Received 1 December 2022, accepted 4 January 2023, date of publication 13 January 2023, date of current version 25 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3236836

RESEARCH ARTICLE

Automatic Design of Metaheuristics for Practical Engineering Applications

DANIEL F. ZAMBRANO-GUTIERREZ[1], JORGE MARIO CRUZ-DUARTE[1], (Member, IEEE), JUAN GABRIEL AVINA-CERVANTES[2], JOSÉ CARLOS ORTIZ-BAYLISS[1], (Member, IEEE), JESÚS JOAQUÍN YANEZ-BORJAS[2], AND IVÁN AMAYA[1], (Senior Member, IEEE)

[1]Advanced Artificial Intelligence Research Group, School of Engineering and Science, Tecnologico de Monterrey, Monterrey, Nuevo León 64849, Mexico
[2]Telematics Research Group, Department of Electronics Engineering, University of Guanajuato, Comunidad de Palo Blanco, Salamanca, Guanajuato 36885, Mexico

Corresponding author: Jorge Mario Cruz-Duarte (jorge.cruz@tec.mx)

This work was supported in part by the Tecnológico de Monterrey under Project FAP 1859, in part by the University of Guanajuato under Project CIIC 171/2022, and in part by the Mexican Council of Science and Technology (CONACyT) under Grant 1046000 and Grant 287479.

ABSTRACT It is common to find multiple metaheuristics to solve continuous optimization problems. However, choosing what optimizer may obtain the best results for a given task requires exhaustive evaluations that are highly application-dependent. Besides, it is necessary to find sufficiently good tuning parameters to achieve satisfactory performance with the selected approach. In this context, the automatic design of algorithms, particularly those based on heuristics, has been increasing in popularity in the previous years due to its undoubted relevance nowadays. This paper explores a novel approach based on hyper-heuristics to carefully select population-based search operators and their tuning parameters to generate metaheuristics capable of dealing with a given practical engineering problem. The proposed strategy is assessed using three highly relevant and illustrative problems: training Artificial Neural Networks, designing PID controllers, and modeling a calorimetric phenomenon based on fractional calculus. In addition, we implement three well-known optimization metaheuristics to compare achieved solutions via the proposed hyper-heuristic strategy, namely Particle Swarm Optimization, Genetic Algorithm, and Cuckoo Search. Results from extensive numerical tests prove that the customized metaheuristics are generally superior to the three well-known algorithms, taking only a few iterations to converge to an optimal solution. This is an excellent indicator of alleviating the effort and expertise required to choose the proper methodology when dealing with real-valued optimization problems.

INDEX TERMS Metaheuristics, hyper-heuristics, PID controllers, artificial neural networks, fractional model design, control theory, fractional calculus.

I. INTRODUCTION

Optimization methods have become a research topic of great interest, especially when designing complex engineering systems [1], [2], [3]. This may be something other than a trendy topic shadowed by sophisticated Artificial Intelligence techniques nowadays. Still, they are the theoretical and practical cornerstones of many modern applications, even those sophisticated methods. Technological advances have enabled it to develop new engineering processes, leading to further optimization applications. From a practical point of view, optimization consists of adjusting or fine-tuning system design parameters based on one or more performing functions. Unfortunately, this task is not trivial in many cases, especially when the estimated searching space is complex, nonlinear, discontinuous, or presents high dimensionality [4], [5], [6], [7]. Indeed, most real-world engineering problems present high design complexity, nonlinear constraints, and vast solution domains. These issues justify using modern techniques capable of dealing with challenging optimization

The associate editor coordinating the review of this manuscript and approving it for publication was Yeliz Karaca.

case studies. One family of these techniques comprises Metaheuristic (MH) algorithms. MHs have become an alternative to gradient-based optimization methods, especially when the latter evolves into ill-conditioned scenarios during the running time in the application. Thereunto, MHs are characterized by their flexibility, versatility, and algorithmic simplicity when dealing with optimization models [8].

Nevertheless, MHs are not magical. One must know how to select the most suitable, stable, and fast (if so) MH for a given case study. Notwithstanding, it is often necessary to know how to configure its internal parameters efficiently to obtain the highest method performance in terms of accuracy and the number of iterations required for global convergence. The levels of expertise and problem knowledge of the designer, engineer, or practitioner play a crucial role in successfully implementing these methodologies. No matter its flavors, the No-Free-Lunch theorem still reigns [9]. Various MHs have been proposed and evaluated over the last decades to solve a colorful palette of challenging problems [10]. However, some of the newly developed metaheuristics do not differ substantially from the general structures of well-reputed and conventional MHs such as Simulated Annealing (SA) [11], Differential Evolution (DE) [12], Genetic Algorithms (GA) [13], Grey-Wolf Optimizer (GWO) [14], and Particle Swarm Optimization (PSO) [15], to mention a few.

This study considers these robust MHs as basic heuristics for being analyzed and extracting their Search Operators (SOs) or simple heuristics to generate metaphor-less MHs automatically. In particular, we observe that many SOs are straightforward methodological or mathematical variations of fundamental optimization blocks, *e.g.*, mutation, crossover, fitness functions, stopping criteria, and random searches. Some previous studies have used distinctive functional blocks of impressive methods to develop procedures to select two or more simple heuristics to obtain improved MHs in specialized applications [16]. However, finding a particular MH model with the proper tuning parameters is complex, and developing new strategies allows us to coin problem-specific methods or algorithms for problems sharing certain features. Hence, contemporary engineering design challenges requiring robust optimization algorithms are why combinatorial algorithms have emerged in this direction [17]. The fast progress of hybrid MHs has led to the development of a new optimization sub-area known as Hyper-Heuristics (HHs) [18]. These also heuristic-based algorithms deal with high-level problems by selecting and modifying simple (or low-level) heuristics to, in consequence, find a good quality solution in the low-level problem domain [19]. In other words, HHs focus on automating the model design and adapting the existing heuristic methods to address the search task more efficiently. In addition, these approaches aim to develop the level of generality at which search methodologies can operate [20]. Because of such versatility, HHs have been efficiently implemented in the literature, such as in combinatorial problems [21]. Looking in this direction, it is worth highlighting reports implementing HH models that rely on simple heuristic-based algorithms such as the so-called SA. The grand reception of Simulated Annealing as a high-level solver is undoubtedly explained by its characteristic simplicity, low computation burden, and remarkable effectiveness [18]. For example, Bai et al. developed a HH framework for solving two problems related to flexible decision support, *i.e.*, scheduling university courses and packing garbage cans [22]. Their results showed that the SA-based HH improved the performance considerably over a Tabu-based HH strategy. Likewise, Garza-Santisteban et al. proposed a high-level solver based on SA for selecting the best heuristic sequence that fulfills the requirements of multiple instances of Job Shop Scheduling problems [23]. Moreover, several hybrid approaches incorporating SA-based processes are easily found in the literature [24], [25], [26]. For example, Mosadegh et al. presented a hyper-heuristic based on Q-Learning and Simulated Annealing for dealing with the mixed-model sequencing problem, including stochastic processing times in a multi-station assembly line [27]. Nevertheless, we find challenging to find reports in the literature that address practical engineering applications with continuous domains as low-level problems using any hyper-heuristic framework. Still, only a few approaches have been reported on benchmark continuous optimization problems [28], [29].

This work studies the implementation and feasibility of an automatic algorithm design strategy for generating metaheuristics tailored for practical engineering applications. In that regard, we use the hyper-heuristic framework proposed in [29] and [30] , which contains a collection of search operators extracted from common metaheuristics and an SA-based high-level solver. For the practical problems, we consider training Artificial Neural Networks (ANNs), designing Proportional–Integral–Derivative (PID) controllers, and modeling calorimetric processes via Fractional Calculus (FC). Therefore, we can structurally select the operators to generalize a solution and obtain optimal results for the conditions of each application. Moreover, it counts on high repeatability and performance to standardize operators for each task. We observe from the results that the MHs generated by the HH approach outperform basic MHs. Therefore, we provide the practitioners with an automatic methodology for finding MHs to deal with real optimization problems. So, the principal innovation of our proposal is that the users require a minimum level of expertise in metaheuristics to find one suitable for their needs. Plus, the main contributions of this study can be summarized as follows:

1) Prove the feasibility of the HHs for automatically designing solvers to deal with practical problems.
2) Illustrate an alternative way, based on metaphor-less MH, to tackle three dissimilar problem families.
3) Analyze the performance of the generated MH in terms of accuracy, the number of steps, and repeatability.

This document is organized as follows. Sect. II presents the theoretical foundations supporting all methods this work covers. Sect. III describes the implemented framework to analyze the case studies. Sect. IV presents a theoretical overview of the proposed applications and how the implemented methodology deals with them. Finally, Sect. V gives the most relevant conclusions.

II. THEORETICAL FOUNDATIONS

A. OPTIMIZATION

Optimization, also called mathematical programming, is focused on estimating the best model parameters that fit the available data, following some criteria among existing alternatives. Therefore, an optimization problem maximizes or minimizes an objective function, achieving the best outcome in practical scenarios. Strictly speaking, the optimization methods search and select the best available values of an objective function, supported mathematically by the following definitions.

Definition 1 (Arbitrary Problem Domain): Let $\mathfrak{X} \subseteq \mathfrak{S}$ be a feasible domain since $\mathfrak{S}$ is an arbitrary domain. This arbitrary domain can be defined according to the problem one is solving. It depends on the level of abstraction, such as continuous, combinatorial, and mixed.

Remark 1 (Particular Problem Domains): For continuous problem domains $\mathfrak{S} = \mathbb{R}^D$, where D stands for the number of dimensions, which is usually fixed and represents the number of design variables in design problems. For combinatorial problems $\mathfrak{S} = \mathbb{Z}^{\varpi}$, with ϖ is also the number of dimensions or cardinality. For mixed domains, we say that $\mathfrak{S} = \mathbb{R}^D \times \mathbb{Z}^{\varpi}$. Lastly, a particular case from the combinatorial domains is the heuristic space, such that $\mathfrak{S} = \mathfrak{H}^{\varpi}$ is the heuristic space.

Definition 2 (Minimization Problem): Let $f(\vec{x})$ be a function defined on the set $\mathfrak{X} \neq \emptyset$ such as $f(\vec{x}) : \vec{x} \in \mathfrak{X} \subseteq \mathbb{R}^D \to \mathbb{R}$. Therefore, the minimization problem is stated as

$$\vec{x}_* = \underset{\vec{x} \in \mathfrak{X}}{\operatorname{argmin}} \{f(\vec{x})\}, \tag{1}$$

where $\vec{x}_* \in \mathfrak{X}$ is the optimal solution that minimizes the objective function, *i.e.*, $f(\vec{x}_*) \leq f(\vec{x}), \forall \vec{x} \in \mathfrak{X}$. For the sake of simplicity, we refer to a minimization problem with the tuple comprising its domain and function, such as $(\mathfrak{X}, f)$.

B. HEURISTICS

A heuristic is a procedure or operator capable of modifying or generating new potential solutions for optimization problems. Although one can find several definitions in the literature, most describe heuristics in combinatorial optimization domains [31]. However, there are a few cases in continuous domains [19]. In general, heuristics can be classified into three groups: low-level, mid-level, and high-level heuristics [19], [31], which are also known as Simple Heuristics (SHs), Metaheuristics (MHs), and Hyper-Heuristics (HHs). In the following lines, we briefly describe each of these groups; one can find further details in [29] and [32].

