A Primary Study on Hyper-Heuristics to Customise Metaheuristics for Continuous Optimisation

Jorge M. Cruz-Duarte*, Ivan Amaya*, José Carlos Ortiz-Bayliss*,
Santiago Enrique Conant-Pablos*, and Hugo Terashima-Marín*
* Tenologico de Monterrey, School of Engineering and Science
E-mails: {jorge.cruz, iamaya2, jcobayliss, sconant, terashima}@tec.mx

***Abstract*—Literature is prolific with metaheuristics for solving continuous optimisation problems. But, in practice, it is difficult to choose one appropriately. Moreover, it is necessary to determine a good enough set of parameters for the selected approach. Hence, this work proposes a strategy based on a hyper-heuristic for tailoring population-based metaheuristics. Moreover, our approach considers search operators from well-known techniques as building blocks for new ones. We test this strategy through four benchmark functions and by varying their dimensions. We obtain metaheuristics with diverse configurations. We observe a possible performance boost when two or more search operators are considered. This could be due to previously unexplored interactions between such operators.**

***Index Terms*—Metaheuristic, Hyper-heuristic, Search operators, Evolutionary computation.**

I. Introduction

Technological breakthroughs offer us comfort and improve our quality of life. But it also gives birth to new continuous optimisation problems. Metaheuristics (MHs), defined as general-purpose methods, have been proposed to tackle such real-world problems. They are characterised by their flexibility, versatility, and algorithmic simplicity when facing a problem. A vast number of metaheuristics claiming to be the best one for solving engineering problems has been reported [1], [2]. However, they are not so general: the *no-free-lunch* theorem reigns. In practice, researchers must know how to properly select a MH for a given problem. Even then, they must also know how to configure its parameters. Therefore, the question of deciding which MH is worth implementing to solve a defined problem remains open.

Throughout the last three decades, several MHs with curious metaphors (or "flavours") have been presented to face various problems [3], [4]. However, they are not so different from conventional metaheuristics such as Simulated Annealing (SA), Differential Evolution (DE), and Particle Swarm Optimisation (PSO). If we disassemble them into simple heuristics, say, search operators, it is easy to notice that many of these are variations of some basic ones, *e.g.,* mutation [5], crossover [6], and Lévy flight [7]. Some authors have taken advantage of this fact for setting a procedure by selecting two or more simple heuristics to render MHs with excellent performances in particular problems [8].

The GIEE-Intelligent Systems at the Tecnológico de Monterrey (México) and the Chinese Academy of Sciences (China) funded this work. I. Amaya thanks Consejo Nacional de Ciencia y Tecnología (CONACyT) for its financial support through Basic Science Project with grant number 287479.

The idea of algorithms for combining heuristics is not entirely new, as it goes back to the 1960s. But only recently it has grown into an optimisation sub-area known as hyper-heuristics (HHs) [9]. Of course, some alternative methodologies have also appeared, but they are beyond the scope of this work.

HHs provide a general approach, which have been defined as a high-level heuristic. Such a heuristic selects or modifies low-level heuristics as a mean of finding better solutions for a problem domain [10]. It is interesting to see that many HHs have been applied to combinatorial problems [11], [12], whilst only a few have dealt with continuous ones [10]. For example, Miranda *et al.* proposed a Hybrid Hyper-Heuristic for Algorithm Design (H3AD) [13]. The authors used H3AD for optimising (or redesigning) the well-known Particle Swarm Optimisation (PSO) algorithm to 60 continuous benchmark problems. Their results showed that, in at least 80% of the times, customised algorithms outperformed their counterparts. Moreover, the selection process reduced computational cost. However, this strategy employed the PSO grammar as design schema. So, it may prove somewhat difficult to extend this idea to other metaheuristics. In a similar approach, Abell *et al.* generated an algorithm portfolio that included DE and PSO, and which targeted the Black-Box Optimisation Benchmarking problems [14]. They demonstrated that, by incorporating feature extraction it is possible to consider problem structure when selecting the best solution method within the portfolio.

Despite the scarce research dealing with high-level solvers for continuous optimisation, the available ones lack a proper generalisation scheme. Hence, this work proposes a HH-based strategy for filling such a knowledge gap. Our approach generates custom population-based MHs to solve continuous optimisation problems. Each custom MH is built by cascading one or more search operators collected from well-known MHs. We test our proposal on four benchmark problems and under several dimensions. Moreover, we consider a different number of operators for tailoring MHs. Therefore, our analysis focuses on the influence of problem dimensionality, as well as on the effects of MH length.

This document is organised as follows. Sect. II introduces the theoretical foundations, whilst Sect. III describes the testing methodology. Then, Sect. IV presents and discusses the obtained data. Finally, Sect. V highlights the most relevant conclusions and lays out some paths for future works.

II. Theoretical Foundations

A. Optimisation

Optimisation is somehow implicit in nature. Even so, it mainly concerns mathematical procedures for reaching the best result of a problem in practical engineering scenarios. In this work, we use the problem definition given by a feasible domain and an objective function to minimise, as follows.

Definition 1 (Problem domain)**.** *Let $\mathfrak{X}$ be a feasible set or problem domain such as it is simply established by*

$$\mathfrak{X} = \{\vec{x} \in \mathbb{R}^D \,:\, (\exists\, \vec{L}, \vec{U} \in \mathbb{R}^D)[\vec{L} \preceq \vec{x} \preceq \vec{U}]\}, \qquad (1)$$

with $\vec{L}$ and $\vec{U}$ as the lower and upper boundary vectors. Thus, D represents the dimensionality of the problem.

Definition 2 (Minimisation problem)**.** *Let $f(\vec{x})$ be a real-valued function defined on a set $\mathfrak{X} \neq \emptyset$ such as $f(\vec{x}) : \vec{x} \in \mathfrak{X} \subseteq \mathbb{R}^D \to \mathbb{R}$. Thus, a minimisation problem is stated as*

$$\vec{x}_* = \underset{\vec{x} \in \mathfrak{X}}{\arg\min} \left\{ f(\vec{x}) \right\}, \qquad (2)$$

where $\vec{x}_* \in \mathfrak{X}$ is the optimal vector (or solution) that minimises the objective function, i.e., $f(\vec{x}_*) \leq f(\vec{x})\ \forall\, \vec{x} \in \mathfrak{X}$.

B. Heuristics

A heuristic is a procedure that creates or modifies a candidate solution for a given problem instance. There are many classifications of heuristics in the literature. Most of them relate to combinatorial optimisation domains [9], whilst rather scarce on continuous ones [10].

In this work, we categorise continuous heuristics in three groups, extending the ideas of [9], [10]: *low-level*, *mid-level*, and *high-level*. These relate to *simple heuristics*, *meta-heuristics*, and *hyper-heuristics*, respectively. Certainly, all of them are heuristics but operate under different conditions and domains. Fig. 1 shows an illustrative example of the aforementioned, which is also detailed below. However, since most heuristics use a set of search agents, we first need to establish the following concepts:

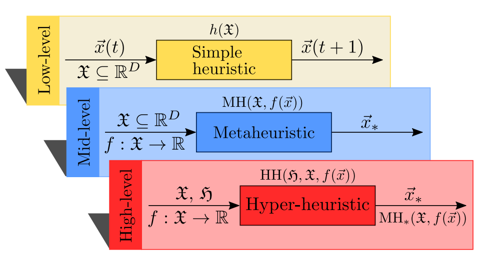


Fig. 1. Heuristics of three levels interacting with the problem domain ($\mathfrak{X}$) and the heuristic space ($\mathfrak{H}$). Since $f(\vec{x})$ is the objective function, $\vec{x}(t)$ is a candidate solution at time t, and $\vec{x}_*$ is the best solution found.

