Metaheuristics in the balance: a survey on memory-saving approaches for platforms with seriously limited resources

Souheila Khalfi[a,b], Fabio Caraffini[c,*], Giovanni Iacca[d]

[a]*Department of Fundamental Informatics and its Applications, Constantine 2 University, Algeria*
[b]*Department of Mathematics and Computer Science, Mila University Center, Algeria*
[c]*Department of Computer Science, Computational Foundry, Swansea University, Swansea SA1 8EN, UK*
[d]*Department of Information Engineering and Computer Science, University of Trento, Italy*

Abstract

In the last three decades, the field of computational intelligence has seen a profusion of population-based metaheuristics applied to a variety of problems, where they achieved state-of-the-art results. This remarkable growth has been fuelled and, to some extent, exacerbated by various sources of inspiration and working philosophies, which have been thoroughly reviewed in several recent survey papers. However, the present survey addresses an important gap in the literature. Here, we reflect on a systematic categorisation of what we call "lightweight" metaheuristics, i.e., optimisation algorithms characterised by purposely limited memory and computational requirements. We focus mainly on two classes of lightweight algorithms: single-solution metaheuristics and "compact" optimisation algorithms. Our analysis is mostly focused on single-objective continuous optimisation. We provide an updated and unified view of the most important achievements in the field of lightweight metaheuristics, background concepts, and most important applications. We then discuss the implications of these algorithms and the main open questions and suggest future research directions.

Keywords: Single-solution algorithms, compact optimisation, evolutionary algorithms, swarm intelligence, metaheuristics.

1. Introduction

Hardware and software technologies are advancing at a fast pace and provide complex computing systems. In recent decades, strong competition among manufacturers has caused intense pressure to completely change the face of commercial electronics [1], leading to the ongoing development of computing devices with ever smaller dimensions but higher performance. These devices can range from extremely small form factor devices (e.g., microcontrollers, wearable devices, wireless sensors, and actuators) to larger devices such as hand-helds or tablets. A major concern in the design of these devices is that they usually perform computations under stringent physical, weight, and cost limitations, as well as real-time constraints and limited power capacities (e.g., with batteries that might be difficult or even impossible to replace/recharge). A categorisation of this kind of device, with its specific limitations, can be found, for example, in [2]. The general goal of manufacturers is to

*Corresponding author
Email addresses: souheila.khalfi@univ-constantine2.dz (Souheila Khalfi), fabio.caraffini@swansea.ac.uk (Fabio Caraffini), giovanni.iacca@unitn.it (Giovanni Iacca)

design optimal products that meet the requirements of the market without violating any hardware-dependent constraints. However, this is difficult in practise, given the impact these restrictions have on memory capacity, computational performance, and battery life.

Most computational problems that arise in such devices can be formulated in the form of an optimisation problem, i.e., one in which the optimal value of the given decision variables must be found with respect to a given objective function[1]. Note that due to the lack of a mathematical formulation or complexity of the problem, etc., these are challenging zeroth-order optimisation scenarios, where no assumption can be made on the properties of $f(\cdot)$. Typical examples are self-tuning the parameters of machine learning algorithms on board a device [3], dynamically adjusting the hardware settings (e.g., camera, microphones, battery consumption, etc.), characterising a user profile or providing customised recommendations [4].

To deal with such scenarios, a *metaheuristic* [5, 6], i.e., general purpose "generate and test" black-box optimisation methods, is the most logical choice. Metaheuristics do not guarantee convergence to the theoretical optimum but offer high applicability without needing any information on the problem at all, but rather learn the problem landscape to search for solutions. Their success in solving various numerical and real-world problems [7, 8] made them popular and the subject of continuous investigations. There are many algorithms of this kind in the literature, and choosing the most suitable for a specific problem is not an easy task [9]. Analysing the problem and tailoring a metaheuristic solver to it is the right approach, when possible. Similarly, tuning the parameters of the optimisation algorithm plays an important role. This can be a time-consuming task, especially when using modern algorithms, which are often based on a hybrid structure [10], and thus have even more parameters to adjust [11]. The latter are usually difficult to implement (which makes them more susceptible to errors) and understand, with some operators not contributing to the final performance on many problems [12] - simplifying them would at least reduce their algorithmic overhead.