C. SIMPLE HEURISTICS

Simple heuristics are a fundamental part of search techniques. SHs directly address the problem domain, and usually are *constructive or perturbative*. Constructive heuristics generate novel solutions from scratch, whereas perturbative heuristics modify or perturb current solutions [33]. However, at the MH control level, we require that SHs can validate whether the process should be stopped or continued. In simple terms, the finalizer is an essential pillar controlling the search procedure many authors use to define MH [34]. Bearing this in mind, we briefly describe simple heuristics and their three categories. These concepts apply to an arbitrary problem domain $\mathfrak{S}$.

Definition 3 (Simple Heuristic): Let $\mathfrak{H}$ be a set of simple heuristics, or heuristic space, with a composition operation $\circ : \mathfrak{H} \times \mathfrak{H} \mapsto \mathfrak{H}$. Let $\mathfrak{H}_i, \mathfrak{H}_0, \mathfrak{H}_f \subset \mathfrak{H}$ be subsets of heuristics that produces, modifies, and chooses between two operators, respectively.

Definition 4 (Initializer): Let $h_i : \mathfrak{S} \mapsto \mathfrak{X} \subseteq \mathfrak{S}$ be a simple heuristic that generates a candidate solution $\vec{x} \in \mathfrak{X}$ within the feasible domain from scratch, i.e., $\vec{x} = h_i\{\mathfrak{X}\}$.

Definition 5 (Search Operator): Let $h_o : \mathfrak{X} \mapsto \mathfrak{X}$ be a simple heuristic that obtains a new position $\vec{x}(t+1) \in \mathfrak{X}$ from the current position $\vec{x}(t) \in \mathfrak{X}$, since t indicates the current iteration. A search operator mostly comprises two basic operations: perturbation and selection. Hence, let $h_p, h_s \in \mathfrak{H}_o$ be also simple heuristics (perturbator and selector) that modify and update the current solution $\vec{x}(t)$, respectively; then $\vec{y} = h_p\{\vec{x}\}$ and $\vec{x}(t+1) = h_s\{\vec{y}\}$. A perturbator always precedes a selector, so $h_o = h_s \circ h_p$.

Definition 6 (Finalizer): Let $h_f : \mathfrak{X} \times \mathbb{Z}_2 \mapsto \mathfrak{H}$ be a simple heuristic that evaluates the current solution quality and chooses which search operator to apply. To do so, it uses information about the iterative procedure in a criteria function $c_f : (\mathbb{Z}_+, \mathbb{R}, \mathfrak{X}, \ldots) \mapsto \mathbb{Z}_2$. Then, $h_f \in \mathfrak{H}_f$ is a finalizer given by

$$h_f\,(h_0)\,\{\vec{x}\} \triangleq \begin{cases} h_e\{\vec{x}\}, & \text{if } c_f(t, f, X, \ldots) = 1, \\ h_f \circ h_0\{\vec{x}\}, & \text{otherwise,} \end{cases} \tag{2}$$

since h_0 is a search operator and h_e is the identity operator.

D. METAHEURISTICS

Metaheuristics are commonly defined refined strategies that control simple heuristics. MHs are in vogue in the literature due to their proven performance in different challenging scenarios, outperforming traditional gradient-based strategies [10], [35]. MHs are widely recognized because of their capabilities and performance in optimally solving complex engineering models [35]. Some representative MHs include the Differential Evolution [12], Particle Swarm Optimization [15], Cuckoo Search Algorithm [36], Spiral Optimization Algorithm [37], and Grey-Wolf Optimization [14], among others [38]. Definition 7 formally describes the general scheme for a metaheuristic according to [32].

Definition 7 (Metaheuristic): Let MH be an iterative metaheuristic procedure that renders an optimal solution $\vec{x}_*$ for a given optimization task using an objective function $f(\vec{x})$ (*cf.* Definition 2). This procedure is represented as a finite sequence of simple heuristics (*cf.* Definition 3) to be applied iteratively until fulfilling a stopping criterion, *i.e.*,

$$\mathrm{MH}_o \triangleq \langle h_i, h_o, h_f \rangle = h_f\,(h_o) \circ h_i, \tag{3}$$

since $h_i \in \mathfrak{H}_\mathrm{i}$ is an initializer, $h_f \in \mathfrak{H}_f$ is a finalizer, and $h_o \in \mathfrak{H}_o$ is a search operator.

E. HYPER-HEURISTICS

The literature describes hyper-heuristics as high-level heuristics that control simple heuristics in optimization solving a given case study [18]. A HH either selects or generates low-level heuristics used to propose hybrid algorithms to address optimization tasks better. They differ from most MH applications because they typically work with the solution search space. So, a HH can be defined, according to Pillay and Qu [19], as follows.

Definition 8 (Hyper-Heuristic): Let $\vec{h} \in H \subseteq \mathfrak{H}^{\varpi}$ be a heuristic configuration from a feasible heuristic collection H within the heuristic space $\mathfrak{H}$. Let $\mathrm{perf}(\vec{h} \mid \mathfrak{X}, f) : H \times \mathfrak{X} \mapsto \mathbb{R}$ be a metric that measures the performance of $\vec{h}$ when it is applied on $(\mathfrak{X}, f)$. Therefore, a hyper-heuristic is a technique that solves

$$\vec{h}_* = \underset{\vec{h} \in H}{\mathrm{argmax}}\{\mathrm{perf}(\vec{h} \mid \mathfrak{X}, f)\}. \tag{4}$$

Hence, this technique searches for the optimal heuristic configuration $\vec{h}_*$ that best approaches to the solution of $(\mathfrak{X}, f)$ with the maximal performance.

Remark 2 (Performance Metric): There is no unique expression for determining the performance, $\mathrm{perf}(\vec{h} \mid \mathfrak{X}, f)$ since it depends on the desired requirements for the heuristic sequence $\vec{h}$. A numerical and practical way to assess this measurement, due to the stochastic nature of almost all the heuristic sequences, is to combine different statistics from several independent runs of $\vec{h}$ over the same problem $(\mathfrak{X}, f)$.

In this work, we employed the performance metric

$$\mathrm{perf}(X_*) = -(\mathrm{med} + \mathrm{iqr})\left(\left\{\forall\, \vec{x}_{r,*} \in X_* | f(\vec{x}_{r,*})\right\}\right), \tag{5}$$

which comprises the negative sum of median and interquartile range of the last fitness values $f\left(\vec{x}_{*,r}\right)$ achieved in N_r runs, $r = 1, \ldots, N_r$, of the same solver.

III. METHODOLOGY

All the experiments in this work were carried out in Python v3.9 running on an ASUS TUF Gaming F17 with AMD Ryzen™ 7 Processor 5700G @ 8 CPU Cores, 16GB RAM, and Windows 10 64-bit. We used the framework CUSTOMHyS v1.0 at https://github.com/jcrvz/customhys and described by Cruz-Duarte et al. to implement the HH search [30]. Figure 1 summarizes the proposed methodology for obtaining optimal metaheuristics for three case studies. For this purpose, we employed Simulated Annealing (SA) to power our approach, which has been widely implemented as a high-level solver for hyper-heuristic applications and is also coded in CUSTOMHyS [30].

This SA-based HH requires the initial hyper-parameters (such as initial temperature, cooling rate, number of steps, and stagnation count), the low-level optimization problem $(\mathfrak{X}, f)$, and the heuristic space (collection of search operators). For the hyper-parameters, we have utilized those provided by default in CUSTOMHyS [30]. Concerning the high-level problem domain, we employed the heuristic space summarized in Table 1. These search operators are composed of a perturbator and a selector.

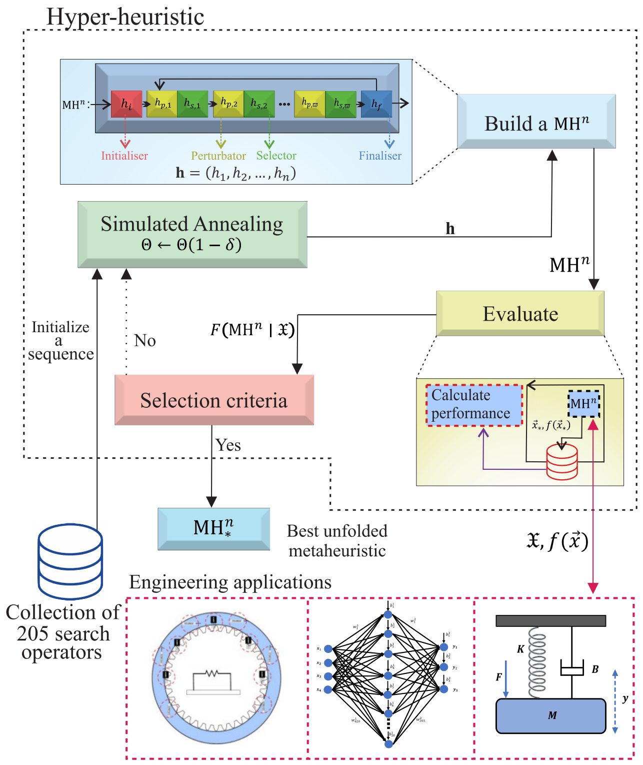


FIGURE 1. Hyper-heuristic methodology implemented to generate optimal metaheuristics for three practical problems.

TABLE 1. Collection of search Operators employed as the heuristic space. These are composed by a perturbator, its tuning parameters, and a default selector, which can be changed.

Perturbator	Tuning parameters	Selector
differential_crossover	crossover_rate	greedy
differential_mutation	num_rands, factor	all
genetic_crossover	mating_pool_factor	all
genetic_mutation	scale, elite_rate, mutation_rate	greedy
swarm_dynamic	factor, self_conf, swarm_con	all
firefly_dynamic	alpha, beta, gamma	all
spiral_dynamic	radius, angle, sigma	all
random_flight	scale, beta	greedy
random_sample	-	all
random_search	scale	greedy
local_random_walk	probability, scale	greedy
central_force_dynamic	gravity, alpha, beta, dt	all
gravitational_search	gravity, alpha	all

Their names are often related to the MH from which they were extracted.

Now, in the high-level problem domain, we utilized a collection of 205 SOs as the heuristic space. This domain is briefly shown in Table 2, where we regard different variations and predefined values for the heuristics tuning parameters according to the data summarized in Table 1. In Table 2, each row groups four versions of the same operator but with a systematic variation of the selector, such as Direct, Greedy, Metropolis, and Probabilistic, except the first row, Random Search, which has a predefined Greedy selector. For the eager reader, Cruz-Duarte et al. provide further details about this collection [30].