Definition 3 (Population)**.** *Let $X(t)$ be a finite set of N candidate solutions for an optimisation problem given by $\mathfrak{X}$ and $f(\vec{x})$ (*cf. *Definition 2) at time t in an iterative procedure,* i.e., *$X(t) = \{\vec{x}_1(t), \vec{x}_2(t), \ldots, \vec{x}_N(t)\}$. Then, $\vec{x}_n(t) \in \mathfrak{X}\ \forall\, n = 1, \ldots, N$ denotes the n-th candidate solution or search agent of, let us say, the population $X(t)$.*

Definition 4 (Best solution)**.** *Let $Z(t)$ be an arbitrary set of solutions, which can be designated as,* e.g., *the entire population $Z(t) = X(t)$, the n-th neighbourhood $Z(t) = Y_n(t)$, and the historical evolution of the n-th candidate $Z(t) = \{\vec{x}_n(0), \vec{x}_n(1), \ldots, \vec{x}_n(t)\}$. Therefore, let $\vec{x}_*(t) \in Z(t)$ be the best solution from $Z(t)$,* i.e., $\vec{x}_*(t) = \arg\min\{f(Z(t))\}$.

1) Simple Heuristics (SHs): They are the atomic unit in terms of search techniques that interact directly with problem domains. SHs are commonly categorised as *constructive* and *perturbative*. As their names suggest, a constructive heuristic renders new solutions from scratch while a perturbative heuristic modifies current solutions [15]. Thus, we adopt these categories adding another one, as Definition 5 describes.

Definition 5 (Simple Heuristic)**.** *Let $h \in \mathfrak{H}$ be a simple heuristic from the heuristic space $\mathfrak{H}$ such that either produces, modifies or evaluates a candidate solution $\vec{x} \in \mathfrak{X}$,* cf. *Definition 1, using a fitness metric,* e.g., *its objective function value $f(\vec{x})$,* cf. *Definition 2. Therefore, three types of simple heuristics can be stated as follows.*

Remark 1 (Initialiser)**.** *Let $h^i \in \mathfrak{H}_i \subset \mathfrak{H}$ be a simple heuristic that generates a candidate solution within the search space $\vec{x} \in \mathfrak{X}$ from scratch,* i.e., *$h^i : \mathbb{R}^D \to \mathfrak{X}$. Then, h^i is called Initialiser. The most common one in the literature is to place the agents within the feasible search space randomly,* e.g., *using a uniform random distribution.*

Remark 2 (Search Operator)**.** *Let $h^p \in \mathfrak{H}_p \subset \mathfrak{H}$ be a simple heuristic that modifies a candidate solution within the search space $\vec{x} \in \mathfrak{X}$ and updates it according to its corresponding fitness value via a selection criterion,* i.e., *$h^p : \mathfrak{X} \to \mathfrak{X}$. Thus, h^i is called search operator or Perturbator. There are several standard selection criteria such as direct, greedy, and Metropolis update.*

Remark 3 (Finaliser)**.** *Let $h^f \in \mathfrak{H}_f \subset \mathfrak{H}$ be a simple heuristic that evaluates the quality of a solution, during an iterative procedure, by using information about,* e.g., *the current solution, its fitness value, the current iteration, the previous candidate solutions, and other measurements. h^f is called Finaliser due to it maps $\mathbb{R}^D$, $\mathfrak{X}$, $\mathbb{R}$ and $\mathbb{Z}_+$ to $\mathbb{Z}_2$, which corresponds to a stop flag—a convergence measurement.*

2) Metaheuristics (MHs): They are defined as master strategies which control SHs. MHs are trendy in the literature because of their proven performance on different scenarios [2], [4]. Without loss of generality, most of the MHs have a scheme that Fig. 2 exemplifies and Definition 6 details.

Definition 6 (Metaheuristic)**.** *Let* MH *be an iterative procedure called metaheuristic that renders an optimal solution $\vec{x}_*$ for a given optimisation problem with an objective function $f(\vec{x})$ (*cf. *Definition 2). This procedure is represented as a finite sequence of simple heuristics (*cf. *Definition 5) to be*

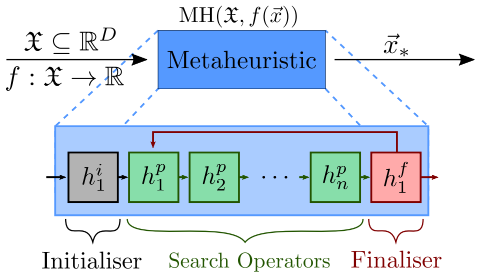


Fig. 2. Scheme of a metaheuristic. It comprises a sequence of simple heuristics (h_1^i, h_i^p, h_1^f) which iterates until a stopping flag is raised.

applied iteratively until a stopping condition is met, i.e., $\mathrm{MH} = (h_1^i, h_1^p, \ldots, h_n^p, h_1^f) \in \mathfrak{H}_i \times \mathfrak{H}_p^n \times \mathfrak{H}_f$.

Remark 4. *Notice that there is only one heuristic for each category of initialisers and finalisers. This is assumed for the sake of simplicity, but it can be extended with ease.*

Remark 5. *(Cardinality) An inherent property of metaheuristics is the cardinality, which is defined as the number of search operators implemented in it,* i.e., *disregarding the initialiser(s) and finaliser(s); ergo* $\#\mathrm{MH} = n$ *(cf. Remark 2). By way of standardisation, we denote a metaheuristic and its cardinality such as* MH^n*, where* $\mathrm{MH}^1 = \mathrm{MH}$.

3) Hyper-heuristics (HHs): Many researchers describe HHs as high-level heuristics controlling simple heuristics in the process of solving an optimisation problem [9]. Therefore, HHs move in the heuristic space to find a heuristic configuration that solves a given problem. With that in mind, a HH can be defined according to [10] as follows.

Definition 7 (Hyper-Heuristic)**.** *Let* $\mathbf{h} \in \mathfrak{H}^n$ *be a heuristic configuration from the heuristic space* $\mathfrak{H}$ (cf. *Definition 5), and let* $F(\mathbf{h}|\mathfrak{X}) : \mathfrak{H}^n \times \mathfrak{X} \to \mathbb{R}$ *be its performance measure function. Recall* $\mathfrak{X}$ *as the problem domain in an optimisation problem with an objective function* $f(\vec{x}) : \mathfrak{X} \to \mathbb{R}$ (cf. *Definitions 1-2). Then, a solution* $\vec{x}_* \in \mathfrak{X}$ *and its corresponding fitness value* $f(\vec{x}_*)$ *are found when a* $\mathbf{h}$ *is applied on* $\mathfrak{X}$*, so its performance* $F(\mathbf{h}|\mathfrak{X})$ *can also be determined. Therefore, let HH be a technique that solves*

$$(\mathbf{h}_*; \vec{x}_*) = \underset{\mathbf{h}\in\mathfrak{H}^n, \vec{x}\in\mathfrak{X}}{\arg\max} \{F(\mathbf{h}|\mathfrak{X})\}. \tag{3}$$

In other words, a HH searches for the optimal heuristic configuration $\mathbf{h}_*$ *that produces the optimal solution* $\vec{x}_*$ *with the maximal performance* $F(\mathbf{h}_*|\mathfrak{X})$.