In this light, many modern metaheuristics are not suitable or thought for optimisation on severely constrained devices, as previously discussed. Memory limitations of the environment hosting them and minimising their computational overhead are not factors that are usually taken into consideration during the development phase. However, there are application domains where such devices are required to be equipped with a quick and simple optimisation routine; e.g., in the Internet of Things (IoT), where cost is usually an issue [13], or in the manufacturing sector, where fast, smaller and energy-efficient systems are a priority. As the current availability does not seem to stop, with the most effective processing Artificial Intelligence (AI) technology having thousands and thousands of parameters to tune, we argue that minimising algorithmic overhead and memory consumption in optimisation algorithms would be a priority in several constructs in the years to come.

Most metaheuristics in the literature are population-based algorithms operating on a set of candidate solutions, a framework that has been shown to have some benefits [14]. However, single-solution algorithms also exist, and the importance of memory consumption in the study of population-

[1] In the remainder of the paper we consider bound-constrained (also misleadingly referred to as "unconstrained") single-objective continuous optimisation problems of the form:

$$\begin{aligned} &\min f(\boldsymbol{x}), \\ &s.t.\ \boldsymbol{x} \in \mathcal{S} \end{aligned}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$ is a candidate solution to the D-dimensional optimisation problem defined through the objective function $f(\cdot)$ (without loss of generality, we assume this to be minimised), and $\mathcal{S}$ is the search space delimited by the upper bound vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_D)$ and the lower bound vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_D)$, s.t. $a_i \leq x_i \leq b_i \ \forall i \in \{1, 2, \ldots, D\}$.

based metaheuristics has been addressed in some recent studies. In [15], the authors adjust the implementation of three different Genetic Algorithms (GA) to embed them in an ATmega328P microcontroller. In their experiments, with 128 individuals represented in 32 bits, approximately 80% of the available data memory (2KB) was consumed. This leaves no room for other background/parallel processes. Execution time can also be problematic, as shown in [16]. Here, evolutionary and swarm computing algorithms are integrated and run on multiple embedded systems, such as a smartphone and three different Raspberry Pi models (and on a PC to use as a baseline for comparisons), on 10 well-known benchmark functions with $D = 10$ and population size $N_P = 10$. The execution time increases significantly in embedded systems and, worryingly, in the smartphone, where each algorithm is at least 10 times slower than on the PC. This shows that an ad hoc algorithm/implementation should be used in such a device to keep execution time realistic. Other authors experimented with other hardware technologies, such as Field Programmable Gate Arrays (FPGAs) [17–19] or Graphical Processing Units (GPUs). For the sake of completeness, it is worth adding that these concepts can be extended to other optimisation scenarios, such as, e.g., multi-objective problems. In this regard, we point to [20] where a Multi-Objective Genetic Algorithm (MOGA) is implemented and executed on a low-end microcontroller.

As a summary of what has been previously discussed, there are important domains where optimisation algorithms that: 1. can attain good solutions with much less memory use, and 2. can be easily embedded into limited hardware platforms. In the remainder of this article, we will refer to these algorithms as "lightweight metaheuristics" [21], or "memory-saving algorithms" [22].

Given that the nature of the problem imposes the number of "design variables" x_i ($i = 1,2,\ldots,D$), reducing the need to store a population of solutions is the main goal of a memory-saving algorithm. We assume that the solutions are represented correctly, i.e., without unnecessary long encodings or redundant design variables. Also, we are selecting simple algorithms with linear $O(D)$ memory complexity like genetic algorithms, as opposed to those requiring memory and computationally more expensive features such as eigenvalue decomposition, manipulation of covariance or Hessian matrices, etc., to function, see e.g., [23–26]. The degenerate case $N_P = 1$ results in a "single-solution metaheuristic" (also referred to as "trajectory methods", "solo search", or "single-agent-based algorithms"). Here, we must pay attention to the working logic of the algorithm. Despite being less common, memory-consuming single-solution algorithms do exist. Examples such as the Rosenbrock algorithm [23], the Powell method [24], and SPAM [27], make use of $D \times D$ matrices stored in memory to perturb the only candidate solution on which they operate, making it difficult to use in memory-constrained environments. In coherence with the algorithms mentioned above, the Nelder-Mead method (also known as the simplex method) [28] was introduced as a derivative-free optimisation algorithm. It starts with an initial solution and iteratively uses a set of solutions forming the vertices of a simplex to move it. This method teeters on the brink of two opposing perspectives, as it may be thought of as a single-solution approach, yet it cannot be considered lightweight in our case because a simplex of $D+1$ points is required for it to function. A similar approach can be taken with Estimation of Distribution Algorithms (EDAs) [29, 30], which evolve a probabilistic model and draw solutions from it. In this case, sampling only one solution is not enough if a memory-saving probabilistic model is not used. This model can be a simplification of existing models to perform a so-called uncorrelated search that does not require storing cross-correlation values between the design variables. The compact algorithm class [31] is an established framework for obtaining memory-saving EDAs.