In addition, Pseudocode 1 describes the procedure associated with the proposed methodology shown in Figure 1. Consider that all the terminology, symbols, and concepts employed are coherent with the theoretical foundations presented in Sect. II. In simple words, the automatic design strategy of metaheuristics proposed in this work corresponds to the well-known Simulated Annealing as a hyper-heuristic. Such a fact is easy to notice from the backbone of Pseudocode 1, where the first block stands for initialization. Then the candidate heuristic sequence is modified and evaluated in the low-level problem in the main loop, followed by the Metropolis selection criterion. All is repeated until the temperature variable reaches a minimal value or the overall procedure is stagnated. The slight but substantial difference between this SA implementation with a typical one is the heuristic space exploration. This process is carried out by performing a randomly selected action from an action set (*i.e.*, add, delete, and perturb) to find a candidate neighbor. Additional details about this implementation can be found in [29] and [30]. Bear in mind that for this work, we slightly adjusted the methods provided by the CUSTOMHyS framework to study the feasibility of this strategy on practical engineering problems, represented by $(\mathfrak{X}, f)$.

Lastly, the three case studies chosen for this work were applications related to Feedforward Neural Networks, Control Systems, and Thermal Modeling based on Fractional Calculus. Each of these cases is detailed in the next section. Nevertheless, it is essential to highlight that each case study has different problem dimensionality. Besides, we repeated each optimization process 50 times to ensure statistical significance, and we used a population size of 30 and a maximum of 30 hyper-heuristic steps. Plus, we analyzed the performance of MHs generated by the hyper-heuristic approach against three traditional MHs from the literature for each of the three case studies. These metaheuristics are Particle Swarm Optimization (PSO), Cuckoo Search (CS), and Gravitational Search Algorithm (GSA). We selected these MHs due to their implementations in similar applications [39], [40], [41], [42], [43], [44], [45], [46], [47].

IV. PRACTICAL APPLICATIONS

This section describes the three case studies we selected to illustrate the automatic design of metaheuristics. We carried out all the experiments from these cases following the methodology described above, and particularities are specified when necessary.

TABLE 2. Collection of 205 search operators employed in this work. Each row systematically groups four versions of the same operator, varying the selector (such as Direct, Greedy, Metropolis, and Probabilistic), except for the first row that corresponds to Random Search with a Greedy selector.

Ids.	Operator family	Variation
0	Random Search	–
1-4	Central Force Dynamic	–
5-8	Differential Mutation	/rand/
9-12		/best/
13-16		/current/
17-20		/current-to-best/
21-24		/rand-to-best/
25-28		/rand-to-best-and-current/
29-32	Firefly Dynamic	Uniform distribution
33-36		Normal distribution
37-40		Lévy distribution
41-44	Genetic Single-point Crossover	Rank Weighting Pairing
45-48		Roulette Wheel Pairing
49-52		Random Pairing
53-56		Tournament Pairing
57-60	Genetic Two-points Crossover	Rank Weighting Pairing
61-64		Roulette Wheel Pairing
65-68		Random Pairing
69-72		Tournament Pairing
73-76	Genetic Uniform Crossover	Rank Weighting Pairing
77-80		Roulette Wheel Pairing
81-84		Random Pairing
85-88		Tournament Pairing
89-92	Genetic Blend Crossover	Rank Weighting Pairing
93-96		Roulette Wheel Pairing
97-100		Random Pairing
101-104		Tournament Pairing
105-108	Genetic Linear Crossover	Rank Weighting Pairing
109-112		Roulette Wheel Pairing
113-116		Random Pairing
117-120		Tournament Pairing
121-124	Genetic Mutation	Uniform distribution
125-128		Normal distribution
129-132		Lévy distribution
133-136	Gravitational Search	–
137-140	Random Flight	Lévy distribution
141-144		Uniform distribution
145-148		Normal distribution
149-152	Local Random Walk	Uniform distribution
153-156		Normal distribution
157-160		Lévy distribution
161-164	Random Sample	–
165-168	Random Search	Uniform distribution
169-172		Normal distribution
173-176		Lévy distribution
177-180	Spiral Dynamic	–
181-184	Inertial Swarm Dynamic	Uniform distribution
185-188		Normal distribution
189-192		Lévy distribution
193-196	Constricted Swarm Dynamic	Uniform distribution
197-200		Normal distribution
201-204		Lévy distribution

A. TRAINING FEEDFORWARD NEURAL NETWORKS

The first practical case study concerns a Machine Learning application, as follows. A Feedforward Neural Networks (FNN) is a particular architecture type of Artificial Neural Networks (ANNs) [48], [49], [50]. Their main

Pseudocode 1 Automatic Design of Metaheuristics Based on Simulated Annealing for Practical Engineering Applications

```
Input: Practical problem's domain $\mathfrak{X}$ and objective function $f(\vec{x})$. Heuristic
    space $\mathfrak{H}_o$, initializer $h_i$, finalizer $h_f$, performance metric perf$(X_*)$,
    population size $N$, and action set $A$. Initial $\Theta_0$ and minimal $\Theta_{\min}$
    temperature, cooling rate $\delta$, maximum cardinality $\varpi_{\max}$, maximum
    number of iterations $t_{\max}$, and stagnation threshold $n_{\max}$.
Output: Best metaheuristic $\langle h_i, \vec{h}_*, h_f \rangle$

 1: $\vec{h} \leftarrow$ Init($\mathfrak{H}_o$)      ▷ Initialize with uniform randomly selected heuristics
 2: $\vec{h}_* \leftarrow \vec{h}$, $n \leftarrow 0$, and $\Theta \leftarrow \Theta_0$      ▷ Initialize other variables
 3: $Q \leftarrow$ EvalPerformance($\vec{h}$)

 4: while $\Theta > \Theta_{\min}$ and $n \leq n_{\max}$ do
 5:     $a \leftarrow$ Choose($A$)      ▷ Choose randomly an action considering $\varpi$
 6:     $\vec{h}_c \leftarrow a\{\vec{h}\}$      ▷ Apply the action to find the candidate neighbor
 7:     $Q_c \leftarrow$ EvalPerformance($\vec{h}_c$)

 8:     $n \leftarrow n + 1$      ▷ Increase the stagnation counter
 9:     if $r \sim \mathcal{U}(0,1) \leq e^{-(Q_c - Q)/\Theta}$ then      ▷ Metropolis selection
10:         $\vec{h}_* \leftarrow \vec{h}_c$ and $Q \leftarrow Q_c$      ▷ Update the current solution
11:         $n \leftarrow 0$      ▷ Reset the stagnation counter
12:     $\Theta \leftarrow \Theta(1 - \delta)$      ▷ Decrease the temperature
13: return $\langle h_i, \vec{h}_*, h_f \rangle$      ▷ Return the best metaheuristic

14: function EvalPerformance($\vec{h}$)
15:     $X_* \leftarrow \emptyset$      ▷ Initialize the set of solutions
16:     for $r = 1$ to $N_r$ do      ▷ Perform repetitions required in (5)
17:         $\vec{x}_{r,*} \leftarrow$ EvalMH($\vec{h}_c$)      ▷ Use $\langle h_i, \vec{h}_c, h_f \rangle$ to solve $(\mathfrak{X}, f)$
18:         $X_* \leftarrow X_* \cup \{\vec{x}_{r,*}\}$      ▷ Save the current solution
19:     return perf$(X_*)$      ▷ Return the performance using (5)

20: function EvalMH($\vec{h}$)
21:     $t \leftarrow 0$      ▷ Initialize iteration counter
22:     $X \leftarrow \{h_i\{\mathfrak{X}\} \mid \forall n = 1, \ldots N\}$      ▷ Initialize population
23:     $F \leftarrow \{f(\vec{x}_n) \mid \forall \vec{x}_n \in X\}$      ▷ Obtain fitness values
24:     $\vec{x}_* \leftarrow \vec{x}_k \in X$ since $k = \text{argmin}\{F\}$      ▷ Current best
25:     while $h_f\{\vec{x}_*\}$ do      ▷ Apply the finalizer
26:         $t \leftarrow t + 1$      ▷ Increase iteration counter
27:         for $o = 1$ to $\varpi$ do      ▷ Apply the $\varpi = \#\vec{h}$ search operators
28:             $h_p, h_s \leftarrow h_o \in \vec{h}$      ▷ Read perturbator and selector
29:             $X, F \leftarrow h_p\{X\}$      ▷ Apply the $o^{\text{th}}$ perturbator
30:             $\vec{x}_* \leftarrow h_s\{X\}$      ▷ Apply the $o^{\text{th}}$ selector
31:     return $\vec{x}_*(\varpi)$      ▷ Return the best candidate solution
```

attraction remains in letting us perceive the computational models structurally to simplify their analysis. So, the neuronal interconnections in this architecture generate a unidirectional information flow, *i.e.*, the signal never passes more than once through a neuron before generating the output response, as shown in Figure 2.

FNNs have become famous for several reasons. First, they usually generalize well in practice [51], *i.e.*, when trained with a large data set, they often provide a correct output for an input not contained in such a training set. Second, the neural training is performed with a reliable algorithm, the well-known Back-Propagation, in a reasonable number of iterations [52]. This algorithm is used to compute the gradient of all weights errors for a given input by propagating the error backward throughout the network. In the end, the training process concludes by determining the optimal weights and biases in the neuronal architecture.

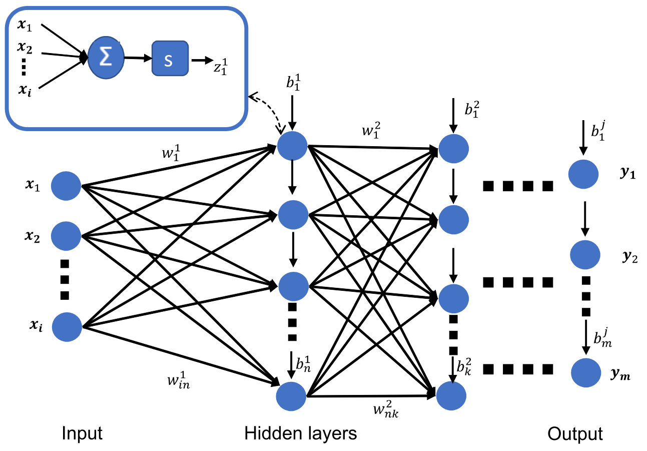


FIGURE 2. Illustrative example of a Feedforward Neural Network.

Keeping this in mind, in the first case study, we considered the problem of generating an optimal metaheuristic algorithm for training an FNN; *i.e.*, for computing the optimal network weights and biases. In this case, the objective function to minimize is the Negative Log-Likelihood function [53], defined as follows,

$$L = -\sum_{i=1}^{n} \left(y_i \log \hat{y}_i + (1 - y_i) \log \left(1 - \hat{y}_i\right)\right). \quad (6)$$

This function computes the error between actual values y_i and the predictions $\hat{y}_i$.

In this case study, we estimated the weights and biases of the neural network architecture using the new metaheuristics generated by the HH framework to obtain repeatable results and good classification accuracy. Firstly, the IRIS dataset containing three classes of IRIS species (*setosa*, *virginica*, and *versicolor*) is loaded. This dataset is quite balanced, containing 50 samples per class. Besides, each sample is represented by a four-feature vector (sepal length, sepal width, petal length, and petal width). Moreover, Figure 3 shows the analyzed neural network architecture, which was composed of an input layer with four neurons, a hidden layer with 20 neurons using the tanh activation function, and an output layer with three neurons employing the softmax function.

The cost function in (6) was limited to using 30 steps in the HH framework as a computing budget, so we obtained different responses for each step performed. Figure 4 shows the first visualization of the preliminary experiments from the HH searching procedure.