Remark 6. *(Heuristic Configuration) When performing a hyper-heuristic process, a heuristic configuration* $\mathbf{h} \in \mathfrak{H}^n$ *is a way of referring to a metaheuristic MH* (cf. *Definition 6).*

III. Methodology

In this work, we followed a dual-stage methodology to test our approach for tailoring metaheuristics (MHs). As a first stage (Fig. 3) we extracted the Search Operators (SOs) from the following 10 well-known MHs: Random Search [16], Simulated Anneling [17], Genetic Algorithms [5], Cuckoo Search [7], Differential Evolution [18], Particle Swarm [19], [20], Firefly [21], Spiral [22], [23], Central Force [24], and Gravitational Search [25]. We extracted such operators via the MH scheme described in Definition 6. For each one of them, we also identified their controlling parameters. Afterward, we generated an expanded SOs database by employing five different value combinations for each parameter of each MH.

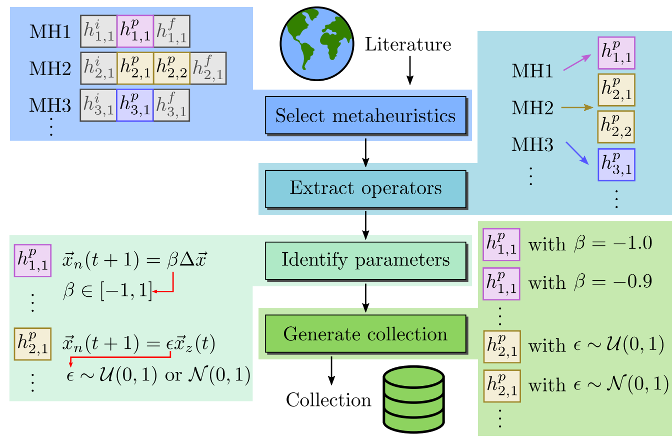


Fig. 3. First stage of the Methodology carried out in this work, which consists of the search operator collection.

In the second stage (Fig. 4), we implemented a Random Search approach to tune the HH. Such a process implied exploring the heuristic space given by the collection of SOs. This method was set to perform 5 steps, with 50 trials per step. Bear in mind that every time the HH builds a candidate MH^n, it does so following Definition 6. So, the metaheuristic considers a uniform random initialisation and its stop criterion is given by a maximum number of iterations. Also, the number of SOs in the MH^n equals its cardinality n. For our testing, population size and number of iterations were set to 30 and 100, respectively. Moreover, we considered cardinality values of 1, 2, and 3. These values were chosen for chiefly two reasons. First and foremost, because MHs available in literature usually fall within one of these values (*cf.* Sect. IV). Second, because we wanted to keep our approach as simple as possible since a higher cardinality expands the size of the search domain for the HH. It is important to highlight that we only tune each HH for five steps, due to computing power restrictions. We are aware of this drawback and plan on dealing with it in a future work. Nonetheless, and since each step considers 50 trials, tuning is carried out for a maximum of 250 function evaluations.

In order for the HH to improve, some performance metric is required. So, we measured the performance of a candidate MH^n, $F(\mathrm{MH}^n|\mathfrak{X})$, via its fitness values. Since the SOs are usually stochastic, we ran each candidate MH^n 100 times and recorded all fitness values. Hence, the final outcome is given by the sum between the median fitness value and its interquartile range.

Due to the exploratory nature of our work, we considered four representative benchmark functions commonly found in

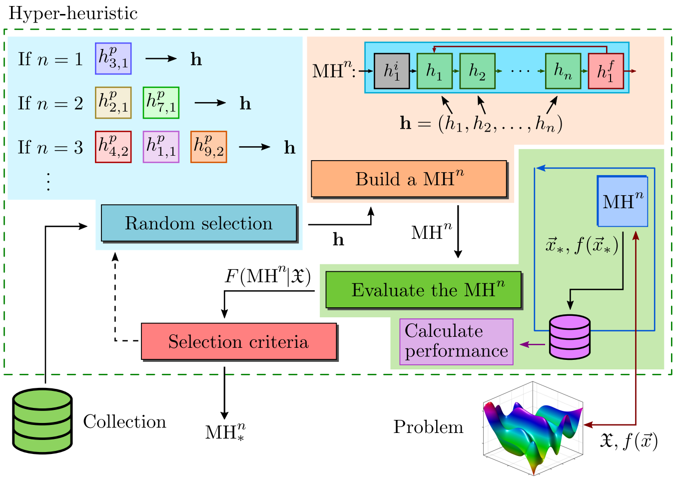


Fig. 4. Second stage of the Methodology carried out in this work, which consists of the hyper-heuristic implementation.

literature. For each one, we selected dimensionality values of 2, 10 and 30. These functions have minima equal to zero $f(\vec{x}_*) = 0$ but different domains $\mathfrak{X}$ and optimum lcoations $\vec{x}_*$. Also, they differ in difficulty due to their shape. So, we sorted them based on difficulty, as follows:

1) De Jong with $\mathfrak{X} = [-1000, 1000]^D$ and $\vec{x}_* = \vec{0}$.
2) Griewank with $\mathfrak{X} = [-600, 600]^D$ and $\vec{x}_* = \vec{0}$.
3) Ackley with $\mathfrak{X} = [-32.78, 32.78]^D$ and $\vec{x}_* = \vec{0}$.
4) Rosenbrock with $\mathfrak{X} = [-5, 5]^D$ and $\vec{x}_* = \vec{1}$.

In this work, all the experiments were carried out in Python™ 3.7.6 running on an ASUS® S46C with an Intel® Core™ i7-3537U CPU at 2.00-2.50 GHz, 6 GB RAM, and Canonical® Ubuntu™ 19.04-64 bit.

IV. Results

Table I presents a list with all the extracted operators. As can be seen, the 10 MHs yielded 22 search operators (SOs). Moreover, by considering 5 values for each parameter of every operator, we obtained a total of 925 SOs within the database. Nonetheless, and for the sake of brevity, we omit the whole collection. Bear in mind that the database could become bigger/smaller by increasing/decreasing the number of values per parameter. By using the MH scheme (*cf.* Fig. 2), we realised that most algorithms exhibit a cardinality of one. The only exceptions are Genetic Algorithms, Differential Evolution and Cuckoo Search, whose cardinality equals two. It is important to note that even if MHs have been split into the building blocks we call SOs, it does not imply that all SOs have the same intricacies. For example, random-based operators such as random sampling and random walk are straightforward since they represent a change in position of the agent. But, as we go deeper into Table I, we find dynamics-based operators that imply several and more elaborate operations. For example, in Gravitational Search the change in position for the agent depends on a velocity calculation that relies on the estimate of an acceleration value.

