell. Manuf.*, vol. 29, no. 3, pp. 603–615, Mar. 2018.
[149] I. A. Chaudhry, A. M. Khan, and A. A. Khan, "A genetic algorithm for flexible job shop scheduling," *Lect. Notes Eng. Comput. Sci.*, vol. 1, pp. 703–708, Jul. 2013.
[150] I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand, and J. Wassick, "Scope for industrial applications of production scheduling models and solution methods," *Comput. Chem. Eng.*, vol. 62, pp. 161–193, Mar. 2014.
[151] A. C. Hax and H. C. Meal, *Hierarchical Integration of Production Planning and Scheduling*. New York, NY, USA: Elsevier, 1973.
[152] F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *J. Cleaner Prod.*, vol. 67, pp. 197–207, Mar. 2014.
[153] 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.
[154] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 3, pp. 462–467, 2017.
[155] C. Lin, K. L. Choy, G. T. S. Ho, S. H. Chung, and H. Y. Lam, "Survey of green vehicle routing problem: Past and future trends," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1118–1138, Mar. 2014.
[156] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, Apr. 1987.
[157] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2017.
[158] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.
[159] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[160] R. K. Arora, "1D optimization algorithms," in *Optimization: Algorithms and Applications*. Boca Raton, FL, USA: CRC Press, 2015, pp. 35–54.
[161] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, no. 3, pp. 141–152, 1985.
[162] H. Hoos and T. Stützle, "MAX MIN ant system," *Future Gener. Comput. Syst.*, vol. 16, Nov. 1999, pp. 889–914, 2000.
[163] U. Feige, "A threshold of ln n for approximating set cover," *J. ACM*, vol. 45, no. 4, pp. 634–652, Jul. 1998.
[164] J. Hastad, "Some optimal inapproximability results," in *Proc. Conf. Annu. ACM Symp. Theory Comput.*, 1997, vol. 48, no. 4, pp. 1–10.
[165] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2005, pp. 1976–1984.
[166] R. Impagliazzo, R. Paturi, and F. Zane, "Which problems have strongly exponential complexity?" *J. Comput. Syst. Sci.*, vol. 63, no. 4, pp. 512–530, Dec. 2001.
[167] B. S. Baker, "Approximation algorithms for NP-complete problems on planar graphs," *J. ACM*, vol. 41, no. 1, pp. 153–180, Jan. 1994.
[168] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artif. Intell.*, vol. 49, nos. 1–3, pp. 61–95, May 1991.
[169] A. K. Mackworth, *Consistency in Networks of Relations*, vol. 8. San Mateo, CA, USA: Morgan Kaufmann, 1977.
[170] A. A. Bulatov, "A dichotomy theorem for nonuniform CSPs," in *Proc. Annu. Symp. Found. Comput. Sci.*, Oct. 2017, pp. 319–330.
[171] T. Feder and M. Y. Vardi, "Monotone monadic SNP and constraint satisfaction," in *Proc. Annu. ACM Symp. Theory Comput.*, vol. F1295, 1993, pp. 612–622.
[172] T. Feder and M. Y. Vardi, "The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory," *SIAM J. Comput.*, vol. 28, no. 1, pp. 57–104, Jan. 1998.
[173] D. Zhuk, "A proof of CSP dichotomy conjecture," in *Proc. Annu. Symp. Found. Comput. Sci.*, Oct. 2017, pp. 331–342.

MELISSA SÁNCHEZ was born in Culiacán, Sinaloa, Mexico, in 1996. She is currently pursuing the B.Sc. degree in mechatronics engineering with the Tecnológico de Monterrey.

From 2017 to 2019, she worked as a Laboratory Instructor at the Tecnológico de Monterrey. Since 2018, she has been a part of the Research Group with Strategic Focus in Autonomous Vehicles, Tecnológico de Monterrey. Since 2019, she has also been working as a Harness Design Intern at John Deere and a Research Assistant at the Research Group with Strategic Focus in Intelligent Systems, Tecnológico de Monterrey. Her research interests include underwater autonomous vehicles (UAVs), combinatorial optimization problems solved through hyper-heuristics, and power electronics.

JORGE M. CRUZ-DUARTE (Member, IEEE) was born in Ocaña, Colombia, in 1990. He received the B.Sc. and M.Sc. degrees in electronics engineering from the Universidad Industrial de Santander, Bucaramanga, Colombia, in 2012 and 2015, respectively, and the Ph.D. degree in electrical engineering from the Universidad de Guanajuato, Guanajuato, Mexico, in 2018.

Since 2019, he has been a Postdoctoral Fellow with the Research Group with Strategic Focus in Intelligent Systems, Tecnológico de Monterrey, Monterrey, Mexico. His research interests include data science, optimization, mathematical methods, thermodynamics, digital signal processing, electronic thermal management, and fractional calculus. He is a member of the Mexican National System of Researcher.

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

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.

HECTOR CEBALLOS was born in Coatzacoalcos, Heroica Veracruz, Mexico, in 1977. He received the B.Sc. degree in computer science engineering from the Instituto Tecnológico de Veracruz, in 2000, and the M.Sc. and Ph.D. degrees in intelligent systems from the Tecnológico de Monterrey, Monterrey, Mexico, in 2003 and 2010, respectively.

Since 2010, he has been the Head of the Scientometrics Office and the Research Vice-Rectory of the Tecnológico de Monterrey. He is currently ascribed to the Research Group with Strategic Focus in Intelligent Systems, Tecnológico de Monterrey. His main research interests include social network analysis, process mining and agent theory, applied to research analytics, and cultural heritage. He is the author of more than 30 papers in journals and conferences, has worked as an expert consulting for bank and IT companies, and promoted the adoption of Semantic Web technologies in academy, government, and industry. He is a member of the Mexican National System of Researcher (SNI) and an Adherent Member of the Mexican Academy on Computing (AMEXCOMP).

HUGO TERASHIMA-MARÍN (Senior Member, IEEE) was born in Minas de Barroterán, Coahuila, Mexico, in 1961. He received the B.Sc. degree in computational systems from the Tecnológico de Monterrey, Monterrey, in 1982, the M.Sc. degree in computer science from The University of Oklahoma, in 1987, the second M.Sc. degree in information technology and knowledge-based systems from The University of Edinburgh, in 1994, and the Ph.D. degree in informatics from the Tecnológico de Monterrey, in 1998.

He is currently a Full Professor with the School of Engineering and Sciences, the Leader of the Research Group with Strategic focus in Intelligent Systems, and the Director of the Graduate Program in Computer Science. His research interests include computational intelligence, heuristics, metaheuristics and hyper-heuristics for combinatorial optimization, characterization of problems and algorithms, constraint handling and applications of artificial intelligence, and machine learning. He is a member of the National System of Researchers, the Mexican Academy of Sciences, and the Mexican Academy of Computing.

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

From 2016 to 2018, he was a Postdoctoral Fellow with the Research Group with Strategic Focus in 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 continuous and discrete problems, through the application of heuristics, metaheuristics, and hyper-heuristics. For the latter, he focuses on finding new ways of using feature transformations for improving performance. He is a member of the Mexican National System of Researchers, the Mexican Academy of Computing, and the Association for Computing Machinery.