The HH procedure generally tends to achieve a metaheuristic solver with great fitness statistics, represented by a lower dispersion and median of the fitness values. This trend is observed in the upper-right chart, where the violin plot shows the fitness output after running 50 times for each MH candidate. In particular, Figure 4 (upper left) depicts how the first MH candidate *evolves* over time, presenting the most significant dispersion and high stagnation. The

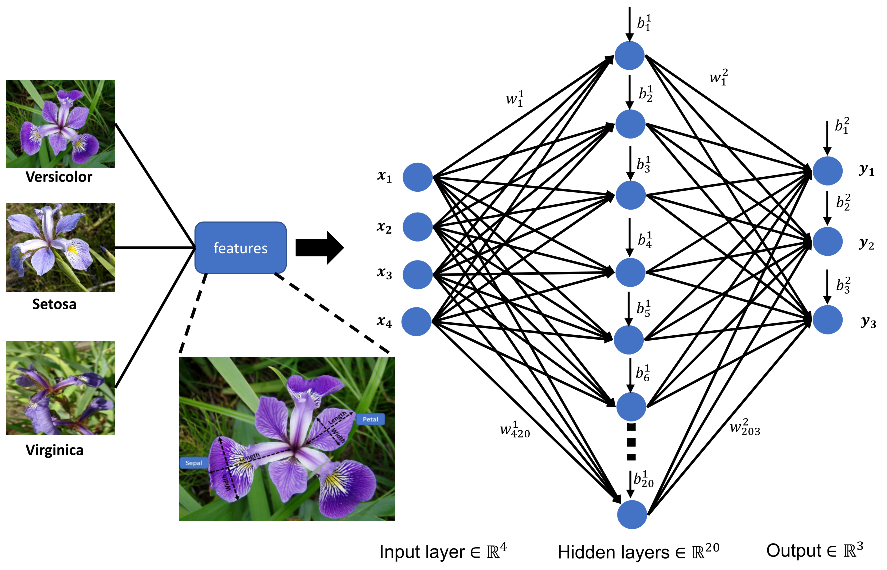


FIGURE 3. Architecture of the Feedforward Neural Network implemented for solving the IRIS classification problem as the practical application of automatic design of metaheuristic algorithms.

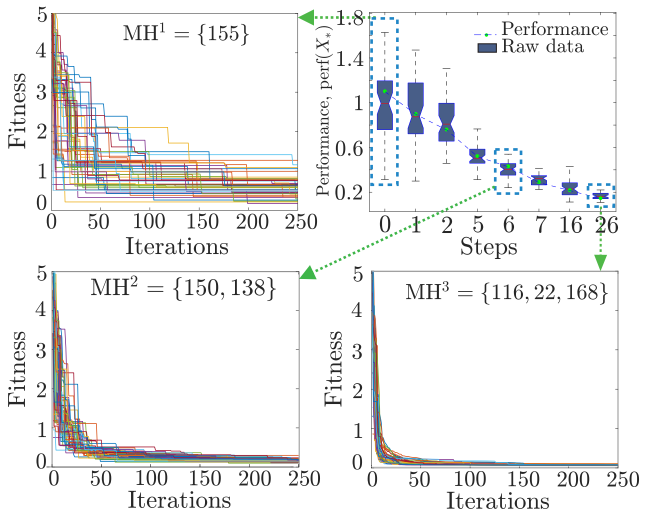


FIGURE 4. Fitness evolution throughout the optimization procedure. The upper-right figure shows the violin plot for each HH step, whereas the others display the MH iterations while tuning the neural network. These MHs correspond to the candidate sequences at the HH steps 0, 6, and 26.

intermediate Step 6, Figure 4 (lower-left), exhibits how the MH candidate improves its solving procedure but cannot reach a satisfactory and convergent fitness value under the imposed number of iterations. After 26 steps, the last MH candidate performance is significantly better, as shown in Figure 4 (lower-right). In this case, each replica in the 26th hyper-heuristic step (each MH run) converged to the performance value of 0.146897252. Recall that this performance metric is associated with the combination of statistics: the sum of the median and the interquartile range of the last fitness values generated in several independent runs on the same problem (*cf.* (4)).

TABLE 3. Statistics in terms of accuracy of the different neural network training tests with the IRIS database performed with the classical MH and the MH_* generated by the HH framework for 50 repetitions.

Method	Mean	Standard Deviation
PSO	0.9592	$\pm 1.810 \times 10^{-2}$
CS	0.8941	$\pm 5.064 \times 10^{-2}$
GSA	0.7376	$\pm 9.718 \times 10^{-2}$
MH_*	0.9932	$\pm 6.663 \times 10^{-4}$

From these results, the best metaheuristic for tuning the studied neural network corresponds to $\{h_{116}, h_{22}, h_{168}\}$ (*cf.* Table 2), where these search operators are:

- h_{116}–*Perturbator*: Genetic Crossover with coefficients equal to 0.5 and a mating pool factor of 0.4; *Selector*: Metropolis.
- h_{22}–*Perturbator*: Differential Mutation with a scale factor of 1.0; *Selector*: Metropolis.
- h_{168}–*Perturbator*: Random Search with a scale factor of 0.01 and a uniform distribution; *Selector*: Metropolis.

Once a candidate MH is generated, we test its performance by carrying out 50 repetitions with a population of 30 agents and up to 250 iterations. Furthermore, three classical MHs, PSO, CS, and GSA, were selected for comparison purposes. For these MHs, the same values of agents and number of iterations were also assigned.

Table 3 displays the statistics for the accuracy achieved from the classical MHs and the one tailored for this case study. In the case of MH_*, an accuracy value of 0.9932 was achieved, surpassing those obtained by PSO, CS, and GSA. In addition, when repeating the experiment, the standard deviation presented a value of $\pm 6.6633 \times 10^{-4}$, indicating a low value of the dispersion of the data. With these results, we can ensure that the generated metaheuristic (MH_*)

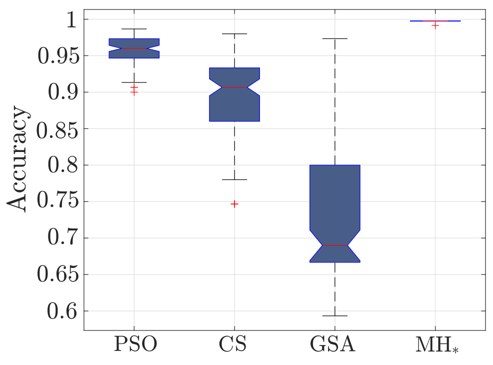


FIGURE 5. Accuracy for the classical metaheuristics (PSO, CS, and GSA) and the tailored MH$_*$ from the hyper-heuristic procedure during the neural network training.

substantially exceeds the classical MH for the case of neural network training.

This degree of repeatability obtained is desired for this type of application. To have a clearer view of this test, we can observe Figure 5. This box plot shows the accuracy of data obtained in the different repetitions. MH$_*$ presents the same accuracy value in almost all the repeats except for one occasion, which gave a value of 0.98. If we look at the classical MHs, we can see good results for CS and PSO with an average accuracy of 0.89 and 0.94, respectively. Although the accuracy values of this MH are good, they do not guarantee that we will obtain the same result when repeating the experiment. Finally, the algorithm that brought the worst results was GSA which presented an accuracy of 0.73, a positive asymmetry of its data, and a high dispersion of values between 0.6 and 0.98. These results verified what we observed in Figure 4, where the fitness function converges to a value close to zero at step 26.

B. DESIGNING CONTROL SYSTEMS

The second practical engineering problem lies in control theory, the cornerstone of almost all modern technologies. Automating industrial processes requires reliable methodologies to control systems to increase production and perform tasks efficiently. The correct implementation of these controllers indeed ensures desired performance for the automated systems. In general, Proportional Integral and Derivative (PID) controllers are the most widely used in the industry, mainly because they can handle systems with various plants with Multiple Inputs and Multiple Outputs (MIMO) [54]. PID controllers frequently work well in relatively simple industrial processes, but problems may appear when handling complex nonlinear systems. For this reason, optimization methods have become significant in control design, allowing the adaptive tuning of control parameters so that the systems respond efficiently to uncertainties [2]. In this context, various Metaheuristics (MHs) have been implemented for tuning control gains to obtain the desired performance. For example, Particle Swarm Optimization [55], Genetic Algorithm [56], and Cuckoo Search [57] have proven to be great alternatives that outperform traditional control design

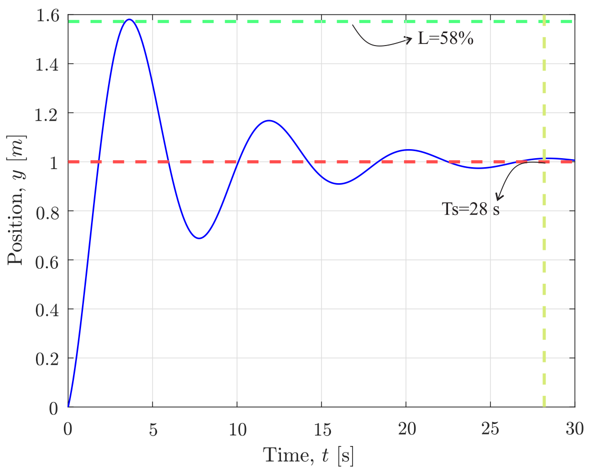


FIGURE 6. Response of a simple uncontrolled Mass-Spring-Damper System to a step input.

methods. Although MHs usually give good results, they only sometimes succeed in fitting every type of controller. In addition, inadequate fitting can cause problems in the analysis of stability and repeatability of the results. Hence, one can use a hyper-heuristic procedure for tailoring the MH that best fits as the problem solver to guarantee the stability and repeatability of the estimated solutions for any well-described model.

In the context of this case study, we considered the tuning problem for controlling a simple mechanical system, such as a Mass-Spring-Damper (MSD) arrangement, which is modeled via a second-order differential equation as shown,

$$m\frac{d^2y}{dt^2} + b\frac{dy}{dt} + ky = f_{\text{ext}}, \tag{7}$$

where m [kg] is the mass, b [kg/s] is the damping coefficient, and k [N/m] is the stiffness. Before continuing, let us consider the numerical solution of an uncontrolled MSD system, with $m = 2$ kg, $b = 0.6$ kg/s, and $k = 1.2$ N/m, while facing an impulsive force $f_{\text{ext}} = \delta(t)$ N, as depicted in Figure 6. We chose these values only for illustrative purposes. To analyze better this resulting signal, we must consider the characteristic features such as the settling time T_s [s], the estimated overshoot at the current controller L_1 [%], the reference overshoot L_0 [%], the relative error between the input and output of the current controller E_{ss} [%], and the relative error between the actual and reference settling time E_{Ts} [%]. The behavior from Figure 6 reveals a high overshoot $L_1 = 58\%$, a long stabilization time $T_s = 28$ s, a steady state error $E_{ss} < 0.1\%$, with a simulation time of at least 40 s.