Fig. 5 summarises the performance achieved by hyper-heuristics on the target problems. Keep in mind that we were forced to implement a wider than usual search domain for De Jong's function. The reason: It is a straightforward multidimensional optimisation problem, so no difference was otherwise perceived. Interesting elements arise when moving to this second stage of testing. For example, we noticed that performance distribution is positively skewed, which is a desired feature as more values are closer to zero. Also, we observed that, in general, a higher cardinality seems to have a positive impact, allowing a better performance on more complex problems. However, care must be taken. In increasing cardinality, the size of the search space is also expanded. Hence, it becomes more difficult to find a proper sequence of search operators. For example, in the simplest scenario (Fig. 5(a)) moving from a cardinality of two to three dramatically improves performance and stability of hyper-heuristics dealing with problems in 10D. But, it also worsens the stability in problems with 30D. We believe this behaviour can be due to the nature of the function. This way, by considering 10 dimensions the problem may be simple enough so that many three-block combinations yield a good performance. Nonetheless, at 30D the problem is more complex and a fewer number of combinations may perform properly. Even so, this could be alleviated by allocating more resources to the tuning stage. Hence, this implies that cardinality must be correctly chosen, or that it could be tuned as a hyper-parameter. As an example of what may happen consider Griewank and Ackley functions (Fig. 5(b) and (c)). For 30 dimensions, MH^1 performed better than the other ones. Similarly, for the Rosenbrock function (Fig. 5(d)) a cardinality of two yielded the best performance at the highest number of dimensions. Therefore, a higher cardinality does not guarantee a better performance, at least for metaheuristics with a limited number of iterations, as those found within this work. Also, bear in mind that most traditional metaheuristics are equivalent to a MH^1, whilst only a few are to a MH^2. Moreover, MH^3 can be recognised in the literature as hybrid or "exotically-named" methods. This implies that the HH may actually yield an "already known" metaheuristic. But, our approach is also flexible enough to provide hybrid algorithms with better performance.

So, it is paramount to analyze the structure of the generated solvers and determine whether they are equivalent to original metaheuristics. Table II to IV shows performance statistics for each cardinality value. Take in mind that the best results across cardinality are highlighted in bold. Hence, Table II indicates that the hyper-heuristic found for Griewank in 30D, and which considered a single search operator (SO), outperformed those with two and three SOs. But, should the problem had been in 2D, the hyper-heuristic with two SOs would have outperformed the other ones, as Table III indicates.

For the smallest configuration (*i.e.,* only one search operator, Table II), all the selected SOs correspond to those inspired by kinematics and gravitation. Particularly, operators based on Gravitational Search were frequent in all the problems, especially when the number of dimensions and the problem complexity increased. These MHs showed high precision in terms of low values for dispersion metrics such as standard

TABLE I

SEARCH OPERATORS EXTRACTED FROM 10 WELL-KNOWN METAHEURISTICS IN THE LITERATURE.