If properly designed, single-solution and compact algorithms can perform well and return sat-

isfactory results in several contexts that require a very low *memory footprint*. These scenarios are abundant in some application fields including bioinformatics [32–34], deep learning [35, 36], evolving hardware [37] and robotics [38, 39].

The goal of this article is to present a unified survey of research in the field of lightweight algorithms, as the existing literature appears inadequate to offer a comprehensive perspective on this class of optimisers. Our work sheds light on what is currently available for dealing with optimisation problems in an environment plagued by various limitations and provides readers with a wide range of application domains. This will benefit both practitioners and algorithm designers exploring hybrid algorithmic solutions. Indeed, it is clear that most of these algorithms are currently not as well known as population-based ones, and it is not rare to encounter statements such as *"To the best of our knowledge, SA* [Simulated Annealing] *, VNS* [Variable Neighbourhood Search] *and TS* [Tabu Search] *. . . are the only existing single-solution metaheuristics in the literature"* [40], suggesting that advances in this field are somehow ignored. Hence, here we combine relevant research lines and place greater emphasis on approaches such as the compact algorithm paradigm in [31] and holistic analysis in [6, 41] to overcome these problems. We gather relevant literature and provide interesting perspectives on modern and historical lightweight heuristics, reporting key notions that give a global view of these algorithms, including the current state of the art that is not included in [6, 31, 41] and is becoming fragmented. Furthermore, we review and report some significant applications of these algorithms, giving examples to practitioners having to deal with these scenarios, and facilitating the search for reactive algorithmic solutions that are already present in the literature in one document. For benchmarking and other performance-related numerical results, we refer to [42–44], and most of the articles included in this survey.

The remainder of this paper is organised as follows.

- Section 2 describes classes of algorithms based on the number of processed candidate solutions and introduces the concept of "lightweight" algorithms for systems having limited resources.
- Section 3 focusses on population-based algorithms and discusses the use of micro-populations.
- Section 4 introduces the Estimation of Distribution Algorithms and discusses their memory requirements.
- Section 5 surveys the existing literature to report and comment on memory-saving algorithms (for both discrete and continuous optimisation) by grouping them into the two main categories of single-solution and compact algorithms.
- Section 6 reports relevant application scenarios.
- Section 7 concludes this work and discusses open issues in the field of lightweight optimisation research.
- Section 8, systematically points out areas of improvement to address in the future.

2. Metaheuristics in the balance

When the environment hosting, the optimiser requires a thrifty use of resources, even between algorithms with linear memory complexity there might be some that are preferable over others

that, on the contrary, require unwanted memory slots [2] to function. In this context, metaheuristics with linear memory footprint can be further classified by considering *number of solutions* stored in memory during the search for optima, as an indicator of the resources needed to run the optimisation process.

For the sake of clarity, we remark that this shall be done only for algorithms that are already "lightweight" in their nature, i.e. metaheuristics that do not require the storage of auxiliary variables for representing or manipulating the candidate solutions. (e.g., covariance matrices, etc.). These classes of algorithms are usually developed in an attempt to obtain high performance in offline problems, but in the context of real-time and onboard optimisation they are often *infeasible* choices and are considered, within the scope of this article, "heavyweight" algorithms as opposed to those with linear memory occupation with D. Note that, as previously discussed, these heavyworking mechanisms can take place in both population-based and single-solution algorithms. In this work, we go even further and carefully select truly lightweight algorithms from those having a linear memory footprint considering the number of memory slots required for them to operate, as graphically depicted in Figure 1.