With this simple simulation, we easily evidence the need to control the dynamic system using a PID controller. So, we implemented a hyper-heuristic approach for finding the optimal metaheuristic algorithm for tuning this controller. Figure 7 presents the overall process we carried out using a signal flow diagram. Using this methodology, we aimed to

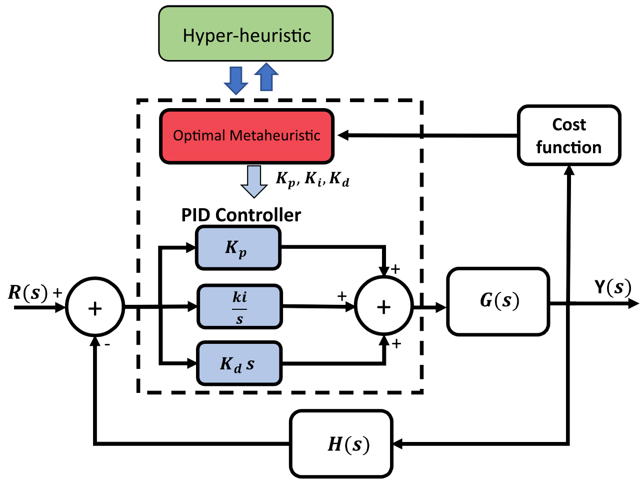


FIGURE 7. Overall hyper-heuristic process implemented for generating a metaheuristic for tuning a PID controller.

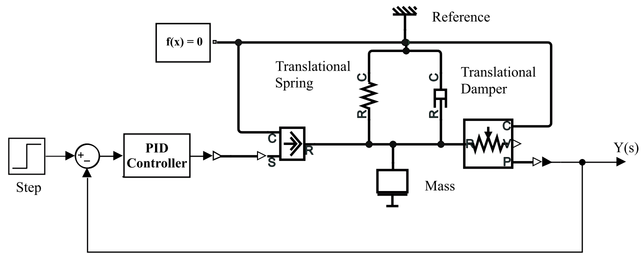


FIGURE 8. Numerical simulation of the damped spring-mass model with a PID controller.

achieve significant efficiency and stability for both the MH and the controller. The gains K_P, K_i, and K_d are determined by the metaheuristic automatically designed by the hyper-heuristic procedure. Each potential solution for this PID controller is simultaneously tested in the dynamic system to obtain the characteristic output features (*i.e.*, L_1, T_s, E_{Ts}, and E_{ss}). To do so, we implemented the following cost function,

$$f(L_1, E_{r1}, E_{r2}) = \alpha\frac{|L_1 - L_o|}{L_o} + \beta|E_{Ts}| + \gamma|E_{ss}|, \qquad (8)$$

since $\alpha, \beta, \gamma \in \mathbb{R}_+$, such that $\alpha + \beta + \gamma = 1$, are regularization parameters to balance the relevance of each output feature error.

The MSD system is regulated by tuning a PID controller to obtain an overshoot of 5%, a settling time of 3 s, and zero steady state error for a Heaviside step input. Thus, the corresponding MSD model was analyzed with Simulink using the diagram shown in Figure 8.

Once the HH procedure has been implemented, we analyze the cost function defined in (8). This will allow us to optimize the PID controller parameters and thus obtain the requested system characteristics (overshoot, settling time, and steady-state error).

We then proceed to find a custom metaheuristic for this problem. Each generated MH uses a population size of 30 and a maximum of 50 iterations. Besides, a maximum of

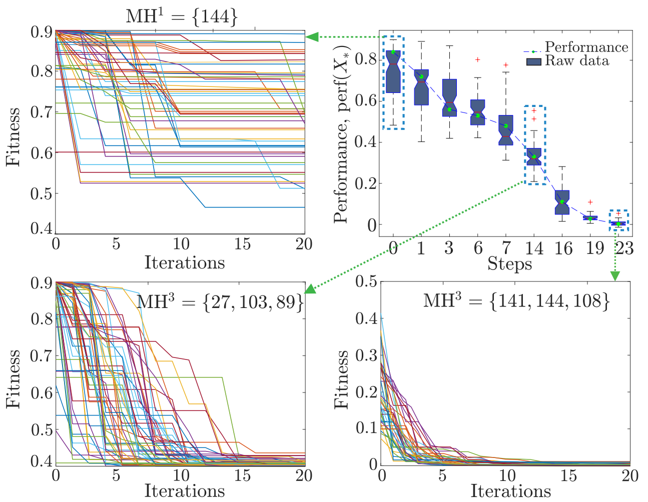


FIGURE 9. Fitness evolution throughout the HH procedure. The upper-right graph shows the violin plot for each HH step, while the others display the MH iterations while tuning the PID controller. These MHs correspond to the candidate sequences at steps 0, 14, and 23, respectively.

30 steps is used during the HH search to achieve the optimal solver. Figure 9 shows a reliable hyper-heuristic tendency to obtain a performing MH solver. First, the upper-right plot of Figure 9 exhibits each step's dispersion profile during the hyper-heuristic search. It is worth noting that the interquartile measure started at step zero with a value of 0.84 and finalized with a remarkable 0.02 at step 23. Figure 9 (upper left) reveals the MH behavior given by the HH framework's first step. As expected, this generated MH presents the most significant dispersion in all the HH evolution. The lower-left plot in the figure displays the MH tailored from step 6, remarking a considerable improvement in its performance. Lastly, Figure 9, lower-right plot, depicts the results evaluation of step 26, corresponding to the tailored MH with the best performance.

Once the steps and replicas in the HH process are completed, the final customized MH for this case study is defined by the sequence $\{h_{141}, h_{144}, h_{108}\}$ (*cf.* Table 2), which is detailed as follows:

- h_{141}–*Perturbator:* Random Flight with a scale factor of 1.0 and a Lévy distribution; *Selector*: Probabilistic.
- h_{144}–*Perturbator:* Random Flight with a scale factor of 1.0 and a Uniform distribution; *Selector*: Metropolis.
- h_{108}–*Perturbator:* Genetic Crossover with coefficients equal to 0.5 and a mating pool factor of 0.4; *Selector*: Metropolis.

Now, concerning the best metaheuristic achieved from this hyper-heuristic search, we test it to verify the performance of the tuned PID controller. Table 4 shows the results in estimating the K_P, K_I, and K_D coefficients of the PID controller. To gather this information, we repeated the experiment 100 times to check the numerical stability and high accuracy of the generated MH. From that, we noticed a very low standard deviation. This behavior guarantees a high

TABLE 4. Statistical analysis for PID Control Parameters obtained by implementing the tailored metaheuristic.

Gain	Mean	Standard Deviation
K_P	20.2519315	$\pm 3.36 \times 10^{-3}$
K_I	10.0363149	$\pm 2.90 \times 10^{-2}$
K_D	18.5300373	$\pm 4.25 \times 10^{-2}$

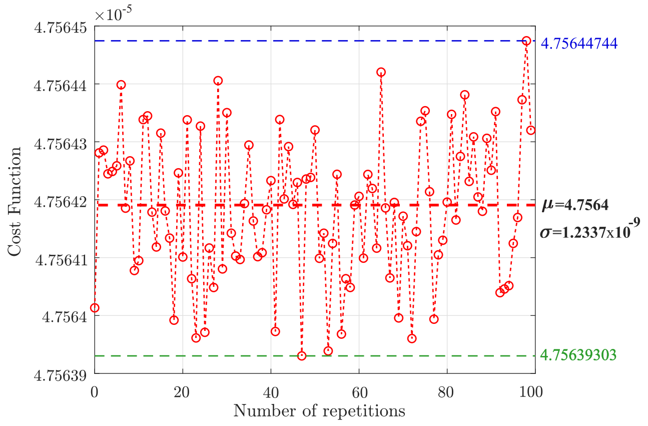


FIGURE 10. Cost function profile versus the number of repetitions. The blue dotted line represents the maximum obtained value, while the green line is the smallest value. The red dotted line represents the average of all experiments.

degree of repeatability, which is sought after when tuning this type of controller. In addition, Figure 10 shows the statistical information related to the cost function values. From this information, we notice a low dispersion in all the replicas carried out, *i.e.*, a mean value of 4.7564×10^{-5} and a standard deviation of 1.2337×10^{-9}.

Next, the designed PID controller is applied to the Mass-Spring-Damper system to verify that the controlled response satisfies the requirements. Plus, we implemented PSO and CS to solve the design problem of this case study, employing the same specifications of iterations and population as in the tailored metaheuristic MH$_*$. Figure 11 illustrates the curves corresponding to the behavior achieved by implementing the PID controllers obtained from each metaheuristic algorithm. In this figure, the horizontal red dotted line limits the observed overshoot of the controlled system, reaching a value of 4.9991%. Note that the vertical red dotted line corresponds to the settling time with a value of 2.996 s, representing a zero steady-state error. Moreover, Figure 11 shows a zoomed oval at the results precisely in the overshoot. Here, we quickly observe that the controllers generated by PSO and CS achieve overshoots of 5.1352% and 5.145%, respectively. Although the results obtained by PSO and CS were good, the tailored metaheuristic MH$_*$ presents higher precision and accuracy than the classical MHs.

C. MODELING FRACTIONAL THERMAL SYSTEMS

This work's last but not least case study comprises the fractional-based modeling of a non-conventional calorimetric process for electronic thermal management applications.

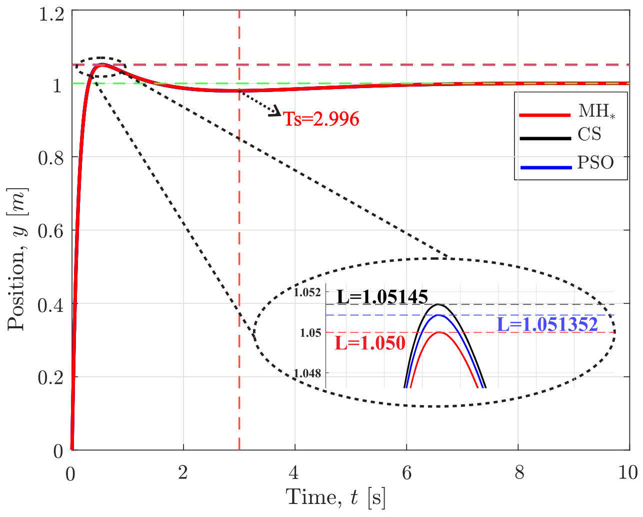


FIGURE 11. Transient profile of the system controlled by a PID controller tuned with classical metaheuristics (PSO, CS) and the tailored metaheuristic MH$_*$.

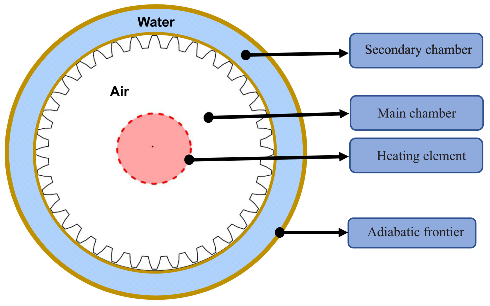


FIGURE 12. Schematic of the non-conventional calorimeter for assessing the thermal heat power that a microelectronic device produces.

We used the ordinary model and the prototype device of this non-conventional calorimeter originally reported in [58]. Figure 12 shows a schematic of the non-conventional calorimeter, and the model corresponding to the temperature of the working fluid is given by

$$\frac{d\theta}{dt} + \frac{\theta}{RC} = \frac{\dot{Q}}{C}, \tag{9}$$

where θ [°C] is the temperature, R [K/W] is the equivalent thermal resistance, C [J/K] is the heat capacity, and $\dot{Q}$ [W] is the heat power generated by the Device Under Test (DUT). Regard that a DUT can be a prototype electronic device or a simple breadboard circuit, among others.