Name, Reference	Expressions[a]	Parameters[b]
Random Sample[c], [16]	$\vec{x}_n(t+1) = \vec{r}$	$\vec{r} \ni r_i \sim \mathcal{U}(-1,1)$
Random Walk[c], [17]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \alpha\vec{r}$	$\alpha \in [0.1, 0.9]$, $\vec{r} \ni r_i \sim \mathcal{U}(-1,1) \vee \mathcal{N}(0,1)$
Local Random Walk[c], [7]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \alpha\vec{r} \odot H(\vec{r} - p) \odot (\vec{x}_{z_1}(t) - \vec{x}_{z_2}(t))$	$\alpha \in [0.1, 0.9]$, $p \in [0.1, 0.9]$, $\vec{r} \ni r_i \sim \mathcal{U}(0,1)$, $z_1, z_2 \sim \mathcal{U}_{\mathcal{I}}(1, N)$, $z_1 \neq z_2$
Genetic Mutation, [5]	$\vec{x}_n(t+1) = (1 - \vec{m}) \odot \vec{x}_n(t) + \alpha\vec{m} \odot \vec{s}$, $\vec{m} = H(\vec{r} - p_m)$, $\forall\, n \in \{\lceil p_e N \rceil, \ldots, N\}$	$\alpha = 1.0$, $\vec{s} \ni s_i \sim \mathcal{U}(-1,1) \vee \mathcal{N}(0,1)$, $\vec{r} \ni r_i \sim \mathcal{U}(0,1)$, $p_e, p_m \in [0.1, 0.9]$
Genetic Single-Point Crossover[d], [5]	$\vec{x}_n(t+1) = (1 - \vec{m}) \odot \vec{u} + \vec{m} \odot \vec{v}$, $\vec{m} = H(\vec{i} - z)$, $\vec{i} = (1, 2, \ldots, D)^\intercal \in \mathbb{R}^D$	$z \sim \mathcal{U}_{\mathcal{I}}(1, D)$
Genetic Two-Points Crossover[d], [5]	$\vec{x}_n(t+1) = (1 - \vec{m}) \odot \vec{u} + \vec{m} \odot \vec{v}$, $\vec{m} = H(\vec{i} - z_1) - H(\vec{i} - z_2)$, $\vec{i} = (1, 2, \ldots, D)^\intercal \in \mathbb{R}^D$	$z_1, z_2 \sim \mathcal{U}_{\mathcal{I}}(1, D) \wedge z_1 < z_2$
Genetic Uniform Crossover[d], [5]	$\vec{x}_n(t+1) = (1 - \vec{m}) \odot \vec{u} + \vec{m} \odot \vec{v}$, $\vec{m} = H(\vec{r} - 0.5)$	$\vec{r} \ni r_i \sim \mathcal{U}(0,1)$
Genetic Blend Crossover[d], [5]	$\vec{x}_n(t+1) = \vec{r} \odot \vec{u} + (1 - \vec{r}) \odot \vec{v}$	
Genetic Linear Crossover[d], [5]	$\vec{x}_n(t+1) = \alpha\vec{u} + \beta\vec{v}$	$\alpha = \beta = 0.5$
Lévy Flight[c], [7]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \alpha\vec{r} \odot (\vec{x}_n(t) - \vec{x}_*(t))$	$\alpha \in [0.1, 0.9]$, $\vec{r} \ni r_i \sim \mathcal{L}(\beta)$, $\beta \in [1.25, 1.75]$
Differential Mutation /rand/M[c], [18]	$\vec{x}_n(t+1) = \vec{x}_{z_1}(t) + F\sum_{m=1}^{M}\left(\vec{x}_{z_{2m}}(t) - \vec{x}_{z_{2m+1}}(t)\right)$	$F \in [0.5, 2.5]$, $M \in \{1, 2, 3\}$
Differential Mutation /best/M[c], [18]	$\vec{x}_n(t+1) = \vec{x}_*(t) + F\sum_{m=1}^{M}\left(\vec{x}_{z_{2m}}(t) - \vec{x}_{z_{2m+1}}(t)\right)$	$z_i \sim \mathcal{U}_{\mathcal{I}}(1, N) : \bigcap_j \{z_j\} = \emptyset$
Differential Mutation /current/M[c], [18]	$\vec{x}_n(t+1) = \vec{x}_n(t) + F\sum_{m=1}^{M}\left(\vec{x}_{z_{2m}}(t) - \vec{x}_{z_{2m+1}}(t)\right)$	
Differential Mutation /rand-to-best/M[c], [18]	$\vec{x}_n(t+1) = \vec{x}_{z_1}(t) + F \cdot (\vec{x}_*(t) - \vec{x}_{z_2}(t)) + F\sum_{m=1}^{M}\left(\vec{x}_{z_{2m+1}}(t) - \vec{x}_{z_{2m+2}}(t)\right)$	
Differential Mutation /current-to-best/M[c], [18]	$\vec{x}_n(t+1) = \vec{x}_n(t) + F \cdot (\vec{x}_*(t) - \vec{x}_{z_1}(t)) + F\sum_{m=1}^{M}\left(\vec{x}_{z_{2m}}(t) - \vec{x}_{z_{2m+1}}(t)\right)$	
Differential Mutation /rand-to-best-and-current/M[c], [18]	$\vec{x}_n(t+1) = \vec{x}_{z_1}(t) + F \cdot (\vec{x}_*(t) - \vec{x}_{z_2}(t) + \vec{x}_{z_3}(t) - \vec{x}_n(t)) + F\sum_{m=1}^{M}\left(\vec{x}_{z_{2m+2}}(t) - \vec{x}_{z_{2m+3}}(t)\right)$	
Inertial Swarm Dynamic, [19]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \vec{v}_n(t+1)$ $\vec{v}_n(t+1) = \omega\vec{v}_n(t) + \phi_1\vec{r}_1 \odot (\vec{x}_{n,*}(t) - \vec{x}_n(t)) + \phi_2\vec{r}_2 \odot (\vec{x}_*(t) - \vec{x}_n(t))$	$\omega \in [0.1, 1.0]$, $\phi_1, \phi_2 \in [1.1, 4.1]$, $\vec{r}_i \ni r_i \sim \mathcal{U}(0,1)\, \forall i \in \{1, 2\}$
Constricted Swarm Dynamic, [20]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \vec{v}_n(t+1)$ $\vec{v}_n(t+1) = \chi\left(\vec{v}_n(t) + \phi_1\vec{r}_1 \odot (\vec{x}_{n,*}(t) - \vec{x}_n(t)) + \phi_2\vec{r}_2 \odot (\vec{x}_*(t) - \vec{x}_n(t))\right)$ $\chi = \sqrt{\kappa} - (\sqrt{\kappa} - 2\kappa/(\phi - \sqrt{\phi(\phi - 4)})H(\phi - 4)$, $\phi = \phi_1 + \phi_2$	$\kappa \in [0.1, 1.0]$, $\phi_1, \phi_2 \in [1.1, 4.1]$ $\vec{r}_i \ni r_i \sim \mathcal{U}(0,1)\, \forall i \in \{1, 2\}$
Firefly Dynamic[c], [21]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \alpha\vec{r} + \beta\sum_{k=1,k\neq n}^{N} H(-\Delta I_{n,k})\Delta\vec{x}_{n,k}e^{-\gamma\lVert\Delta\vec{x}_{n,k}\rVert_2^2}$ $\Delta I_{n,k} = f(\vec{x}_k) - f(\vec{x}_n)$, $\Delta\vec{x}_{n,k} = \vec{x}_k(t) - \vec{x}_n(t)$	$\alpha \in [0.0, 0.5]$, $\beta = 1.0$, $\gamma \in [10, 990]$, $\vec{r} \ni r_i \sim \mathcal{U}(-0.5, 0.5) \vee \mathcal{N}(0,1)$
Spiral Dynamic[e], [23]	$\vec{x}_n(t+1) = \vec{x}_*(t) - \vec{r}\mathbf{R}_D(\theta)(\vec{x}_n(t) - \vec{x}_*(t))$	$\mathbf{R}_D(\theta) \in \mathbb{R}^{D\times D}$, $\theta \in [0.001°, 179°]$, $\vec{r} \ni r_i \sim \mathcal{U}(r_0 - \sigma, r_0 + \sigma)$, $r_0 \in [0.001, 0.99]$, $\sigma \in [0.0, 0.5]$
Central Force Dynamic, [24]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \frac{1}{2}\vec{a}_n(t)\Delta t^2$, $\vec{a}_n(t) = G\sum_{k=1,k\neq n}^{N} H(\Delta M_{n,k})(\Delta M_{n,k})^{\alpha}\frac{\Delta\vec{x}_{n,k}}{\lVert\Delta\vec{x}_{n,k}\rVert_2^{\beta}+\varepsilon}$ $\Delta M_{n,k} = f(\vec{x}_k(t)) - f(\vec{x}_n(t))$, $\quad \Delta\vec{x}_{n,k} = \vec{x}_k(t) - \vec{x}_n(t)$	$G \in [0.0, 0.01]$, $\alpha \in [0.0, 0.01]$, $\beta \in [1.25, 1.75]$, $\Delta t = 1.0$, $\varepsilon = 10^{-23}$
Gravitational Search, [25]	$\vec{x}_n(t+1) = \vec{x}_n(t) + \vec{v}_n(t+1)$, $\vec{v}_n(t+1) = \vec{r} \odot \vec{v}_n(t) + \vec{a}_n(t)$ $\vec{a}_n(t) = Ge^{-\alpha t}\sum_{k=1,k\neq n}^{N} M_k(t)\vec{r}_k \odot \frac{\Delta\vec{x}_{n,k}}{\lVert\Delta\vec{x}_{n,k}\rVert_2+\varepsilon}$ $M_k(t) = \frac{f(\vec{x}_\circ(t)) - f(\vec{x}_n(t))}{Nf(\vec{x}_\circ(t)) - \sum_{k=1}^{N} f(\vec{x}_k(t))}$	$\vec{r} \ni r_i \sim \mathcal{U}(0,1)$, $G \in [0.0, 1.0]$, $\alpha \in [0.0, 0.04]$, $\varepsilon = 10^{-23}$,

[a] $\vec{x}_n(t) \in X(t)$ is the vector position of the n-th agent (*cf.* Definition 3), and $\vec{x}_*(t) = \arg\min\{\bigcup_{n=1}^{N} f(\vec{x}_n(t))\}$ and $\vec{x}_\circ(t) = \arg\max\{\bigcup_{n=1}^{N} f(\vec{x}_n(t))\}$ correspond to the best and worst positions from the current population. $\odot$ is the Hadamard-Schur product and $H : \mathbb{R}^D \to \mathbb{Z}_2^D$ is the component-wise Heaviside function with $H(0) = 1$.

[b] $\vec{r}$ and $\vec{s}$ are vectors of i.i.d. random variables with either Uniform $\mathcal{U}(y_l, y_u)$, Normal $\mathcal{N}(\mu, \sigma)$ or Lévy stable $\mathcal{L}(\beta)$ distribution. z_i stands an integer random variable with uniform distribution $\mathcal{U}_{\mathcal{I}}(y_l, y_u)$. Parameter values were picked up from an evenly spaced sequence of five values in their intervals defined in the third column; *e.g.,* for $\alpha \in [0.1, 0.9]$, we considered $\alpha = 0.1, 0.3, 0.5, 0.7$, and 0.9.

[c] These operators use a greedy selection mechanism for updating new positions, *i.e.,* $\vec{x}_n(t+1)$ is accepted iff $f(\vec{x}_n(t+1)) < \vec{x}_n(t)$. The others use a direct update.