It is easy to notice that the model in (9) is an idealistic approach for a practical device with a multi-physics behaviour. For this reason, Cruz-Duarte et al. developed a sophisticated signal processing algorithm to effectively measure entering the working fluid [58]. Therefore, we propose a different alternative to model this process, starting from this simple model but taking advantage of fractional calculus. This area has proven its strength in describing practical and noisy engineering scenarios [59], [60].

Fractional Calculus (FC) is considered a generalization of the well-known integer-order calculus because it comprises derivatives and integrals with non-integer orders. Indeed, these operators go beyond orders with real numbers, such as complex quantities [61]. As we mentioned before, this tool has helped model sophisticated phenomena or systems, for example, those related to thermal processes [62] and ultracapacitor discharging cycles [63]. Different fractional-based operators have been proposed in the literature. Some of the most popular ones are Caputo-Dzhrbashyan [64], Riemann-Liouville [65], and Grünwald-Letnikov [66]. Each of these operators has different mathematical properties, which one needs to examine and adjust to align with the problem under analysis.

In this work, we used the Caputo-Dzhrbashyan derivative [67], a variant of the Riemann-Liouville operator. This operator is formally defined by

$$ {}^{\mathrm{CD}}_{\;\;0}D^{\alpha}_{t}\, z(t) = \frac{1}{\Gamma(1-\alpha)} \int_{0}^{t} (t-\tau)^{-\alpha} \left.\frac{dz}{dt}\right|_{t=\tau} d\tau, \qquad (10) $$

where $z(t) : \mathbb{R} \to \mathbb{R}$ is an arbitrary real function and $\alpha \in [0, 1]$ is the fractional-order. Expression (10) is used because it adds a versatile transformation kernel, widening the number of applications involving fractional-order differential equations [68].

Keeping the simple model using heat transfer concepts and traditional calculus and the Caputo-Dzhrbashyan derivative, we can easily restate the equivalent fractional-order model such as,

$$ {}^{\mathrm{CD}}_{\;\;0}D^{\alpha}_{t}\,\theta + \frac{\theta}{RC} = \frac{\dot{Q}}{C}, \quad \theta(0^{+}) = \theta_0. \qquad (11) $$

To find the solution for this fractional differential equation, we utilized the Mittag-Leffler function, which is considered as an natural extension to the Euler function [69]. The resulting expression, the fractional-order model, is given by

$$ \theta_T(t) = \theta_0 + R\dot{Q}t^{\alpha}\mathrm{E}_{\alpha,\alpha+1}\left[-\left(\frac{t}{RC}\right)^{\alpha}\right], \qquad (12) $$

since the relative temperature $\theta(t) = \theta_T(t) - \theta_0$ is included to consider variations at room temperature θ_0, this leads to real temperature $\theta_T(t)$ estimation.

Lastly, the fractional model in (12) describes the temperature behavior of the working fluid contained in the calorimeter's reservoir during a microelectronic heat power estimation analysis. The problem in this case study is finding the optimal value for the fractional order α to fit the experimental temperature data better and, naturally, describe the calorimeter plant. This is possible via the so-called least squares problem, such as

$$ \vec{x}_{*} = \underset{\vec{x}}{\mathrm{argmin}} \left\{ \sum_{i=1}^{m} \left[\hat{\theta}_i - \theta_T(\vec{x}, \hat{t}_i) \right]^2 \right\}, \qquad (13) $$

where $\vec{x}_{*}$ stands for the design vector comprising the parameters that describe the calorimeter's model. In our particular case, $\vec{x} = (\theta_o, \dot{Q}, R, C, \alpha)^{\intercal}$. Moreover, $\hat{\Theta} \ni \hat{\theta}_i$ is

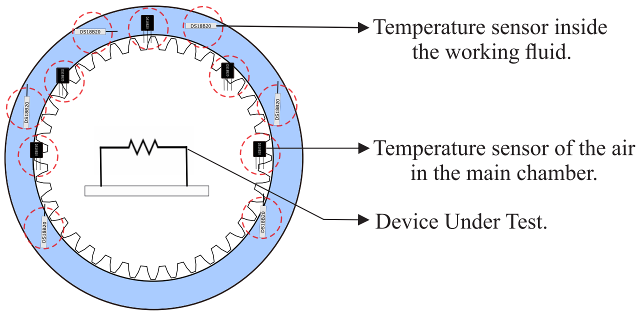

Temperature sensor inside the working fluid.
Temperature sensor of the air in the main chamber.
Device Under Test.
DS18B20

FIGURE 13. Distribution of temperature sensors used by the non-conventional calorimeter to measure the temperature inside the working fluid and the air in the main chamber.

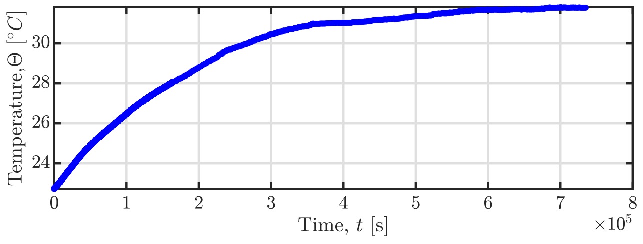


FIGURE 14. Average temperature behavior recorded by sensors inside the working fluid.

the dataset with experimental measurements of the working fluid temperature during the analysis of a particular DUT; $\hat{T} \ni \hat{t}_i$ corresponds to the temporal stamps associated with temperatures $\hat{\Theta}$. These data were achieved using the prototype presented in [58] and the distribution for the temperature sensors depicted in Figure 13. Plus, Figure 14 shows an illustrative example of the raw temperature data captured from a 150 Ω resistor with 10 W power and energized with a 45 V voltage source as the DUT.

It is well-known that the problem stated in (13) is a non-linear regression easy to solve via a soft-computing methodology such as a metaheuristic (MH). The challenge was finding the proper method to solve the low-level problem, which was possible by employing the methodology described in the previous section. To solve the fitting problem, we defined a population of 40 agents performing up to 100 iterations and using up to three search operations per iteration. Figure 15 shows the HH search in tailoring the optimal MH (the upper-right plot), and the other three curves detail particular HH steps from the perspective of the metaheuristic evolution. Likewise the previous case studies, this HH procedure follows the same pattern toward minimizing the mean squared error. As one may notice, the HH evolution is faster in this practical application than in other case studies. For example, we rapidly observe that 30 iterations are enough for reaching a good fitness value in the 10th step of the hyper-heuristic search; *cf.* the lower-left plot in Figure 15. Finally, this process obtained the best results at the 17th step, where the dispersion of the fitness values (mean squared error) is substantially

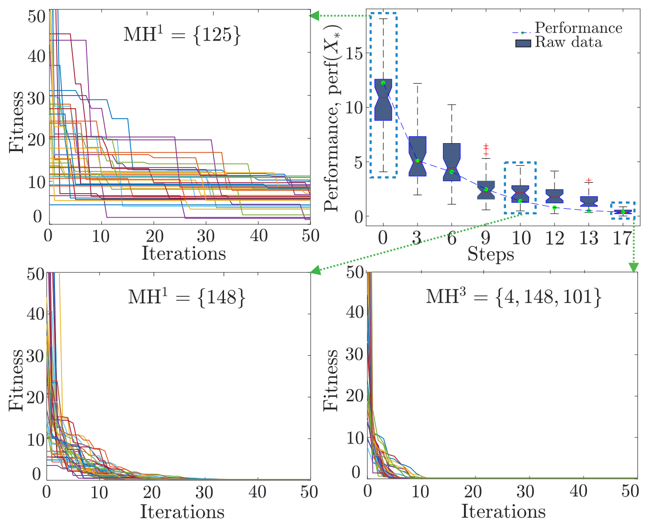


FIGURE 15. Evolution of the mean squared error values when finding an optimal metaheuristic that fits the fractional-based calorimetric model. The upper-right plot stands for the HH perspective, and the others for the MH one.

TABLE 5. Estimated parameters of the fractional-based calorimetric model using the optimal metaheuristic automatically tailored by the hyper-heuristic methodology.

Par. [Unit]	Best	Mean	St. Deviation
C [J/K]	1.51160×10^5	1.5309×10^5	$\pm1.7551\times10^{-2}$
R [K/W]	0.12893	0.12824	$\pm2.1328\times10^{-3}$
θ_0 [°C]	2.27200	2.26583	$\pm1.1335\times10^{-2}$
$\dot{Q}$ [W]	0.73035	0.72792	$\pm2.8028\times10^{-3}$
α	0.99349	0.99341	$\pm3.2302\times10^{-3}$

smaller when compared to the initial candidate metaheuristic. Quantitatively speaking, the interquartile range of the fitness values started at 12.05 in the first HH step and ended at 0.12, with the last step corresponding to the optimal MH for this problem.

Once the steps and replicas in the HH process are completed, the final generated MH is defined by $\{h_4, h_{148}, h_{101}\}$ (*cf.* Table 2), corresponding to the following search operators:

- h_4–*Perturbator*: Central Force Dynamic as the perturbator; *Selector*: Metropolis.
- h_{148}–*Perturbator*: Random Flight with a scale factor of 1.0 and with Gaussian distribution as the perturbator; *Selector*: Direct.
- h_{101}–*Perturbator*: Genetic Crossover with coefficients equal to 0.5 and a mating pool factor of 0.4; *Selector*: Metropolis.

The experiment mentioned above was repeated 50 times to guarantee statistical robustness, so we estimated the parameters of the fractional-based calorimetric model summarized in Table 5. For each model's parameter, we present the best, mean, and standard deviation values since the best column corresponds to that configuration rendering the minimal MSE of 0.1903.

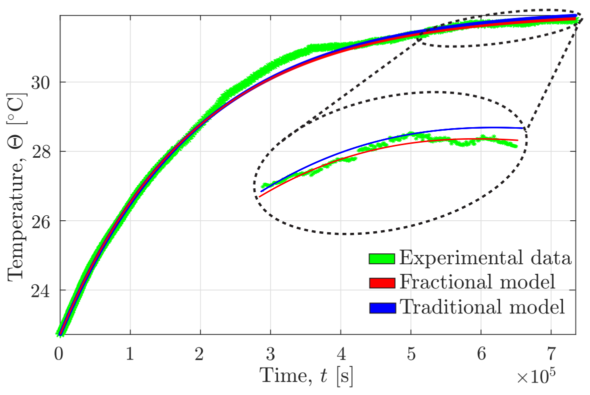


FIGURE 16. Comparison of fractional and traditional models against the experimental data.

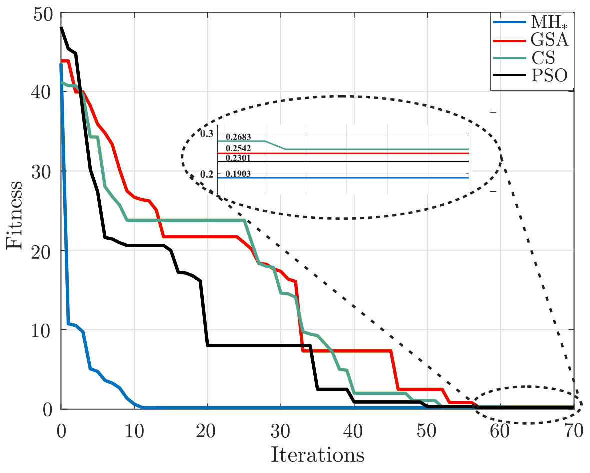


FIGURE 17. Performance evaluation of some classical metaheuristics (such as PSO, CS, and GSA) and the tailored MH_* via the hyper-heuristic procedure. All the metaheuristics were employed for estimating the parameters of the fractional-based calorimetric model.

Furthermore, Figure 16 displays the comparison between the fractional model achieved by the generated metaheuristic and the traditional model achieved via the Levenberg-Marquart optimization method. We also show the experimental data for contrasting these behaviors. We obtained the traditional model by solving a first-order differential equation, whose solution is a trivial exponential function. Figure 16 shows a close-up of the final part of the graph, where the fractional model can fit the experimental data more accurately.

Lastly, as we did in the previous case studies, we implemented three well-known metaheuristics from the literature to compare their performance against the designed algorithm for this problem. The new generated MHs showed a better performance regarding the number of iterations and minimum error, as shown in Figure 17 (blue line). Also, we detected no stagnation due to the stoppage of this MH. It is noteworthy that the PSO, CS, and GSA were tuned and conditioned according to the literature. Besides, the population size was increased from 30 individuals to 60 for the classical MHs to compare the algorithms satisfactorily. The minimal fitness achieved by the generated MH was 0.1903, while PSO, GSA, and CS reached 0.2301, 0.2542, and 0.2683.

V. CONCLUSION

This paper proposed a novel and practical methodology to automatically design Metaheuristics (MHs) through a Hyper-Heuristic (HH) procedure for solving practical optimization engineering problems. The HH process is powered by the so-called Simulated Annealing algorithm, which searches within the heuristic space for a sequence that, at a lower level, explores the continuous problem domain to find the optimal solution. Therefore, we provided the practitioners with an automatic methodology for finding MHs to deal with real optimization problems. So, due to the nature of these algorithms, one can achieve excellent performance when using an MH tailored for a particular scenario on a similar problem. Even if the solution goodness is limited, such an MH serves as a seed algorithm for further adjustments. The principal innovation of our proposal is that the users require a minimum level of expertise in metaheuristics to find one suitable for their needs.

To illustrate the advantages of the proposed automatic design methodology, we selected three case studies from different fields. Such problems are the training of neural networks for image classification, the design of PID controllers for mechanical systems, and the modeling of a non-conventional calorimetric system via fractional calculus. We proved that the hyper-heuristic methodology is a reliable alternative for designing population-based metaheuristics that solve continuous engineering problems and exhibit a low computational cost. These optimal metaheuristics can be used in similar applications to those they were designed for or be taken as the initial seeds for more sophisticated techniques.

It is evident that this is only an *hors d'oeuvre* to several astonishing research and innovative applications. So, we expect to expand upon the number of real engineering applications and verify the standardization of the generated algorithms for similar problems. It is necessary to have real engineering problems that serve as benchmark test functions for these generated metaheuristics. In particular, we plan to work on problems related to renewable energy sources, representing an excellent research field with lots of potential for noteworthy contributions. We are moving ahead to study a more autonomous way to explore the heuristic space, an extended heuristic space, which includes the simple heuristics and their tuning parameters. For that purpose, a solid theoretical basis is required.

CONFLICTS OF INTEREST

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

REFERENCES

[1] N. F. Alkayem, M. Cao, L. Shen, R. Fu, and D. Sumarac, "The combined social engineering particle swarm optimization for real-world engineering problems: A case study of model-based structural health monitoring," *Appl. Soft Comput.*, vol. 123, Jul. 2022, Art. no. 108919.

[2] S. B. Joseph, E. G. Dada, A. Abidemi, D. O. Oyewola, and B. M. Khammas, "Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems," *Heliyon*, vol. 8, no. 5, May 2022, Art. no. e09399.

[3] S. Nematzadeh, F. Kiani, M. Torkamanian-Afshar, and N. Aydin, "Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases," *Comput. Biol. Chem.*, vol. 97, Apr. 2022, Art. no. 107619.

[4] K. Hussain, W. Zhu, and M. N. Mohd Salleh, "Long-term memory Harris' hawk optimization for high dimensional and optimal power flow problems," *IEEE Access*, vol. 7, pp. 147596–147616, 2019.

[5] S. Bashath and A. R. Ismail, "Comparison of swarm intelligence algorithms for high dimensional optimization problems," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 11, no. 1, pp. 300–307, 2018.

[6] S. Agarwal, A. P. Singh, and N. Anand, "Evaluation performance study of firefly algorithm, particle swarm optimization and artificial bee colony algorithm for non-linear mathematical optimization functions," in *Proc. 4th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2013, pp. 1–8.

[7] S. Salcedo-Sanz, "Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures," *Phys. Rep.*, vol. 655, pp. 1–70, Oct. 2016.

[8] S. S. Rao, *Engineering Optimization: Theory and Practice*. Hoboken, NJ, USA: Wiley, 2019.

[9] S. P. Adam, S.-A. N. Alexandropoulos, P. M. Pardalos, and M. N. Vrahatis, "No free lunch theorem: A review," in *Approximation and Optimization*. Cham, Switzerland: Springer, 2019, pp. 57–82.

[10] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Comput. Ind. Eng.*, vol. 137, Nov. 2019, Art. no. 106040.

[11] L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated annealing algorithm for deep learning," *Proc. Comput. Sci.*, vol. 72, pp. 137–144, Jan. 2015.

[12] Y. Xue, Y. Tong, and F. Neri, "An ensemble of differential evolution and Adam for training feed-forward neural networks," *Inf. Sci.*, vol. 608, pp. 453–471, Aug. 2022.

[13] S. Sharma and V. Kumar, "Application of genetic algorithms in healthcare: A review," in *Next Generation Healthcare Informatics* (Studies in Computational Intelligence), B. K. Tripathy, P. Lingras, A. K. Kar, and C. L. Chowdhary, Eds. Singapore: Springer, 2022, pp. 75–86.

[14] V. Kumar, I. Sharma, and S. Sharma, "A comprehensive survey on grey wolf optimization," *Recent Adv. Comput. Sci. Commun.*, vol. 15, no. 3, pp. 323–333, Mar. 2022.

[15] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, vol. 10, pp. 10031–10061, 2022.

[16] J. D. Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. C. Coello, and F. Herrera, "Bio-inspired computation: Where we stand and what's next," *Swarm Evol. Comput.*, vol. 48, pp. 220–250, Aug. 2019.

[17] A. C. B. Monteiro, R. P. Franca, R. Arthur, and Y. Iano, "The fundamentals and potential of heuristics and metaheuristics for multiobjective combinatorial optimization problems and solution methods," in *Multi-Objective Combinatorial Optimization Problems and Solution Methods*, M. Toloo, S. Talatahari, and I. Rahimi, Eds. Cambridge, MA, USA: Academic Press, 2022, pp. 9–29.

[18] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: Revisited," in *Handbook of Metaheuristics* (International Series in Operations Research and Management Science), M. Gendreau and J.-Y. Potvin, Eds. Cham, Switzerland: Springer, 2019, pp. 453–477.

[19] N. Pillay and R. Qu, *Hyper-Heuristics: Theory and Applications*. Berlin, Germany: Springer, 2018.

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

[21] 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.

[22] R. Bai, J. Blazewicz, E. K. Burke, G. Kendall, and B. McCollum, "A simulated annealing hyper-heuristic methodology for flexible decision support," *4OR*, vol. 10, no. 1, pp. 43–66, Mar. 2012.

[23] 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.

[24] M. Kalender, A. Kheiri, E. Ozcan, and E. K. Burke, "A greedy gradient-simulated annealing hyper-heuristic for a curriculum-based course timetabling problem," in *Proc. 12th U. K. Workshop Comput. Intell. (UKCI)*, Sep. 2012, pp. 1–8.

[25] H. M. Kartika and M. Ahmad, "Self adaptive and simulated annealing hyper-heuristics approach for post-enrollment course timetabling," *J. Phys., Conf.*, vol. 1577, no. 1, Jul. 2020, Art. no. 012033.

[26] K. Czerniachowska and M. Hernes, "Simulated annealing hyper-heuristic for a shelf space allocation on symmetrical planograms problem," *Symmetry*, vol. 13, no. 7, p. 1182, Jun. 2021.

[27] H. Mosadegh, S. M. T. Fatemi Ghomi, and G. A. Suer, "Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and Q-learning based simulated annealing hyper-heuristics," *Eur. J. Oper. Res.*, vol. 282, no. 2, pp. 530–544, Apr. 2020.

[28] J. M. Cruz-Duarte, J. C. Ortiz-Bayliss, I. Amaya, and N. Pillay, "Global optimisation through hyper-heuristics: Unfolding population-based metaheuristics," *Appl. Sci.*, vol. 11, no. 12, p. 5620, Jun. 2021.

[29] J. M. Cruz-Duarte, I. Amaya, J. C. Ortiz-Bayliss, S. E. Conant-Pablos, H. Terashima-Marín, and Y. Shi, "Hyper-heuristics to customise metaheuristics for continuous optimisation," *Swarm Evol. Comput.*, vol. 66, Oct. 2021, Art. no. 100935.

[30] J. M. Cruz-Duarte, I. Amaya, J. C. Ortiz-Bayliss, H. Terashima-Marín, and Y. Shi, "CUSTOMHyS: Customising optimisation metaheuristics via hyper-heuristic search," *SoftwareX*, vol. 12, Jul. 2020, Art. no. 100628.

[31] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics* (International Series in Operations Research & Management Science), vol. 272. Cham, Switzerland: Springer, 2019.

[32] J. M. Cruz-Duarte, J. C. Ortiz-Bayliss, I. Amaya, Y. Shi, H. Terashima-Marin, and N. Pillay, "Towards a generalised metaheuristic model for continuous optimisation problems," *Mathematics*, vol. 8, no. 11, pp. 1–23, Nov. 2020.

[33] G. Woumans, L. De Boeck, J. Belien, and S. Creemers, "A column generation approach for solving the examination-timetabling problem," *Eur. J. Oper. Res.*, vol. 253, no. 1, pp. 178–194, Aug. 2016.

[34] S. M. Almufti, "Historical survey on metaheuristics algorithms," *Int. J. Sci. World*, vol. 7, no. 1, p. 1, Nov. 2019.

[35] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: A comprehensive survey," *Artif. Intell. Rev.*, vol. 52, no. 4, pp. 2191–2233, Dec. 2019.

[36] 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.

[37] J. M. Cruz-Duarte, I. Martin-Diaz, J. U. Munoz-Minjares, L. A. Sanchez-Galindo, J. G. Avina-Cervantes, A. Garcia-Perez, and C. R. Correa-Cely, "Primary study on the stochastic spiral optimization algorithm," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Nov. 2017, pp. 1–6.

[38] A. K. Kar, "Bio inspired computing—A review of algorithms and scope of applications," *Exp. Syst. Appl.*, vol. 59, pp. 20–32, Oct. 2016.