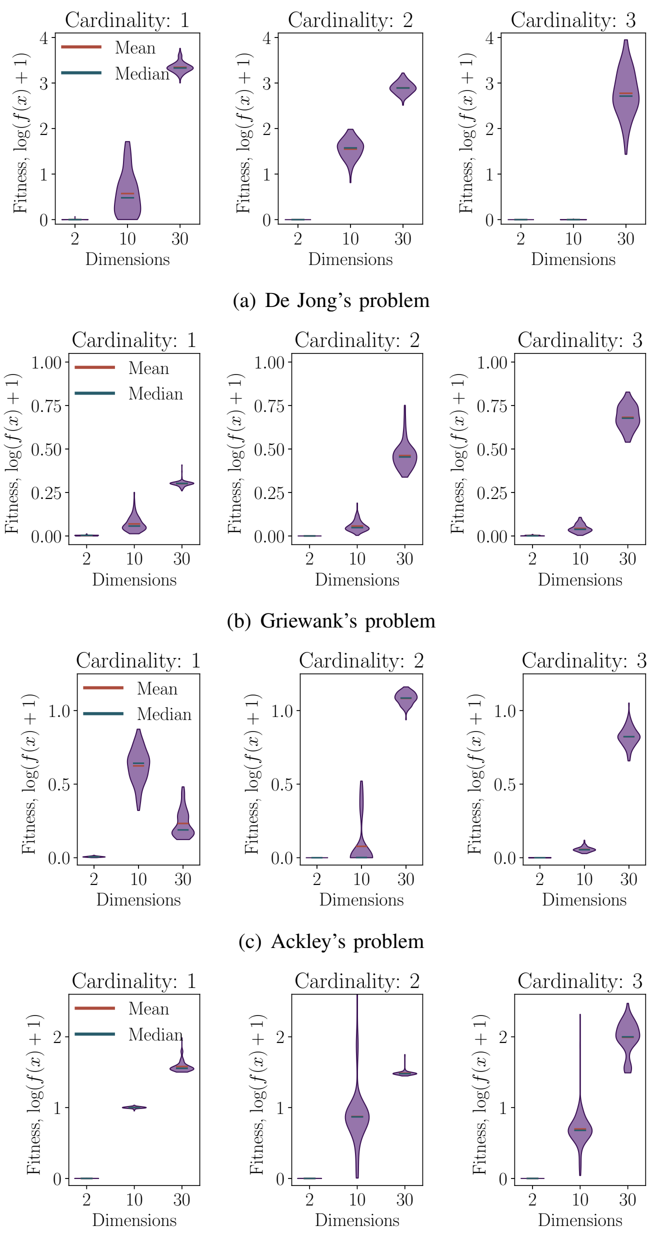
[d] $\vec{u}$ and $\vec{v}$ are the parents selected from a mating pool of size $\lfloor p_m N \rceil$, since $p_m \in [0, 1]$ is the mating pool factor and N is the population size, by using a pairing scheme such as Roulette Wheel, Rank Weighting, Random, Even-and-Odd, and Tournament with given size of two or three and probability of 100 % [5].

[e] $\mathbf{R}_D(\theta)$ is the rotation matrix determined by the product of all the combinations of two-dimensional rotation matrices by utilising the Euler-Rodrigues's rotation formula.

deviation and interquartile range. Besides, they rendered the best solutions for functions Griewank and Ackley in 30 dimensions. For metaheuristics with more than one operator, Tables III and IV, we noticed excellent results for problems with 2 and 10 dimensions. Mainly, MHs with a cardinality of two achieved the best solutions for functions De Jong and Rosenbrock. It is also worth mentioning that the heuristic configurations found are diverse, and they may never have been thought in the literature.

Besides, it is important to remark that the selected Spiral Dynamic operators use random radii between $r_0 - \sigma$ and $r_0 + \sigma$ with $\sigma > r_0$. In magnitude, these values provide faster convergence to the solution, ideal for intensification. In sign, when they are negatives, they add 180° to the rotation angle of the spiral trajectories. So, agents have the chance of being reflected and boost their convergence towards the rotation reference. This dynamic remains almost unknown in the literature, and it would be interesting to deepen on it because of its high potential for intensification phases.

Furthermore, it is essential to mention that these results are



(a) De Jong's problem

(b) Griewank's problem

(c) Ackley's problem

(d) Rosenbrock's problem

Fig. 5. Results of four problems, with different dimensionalities, achieved by metaheuristics obtained via the hyper-heuristic varying their cardinalities.

preliminary and are sharply limited to MHs which only have a hundred iterations to solve a problem. Undoubtedly, this has some pros and cons, depending on the application. For this reason, we do not rush to conclude that MHs with more than two search operators have a poor performance. Indeed, we are confident that they may perform better if more tuning steps are allowed. But, this requires more computing resources than those which were available. Nonetheless, we plan on tackling this problem in the near future.

V. Conclusions and Future Works

In this work, we proposed a strategy based on a hyper-heuristic (HH) search to generate custom population-based metaheuristics (MHs) for solving continuous optimisation problems. For that purpose, we split each MH into building blocks defined as search operators (SOs). So, the HH builds a MH given by a set of such blocks. Moreover, we defined a property deemed as cardinality, which represents the length of the set. We collected 22 different operators from 10 well-known MHs available in literature. Such a collection represented the heuristic space for our HH. Subsequently, we implemented a Random Search approach for tuning the HH over four benchmark problems (*i.e.,* De Jong, Griewank, Ackley, and Rosenbrock). We considered the sum of median and interquartile range of fitness values as the performance metric. Moreover, we analysed the performance of our proposed approach over three dimensionalities for each problem (2, 10, and 30), and while creating MHs with three cardinality values (1, 2, and 3). All the MHs were set to carry out only 100 iterations using 30 agents.

Our data revealed diverse configurations, which may never have been thought of. For example, the MH^3 (*cf.* Table IV) found for Rosenbrock in 30D is a sequence based on the Lévy Flight, Firefly Dynamic, and Spiral Dynamic operators. In this case, the first and third operators promote the exploration and exploitation of search space, respectively. Meanwhile, the second operator refines the previous perturbations in a deterministic fashion. The reason: $\alpha = 0.0$, so randomness is nullified. Even though, we could name them with sophisticated names or explain them with creative metaphors, that was not the focus of our work. Instead, we strove to propose a way for creating custom metaheuristics by exploring and exploding the nature of currently available methods.

In particular, we observed that MHs with a single operator render high precision. This can be reflected in low dispersion metrics (standard deviation and interquartile range). Metaheuristics with more than one operator achieved excellent results for problems with 2D and 10D. However, for problems in 30D performance was hindered. A possible reason for this phenomenon may fall upon the number of iterations available to solve the problem with the candidate MH. Likewise, the effect may be due to the number of tuning iterations. Increasing cardinality seemingly has a positive impact on problems with higher dimensions. Nonetheless, this also expands the search domain so more tuning iterations should be allowed. Conversely, a different stopping criteria could be implemented. In any case, this implies an increase in computing requirements so it should be carefully dealt with.

This primary work lays the groundwork for future research. We plan on increasing the size of the heuristic collection by including other well-known SOs from the literature. Also, we shall seek a way for considering cardinality and population size, for example, as design variables. This way, the HH would not only find a proper set of search operators, but also the set length that should be used and the number of agents that should be considered in the search. We also look forward to complementing the proposed strategy with Data Science techniques to enhance exploration of the heuristic space. This way, a more robust approach could be used when tuning the

TABLE II

HYPER-HEURISTIC PERFORMANCE IN DIFFERENT OPTIMISATION PROBLEMS AND NUMBER OF DIMENSIONS, WHEN CUSTOMISING METAHEURISTICS WITH A CARDINALITY OF ONE. VALUES IN BOLD REPRESENT THE BEST RESULT BETWEEN DATA FROM TABLE II TO IV.

Problem	Dim.	Avg.	St. Dev.	Med.	IQR	Min.	Max.	Best Search Operator
De Jong	2	1e-03	0.016	1e-84	3e-82	4e-90	0.163	Spiral Dynamic with $r_0 = 0.001$, $\theta = 90°$, $\sigma = 0.5$
	10	6.270	10.68	2.018	4.890	5e-03	50.78	Inertial Swarm Dynamic with $\omega = 0.325$, $\phi_1 = 2.6$, $\phi_2 = 1.85$
	30	2311	729.4	2140	568	1006	5767	Gravitational Search with $G = 0.5$, $\alpha = 0.0$
Griewank	2	5e-03	5e-03	7e-03	7e-03	1e-13	0.027	Inertial Swarm Dynamic with $\omega = 0.55$, $\phi_1 = 2.6$, $\phi_2 = 1.1$
	10	0.179	0.124	0.140	0.139	0.030	0.780	Constricted Swarm Dynamic with $\kappa = 0.325$, $\phi_1 = 1.85$, $\phi_2 = 1.85$
	30	**1.001**	**0.083**	**1.004**	**0.048**	**0.817**	**1.561**	Gravitational Search with $G = 0.75$, $\alpha = 0.02$
Ackley	2	0.015	8e-03	0.014	0.011	1e-03	0.038	Gravitational Search with $G = 0.5$, $\alpha = 0.03$
	10	3.358	1.13	3.39	1.502	1.096	6.478	Central Force Dynamic with $G = 0.005$, $\alpha = 0.0075$, $\beta = 1.5$
	30	**0.749**	**0.396**	**0.546**	**0.461**	**0.329**	**2.034**	Gravitational Search with $G = 0.75$, $\alpha = 0.02$
Rosenbrock	2	5e-06	9e-06	2e-06	5e-06	5e-09	7e-05	Central Force Dynamic with $G = 0.0075$, $\alpha = 0.01$, $\beta = 1.25$
	10	9.035	**0.271**	9.020	**0.383**	7.992	**9.808**	Gravitational Search with $G = 0.75$, $\alpha = 0.01$
	30	37.76	8.891	34.93	4.318	30.7	95.61	Gravitational Search with $G = 1.0$, $\alpha = 0.01$

TABLE III

HYPER-HEURISTIC PERFORMANCE IN DIFFERENT OPTIMISATION PROBLEMS AND NUMBER OF DIMENSIONS, WHEN CUSTOMISING METAHEURISTICS WITH A CARDINALITY OF TWO. VALUES IN BOLD REPRESENT THE BEST RESULT BETWEEN DATA FROM TABLE II TO IV.

Problem	Dim.	Avg.	St. Dev.	Med.	IQR	Min.	Max.	Best Search Operators
De Jong	2	4e-79	3e-78	4e-82	6e-81	6e-92	2e-77	Spiral Dynamic with $r_0 = 0.24825$, $\theta = 90°$, $\sigma = 0.5$ Differential Mutation/current-to-best/1 with $F = 2.0$
	10	38.88	19.07	36.58	23.8	5.489	95.43	Inertial Swarm Dynamic with $\omega = 0.1$, $\phi_1 = 3.35$, $\phi_2 = 1.1$ Central Force Dynamic with $G = 0.005$, $\alpha = 0.01$, $\beta = 1.75$
	30	**818.9**	**252**	777.2	**315.7**	**324.8**	**1661**	Constricted Swarm Dynamic with $\kappa = 0.775$, $\phi_1 = 4.1$, $\phi_2 = 1.85$ Central Force Dynamic with $G = 0.01$, $\alpha = 0.0075$, $\beta = 1.5$
Griewank	2	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	Differential Mutation/rand-to-best/3 with $F = 0.5$ Differential Mutation/best/2 with $F = 0.5$
	10	0.143	0.085	0.117	0.084	**7e-03**	0.546	Inertial Swarm Dynamic with $\omega = 0.775$, $\phi_1 = 1.85$, $\phi_2 = 1.85$ Constricted Swarm Dynamic with $\kappa = 1.0$, $\phi_1 = 1.1$, $\phi_2 = 4.1$
	30	1.949	0.590	1.847	0.535	1.178	4.642	Differential Mutation/current-to-best/1 and $F = 1.5$ Constricted Swarm Dynamic with $\kappa = 0.325$, $\phi_1 = 2.6$, $\phi_2 = 3.35$
Ackley	2	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	Differential Mutation/best/2 and $F = 2.0$ Constricted Swarm Dynamic with $\kappa = 1.0$, $\phi_1 = 1.85$, $\phi_2 = 4.1$
	10	0.294	0.621	**3e-03**	**0.013**	**4e-05**	2.319	Differential Mutation/rand-to-best/3 with $F = 0.5$ Inertial Swarm Dynamic with $\omega = 0.1$, $\phi_1 = 1.1$, $\phi_2 = 2.6$
	30	11.16	1.131	11.17	1.611	7.667	13.49	Random Walk with $r_i \sim \mathcal{U}(-1, 1)$, $\alpha = 0.1$ Differential Mutation/current-to-best/1 with $F = 0.5$
Rosenbrock	2	1e-06	9e-06	8e-21	9e-17	2e-31	9e-05	Differential Mutation/rand-to-best-and-current/3 with $F = 2.0$ Spiral Dynamic with $r_0 = 0.4955$, $\theta = 179°$, $\sigma = 0.125$
	10	16.49	53.83	6.401	3.02	**0.013**	487	Local Random Walk with $p = 0.5$, $\alpha = 0.1$ Inertial Swarm Dynamic with $\omega = 0.1$, $\phi_1 = 1.1$, $\phi_2 = 2.6$
	30	**29.88**	**3.798**	**29.17**	**0.784**	**26.98**	**55.13**	Gravitational Search with $G = 1.0$, $\alpha = 0.02$ Gravitational Search with $G = 0.75$, $\alpha = 0.04$

hyper-heuristic.

REFERENCES

[1] K. Sörensen, M. Sevaux, and F. Glover, "A history of metaheuristics," *Handbook of Heuristics*, vol. 2-2, pp. 791–808, 2018.

[2] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2191–2233, 2019.

[3] K. Sörensen, "Metaheuristics-the metaphor exposed," *International Transactions in Operational Research*, vol. 22, no. 1, pp. 3–18, 2015.

[4] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Computers & Industrial Engineering*, vol. 137, p. 106040, 2019.

[5] C. W. Ahn, *Practical genetic algorithms*, vol. 18. Wiley-Interscience, 2006.

[6] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[7] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp. 210–214, IEEE, 2009.

[8] J. Del 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 and Evolutionary Computation*, vol. 48, pp. 220–250, 2019.

[9] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: revisited,"

TABLE IV
HYPER-HEURISTIC PERFORMANCE IN DIFFERENT OPTIMISATION PROBLEMS AND NUMBER OF DIMENSIONS, WHEN CUSTOMISING METAHEURISTICS WITH A CARDINALITY OF THREE. VALUES IN BOLD REPRESENT THE BEST RESULT BETWEEN DATA FROM TABLE II TO IV.