[39] H. Feng, W. Ma, C. Yin, and D. Cao, "Trajectory control of electro-hydraulic position servo system using improved PSO-PID controller," *Autom. Construct.*, vol. 127, Jul. 2021, Art. no. 103722.

[40] C. Chiñas-Palacios, C. Vargas-Salgado, J. Aguila-Leon, and E. Hurtado-Pérez, "A cascade hybrid PSO feed-forward neural network model of a biomass gasification plant for covering the energy demand in an AC microgrid," *Energy Convers. Manag.*, vol. 232, Mar. 2021, Art. no. 113896.

[41] A. Dabiri, B. P. Moghaddam, and J. A. T. Machado, "Optimal variable-order fractional PID controllers for dynamical systems," *J. Comput. Appl. Math.*, vol. 339, pp. 40–48, Sep. 2018.

[42] S. Marso and M. El Merouani, "Predicting financial distress using hybrid feedforward neural network with cuckoo search algorithm," *Proc. Comput. Sci.*, vol. 170, pp. 1134–1140, Jan. 2020.

[43] O. Karahan, "Design of optimal fractional order fuzzy PID controller based on cuckoo search algorithm for core power control in molten salt reactors," *Prog. Nucl. Energy*, vol. 139, Sep. 2021, Art. no. 103868.

[44] D. Yousri and S. Mirjalili, "Fractional-order cuckoo search algorithm for parameter identification of the fractional-order chaotic, chaotic with noise and hyper-chaotic financial systems," *Eng. Appl. Artif. Intell.*, vol. 92, Jun. 2020, Art. no. 103662.

[45] Y. Muhammad, M. A. Z. Raja, M. A. A. Shah, S. E. Awan, F. Ullah, N. I. Chaudhary, K. M. Cheema, A. H. Milyani, and C.-M. Shu, "Optimal coordination of directional overcurrent relays using hybrid fractional computing with gravitational search strategy," *Energy Rep.*, vol. 7, pp. 7504–7519, Nov. 2021.

[46] F. Olivas, F. Valdez, P. Melin, A. Sombra, and O. Castillo, "Interval type-2 fuzzy logic for dynamic parameter adaptation in a modified gravitational search algorithm," *Inf. Sci.*, vol. 476, pp. 159–175, Feb. 2019.

[47] R. García-Ródenas, L. J. Linares, and J. A. López-Gómez, "Memetic algorithms for training feedforward neural networks: An approach based on gravitational search algorithm," *Neural Comput. Appl.*, vol. 33, no. 7, pp. 2561–2588, Apr. 2021.

[48] P. Baldi and R. Vershynin, "The capacity of feedforward neural networks," *Neural Netw.*, vol. 116, pp. 288–311, Aug. 2019.

[49] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, 1997.

[50] M. H. Sazli, "A brief review of feed-forward neural networks," *Commun., Fac. Sci., Univ. Ankara*, vol. 50, pp. 11–17, Jan. 2006.

[51] M. Senthilkumar, "Use of artificial neural networks (ANNs) in colour measurement," in *Colour Measurement*, M. Gulrajani, Ed. Oxford, U.K.: Woodhead, 2010, pp. 125–146. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781845695590500157

[52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[53] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Statist. Planning Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[54] T. Hagiwara, K. Yamada, S. Matsuura, and S. Aoyama, "A design method for modified PID controllers for multiple-input/multiple-output plants," *J. Syst. Des. Dyn.*, vol. 6, no. 2, pp. 131–144, 2012.

[55] S. Ahmadi, S. Abdi, and M. Kakavand, "Maximum power point tracking of a proton exchange membrane fuel cell system using PSO-PID controller," *Int. J. Hydrogen Energy*, vol. 42, no. 32, pp. 20430–20443, Aug. 2017.

[56] S. M. H. Mousakazemi, "Comparison of the error-integral performance indexes in a GA-tuned PID controlling system of a PWR-type nuclear reactor point-kinetics model," *Prog. Nucl. Energy*, vol. 132, Feb. 2021, Art. no. 103604.

[57] Z. Bingul and O. Karahan, "A novel performance criterion approach to optimum design of PID controller using cuckoo search algorithm for AVR system," *J. Franklin Inst.*, vol. 355, no. 13, pp. 5534–5559, 2018.

[58] J. M. Cruz-Duarte, J. G. Avina-Cervantes, A. Garcia-Perez, J. A. Andrade-Lucio, R. Correa, and A. M. Morega, "Novel calorimetric approach for thermal analysis of microelectronic devices," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 11, no. 9, pp. 1442–1451, Sep. 2021.

[59] J. M. Cruz-Duarte, M. Guía-Calderón, J. J. Rosales-García, and R. Correa, "Determination of a physically correct fractional-order model for electrolytic computer-grade capacitors," *Math. Methods Appl. Sci.*, vol. 44, no. 6, pp. 4366–4380, Apr. 2021.

[60] L. M. Jiménez, J. M. Cruz-Duarte, J. E. Escalante-Martínez, and J. J. Rosales-García, "Analytical and experimental study for mechanical vibrations of a two-coupled spring masses system via Caputo-based derivatives," *Math. Methods Appl. Sci.*, pp. 1–19, Apr. 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/mma.7421

[61] M. D. Ortigueira, *Fractional Calculus for Scientists and Engineers*. Berlin, Germany: Springer, 2011, vol. 84, pp. 5–41. [Online]. Available: https://www.springer.com/series/7818 and https://link.springer.com/book/10.1007/978-94-007-0747-4

[62] A. Dzielinski and D. Sierociuk, *Fractional Order Model of Beam Heating Process and Its Experimental Verification*. Dordrecht, The Netherlands: Springer, 2010, pp. 287–294.

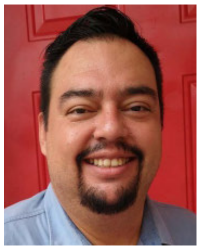
[63] J. T. Machado, M. F. Silva, R. S. Barbosa, I. S. Jesus, C. M. Reis, M. G. Marcos, and A. F. Galhano, "Some applications of fractional calculus in engineering," *Math. Problems Eng.*, Jan. 2010.

[64] A. B. Malinowska and D. F. M. Torres, "Fractional calculus of variations for a combined caputo derivative," *Fractional Calculus Appl. Anal.*, vol. 14, no. 4, pp. 523–537, Dec. 2011.
[65] T. F. Nonnenmacher and R. Metzler, "On the Riemann–Liouville fractional calculus and some recent applications," *Fractals*, vol. 3, no. 3, pp. 557–566, Sep. 1995.
[66] H. Jalalinejad, A. Tavakoli, and F. Zarmehi, "A simple and flexible modification of Grünwald–Letnikov fractional derivative in image processing," *Math. Sci.*, vol. 12, no. 3, pp. 205–210, Sep. 2018.
[67] A. N. Kochubei, "Fractional-hyperbolic systems," *Fractional Calculus Appl. Anal.*, vol. 16, no. 4, pp. 860–873, Dec. 2013.
[68] M. Caputo, "Linear model of dissipation whose Q is almost frequency independent," *Geophys. J. Roy. Astronomical Soc.*, vol. 13, pp. 529–539, Nov. 1967.
[69] A. A. Kilbas, M. Saigo, and R. K. Saxena, "Generalized Mittag-Leffler function and generalized fractional calculus operators," *Integr. Transf. Special Functions*, vol. 15, no. 1, pp. 31–49, 2004.

DANIEL F. ZAMBRANO-GUTIERREZ received the B.Eng. degree in electronics from the Universidad Nacional Experimental del Táchira, San Cristóbal, Venezuela, in 2016, and the M.Eng. degree in electrical engineering (instrumentation and digital systems) from the Universidad de Guanajuato, Mexico, in 2021. He is currently pursuing the Ph.D. degree in computer science with the Tecnólogico de Monterrey. His research interests include control systems, bio-inspired optimization methods, computer vision, fractional calculus, pattern recognition, and image processing.

JORGE MARIO CRUZ-DUARTE (Member, IEEE) was born in Ocaña, Norte de Santander, Colombia, in 1990. He received the B.Sc. and M.Sc. degrees in electronics engineering from the Universidad Industrial de Santander (UIS), Bucaramanga, Santander, Colombia, in 2012 and 2015, respectively, and the Ph.D. degree in electrical engineering from the Universidad de Guanajuato (UGTO), Guanajuato, Mexico, in 2018. From 2019 to 2021, he was on a postdoctoral stay at the Research Group with Strategic Focus on Intelligent Systems at the Tecnológico de Monterrey (TEC) in collaboration with the Chinese Academy of Sciences (CAS). Since 2021, he has been a Researcher Professor with the Research Group on Advanced Artificial Intelligence, TEC, and a member of the Mexican National System of Researchers (SNI-CONACyT) and AMEXCOMP. His research interests include automatic design, heuristics, fractional calculus, applied thermodynamics, data science, and artificial intelligence.

JUAN GABRIEL AVINA-CERVANTES received the B.S. degree in communications and electronics engineering and the master's degree in electrical engineering (instrumentation and digital systems) from the University of Guanajuato, in 1998 and 1999, respectively, and the joint Ph.D. degree in informatics and telecommunications from the Institut National Polytechnique de Toulouse and LAAS-CNRS, France, in 2005. Currently, he is a Researcher and a full-time Professor at the University of Guanajuato. His research interests include the artificial vision for outdoor mobile robotics, pattern recognition, control systems, optimization, and image processing.



JOSÉ CARLOS ORTIZ-BAYLISS (Member, IEEE) was born in Culiacan, Sinaloa, Mexico, in 1981. He received the B.Sc. degree in computer engineering from the Universidad Tecnologica de la Mixteca, in 2005, the M.Sc. degree in computer science and the Ph.D. degree from the Tecnologico de Monterrey, in 2008 and 2011, respectively, the M.Ed. degree from the Universidad del Valle de Mexico, in 2017, the B.Sc. degree in project management from the Universidad Virtual del Estado de Guanajuato, in 2019, and the M.Ed.A. degree from the Instituto de Estudios Universitarios, in 2019. He is currently an Assistant Research Professor with the School of Engineering and Sciences, Tecnologico 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.

JESÚS JOAQUÍN YANEZ-BORJAS received the B.Sc. and M.Sc. degrees in electronics engineering from the Universidad Nacional Experimental del Táchira, San Cristóbal, Venezuela, in 2005 and 2013, respectively, and the M.Eng. degree in instrumentation and digital systems and the Ph.D. degree in electrical engineering from the Universidad de Guanajuato, Mexico, in 2017 and 2021, respectively. He currently belongs to the Telematics Group, Electronics Engineering Department, University of Guanajuato. His research interests include, condition and structural health monitoring, data science, digital signal and image processing, and pattern recognition.

IVÁN AMAYA (Senior 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, Tecnologico de Monterrey (TEC). Since then, he has been a Research Professor with the School of Engineering and Sciences, TEC. His research interests include numerical optimization of both, continuous and discrete problems, through the application of heuristics, metaheuristics, hyper-heuristics, and finding new ways of using feature transformations for improving hyper-heuristic performance. He is a member of the Mexican National System of Researchers, the Mexican Academy of Computing, and the Association for Computing Machinery.