Problem	Dim.	Avg.	St. Dev.	Med.	IQR	Min.	Max.	Best Search Operators
De Jong	2	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	Gravitational Search with $G = 0.5$, $\alpha = 0.03$ Inertial Swarm Dynamic with $\omega = 1.0$, $\phi_1 = 4.1$, $\phi_2 = 3.35$ Genetic Linear Crossover with $\alpha = \beta = 0.5$, Random Pairing, $p_m = 0.9$
	10	**1e-03**	**4e-03**	**1e-04**	**3e-04**	**4e-07**	**0.0226**	Inertial Swarm Dynamic with $\omega = 0.1$, $\phi_1 = 1.85$, $\phi_2 = 4.1$ Constricted Swarm Dynamic with $\kappa = 1.0$, $\phi_1 = 2.6$, $\phi_2 = 4.1$ Differential Mutation/best/1 with $F = 2.5$
	30	1149	1558	**515.8**	1097	26.16	8904	Genetic Two-Points Crossover with Tournament Pairing (2, 100%), $p_m = 0.7$ Differential Mutation/rand-to-best-and-current/2 with $F = 1.0$ Constricted Swarm Dynamic with $\kappa = 0.55$, $\phi_1 = 2.6$, $\phi_2 = 1.1$
Griewank	2	4e-03	4e-03	1e-03	7e-03	1e-06	0.020	Inertial Swarm Dynamic with $\omega = 0.325$, $\phi_1 = 4.1$, $\phi_2 = 1.1$ Spiral Dynamic with $r_0 = 0.001$, $\theta = 45.5^\circ$, $\sigma = 0.25$ Central Force Dynamic with $G = 0.005$, $\alpha = 0.0$, $\beta = 1.75$
	10	**0.107**	**0.061**	**0.091**	**0.081**	**7e-03**	**0.279**	Inertial Swarm Dynamic with $\omega = 1.0$, $\phi_1 = 2.6$, $\phi_2 = 1.1$ Spiral Dynamic with $r_0 = 0.4955$, $\theta = 1^\circ$, $\sigma = 0.375$ Firefly Dynamic with $r_i \sim \mathcal{N}(0, 1)$, $\alpha = 0.25$, $\beta = 1.0$, $\gamma = 500.0$
	30	3.872	0.757	3.758	1.156	2.460	5.723	Constricted Swarm Dynamic with $\kappa = 0.775$, $\phi_1 = 3.35$, $\phi_2 = 3.35$ Genetic Mutation with $\alpha = 1.0$, $p_e = 0.225$, $p_m = 0.3$, $s_i \sim \mathcal{N}(0, 1)$ Firefly Dynamic with $r_i \sim \mathcal{U}(0, 1)$, $\alpha = 0.125$, $\beta = 1.0$, $\gamma = 500.0$
Ackley	2	7e-17	5e-16	**0.000**	**0.000**	**0.000**	4e-15	Genetic Blend Crossover with Random Pairing, $p_m = 0.3$ Constricted Swarm Dynamic with $\kappa = 1.0$, $\phi_1 = 2.6$, $\phi_2 = 1.1$ Constricted Swarm Dynamic with $\kappa = 0.775$, $\phi_1 = 4.1$, $\phi_2 = 1.1$
	10	**0.140**	**0.044**	0.133	0.047	0.067	**0.315**	Lévy Flight with $\alpha = 0.9$, $\beta = 1.625$ Local Random Walk with $\alpha = 0.5$, $p = 0.1$ Central Force Dynamic with $G = 0.0025$, $\alpha = 0.0$, $\beta = 1.375$
	30	5.720	1.043	5.65	1.212	3.554	10.26	Central Force Dynamic with $G = 0.0075$, $\alpha = 0.005$, $\beta = 1.75$ Constricted Swarm Dynamic with $\kappa = 0.325$, $\phi_1 = 1.85$, $\phi_2 = 4.1$ Differential Mutation/best/3 with $F = 0.5$
Rosenbrock	2	**5e-25**	**3e-24**	**1e-29**	**1e-28**	**0.000**	**3e-23**	Lévy Flight with $\alpha = 0.7$, $\beta = 1.5$ Firefly Dynamic with $r_i \sim \mathcal{N}(0, 1)$, $\alpha = 0.0$, $\beta = 1.0$, $\gamma = 745.0$ Spiral Dynamic with $r_0 = 0.001$, $\theta = 179^\circ$, $\sigma = 0.375$
	10	**6.742**	21.32	**3.777**	1.475	0.099	206.3	Constricted Swarm Dynamic with $\kappa = 1.0$, $\phi_1 = 2.6$, $\phi_2 = 1.85$ Constricted Swarm Dynamic with $\kappa = 1.0$, $\phi_1 = 1.1$, $\phi_2 = 3.35$ Differential Mutation/rand-to-best/2 with $F = 2.0$
	30	109.5	49.75	98.73	50.01	29.96	295.6	Constricted Swarm Dynamic with $\kappa = 0.775$, $\phi_1 = 1.85$, $\phi_2 = 3.35$ Differential Mutation/rand-to-best/3 with $F = 0.5$ Central Force Dynamic with $G = 0.01$, $\alpha = 0.0075$, $\beta = 1.75$

in *Handbook of Metaheuristics*, pp. 453–477, Springer, 2019.

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

[11] I. Amaya, J. C. Ortiz-Bayliss, A. Rosales-Pérez, A. E. Gutiérrez-Rodríguez, S. E. Conant-Pablos, H. Terashima-Marín, and C. A. Coello Coello, "Enhancing Selection Hyper-Heuristics via Feature Transformations," *IEEE Computational Intelligence Magazine*, vol. 13, no. 2, pp. 30–41, 2018.

[12] I. Amaya, J. C. Ortiz-Bayliss, S. Conant-Pablos, and H. Terashima-Marin, "Hyper-heuristics Reversed: Learning to Combine Solvers by Evolving Instances," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1790–1797, IEEE, jun 2019.

[13] P. B. Miranda, R. B. Prudêncio, and G. L. Pappa, "H3ad: A hybrid hyper-heuristic for algorithm design," *Information Sciences*, vol. 414, pp. 340–354, 2017.

[14] T. Abell, Y. Malitsky, and K. Tierney, "Fitness Landscape Based Features for Exploiting Black-Box Optimization Problem Structure," Tech. Rep. December, IT University, Copenhagen, 2012.

[15] G. Woumans, L. De Boeck, J. Beliën, and S. Creemers, "A column generation approach for solving the examination-timetabling problem," *European Journal of Operational Research*, vol. 253, no. 1, pp. 178–194, 2016.

[16] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. Feb, pp. 281–305, 2012.

[17] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[18] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution-An updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization (pso)," in *Proc. IEEE International Conference on Neural Networks, Perth, Australia*, pp. 1942–1948, 1995.

[20] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 1, pp. 58–73, 2002.

[21] X.-S. Yang *et al.*, "Firefly algorithm," *Nature-inspired metaheuristic algorithms*, vol. 20, pp. 79–90, 2008.

[22] K. Tamura and K. Yasuda, "Primary study of spiral dynamics inspired optimization," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 6, no. S1, pp. S98–S100, 2011.

[23] 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 *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, 1, pp. 1–6, IEEE, nov 2017.

[24] R. A. Formato, "Central force optimization: A new deterministic gradient-like optimization metaheuristic," *Opsearch*, vol. 46, no. 1, pp. 25–51, 2009.

[25] A. Biswas, K. K. Mishra, S. Tiwari, and A. K. Misra, "Physics-Inspired Optimization Algorithms: A Survey," *Journal of Optimization*, vol. 2013, pp. 1–16, 2013.