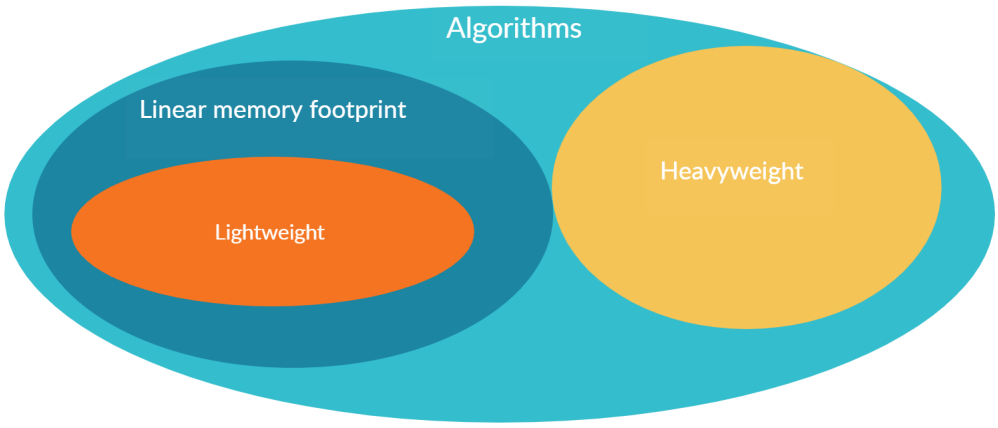

Figure 1: Metaheuristics in the balance: heavyweight and linear memory footprint.

It is important to note that lightweight algorithms consist of algorithms with approximately two memory slots (one best solution plus an additional auxiliary solution to produce a new solution). Approaches with this feature are from the previously introduced classes of single-solution and compact algorithms, which we refer to as "sMeta" and "cMeta" in the remainder of this paper for brevity. In line with this notation, we also use the expressions "pMeta" and "μMeta" for population-based algorithms and population-based algorithms working with so-called micro-populations of only a few individuals, respectively.

Note that we not only report algorithms that have a modest memory footprint, but also select the most successful design strategies that allow satisfactory performance despite using a small number of memory slots. As most computation algorithms mimic the behaviour of popular pMeta algorithms, for the sake of completeness, we next discuss key points on pMeta algorithms. This allows us to

[2] A vector of real values, for example, floating or doubles, of the same size D as the problem.

better introduce the μMeta algorithm surveyed in this work. These represent the simplest way to obtain a lightweight algorithm and can be found to exist in some memory-constrained environments.

3. Populations and μPopulations

Population-based algorithms have been the go-to solution for solving a large number of (constrained or unconstrained, single-objective or multi-objective) optimisation problems for many years now. These methods have been proven to be key to solving many real-world problems and are now being developed in continuous development. For more information on the many paradigms existing in the literature, the most established being Evolutionary Computing (EC), Swarm Intelligence (SI), and Hyperheurstics/Memetic Computing, we point to relevant books [6, 45–47] and surveys [48–54]. Modern hybrid structures employing machine learning components also populate the literature [55, 56]. Note that most surveys either focus on a specific algorithmic family or classify wide ranges of metaheuristics based on their inspiring metaphors. Such metaphors have been very useful to the research community, developing the first nature-inspired algorithms. However, there is a clear recent trend in designing optimisation heuristics simply by using an inspiring metaphor as the main driving force and motivation. This is generating a plethora of algorithms whose contributions to the field are arguable and that are often poorly benchmarked and compared to similar strategies that were already available in the literature. This is evident from recent metaheuristic surveys, many of which focus on such metaphor-led algorithms and their variants [57–60]. In this survey, we mention some of these algorithms, depending on the relevance of the message of the corresponding article, as it is important to survey the totality of the current literature. However, we recommend that one always check the literature to avoid reproposing similar ideas under different names, use a more theoretical or empirically informed approach, and follow good practises [61–63] when designing novel algorithms. Proper benchmarking should also be performed. In summary, we are in favour of making progress in algorithm design and using metaphors as a means of conveying complex information, but we share the same doubts/opinions of [10, 64–66]. Interestingly, there are surveys on the performance of a wide range of metaheuristics on specific artificial testbed problems and real-world scenarios [67–72]. These suggest practical insights on applying algorithms and highlight the importance of performing a thorough parameter tuning phase (and self-adaptive algorithms might also have some parameters to tune).

Adapting algorithm parameters to ensure optimal performance can be a challenging task [73], with the exception of the population size value. We now know that high values are not necessarily recommended, but in most cases the common belief from the literature seems to be that increasing this parameter is beneficial over noisy, highly multimodal, and large-scale problems. For example, the study in [74] suggests $N_P = 10 \cdot D$ for Differential Evolution (DE), which can be impractical for large-scale or real-world time-expensive problems. Moreover, this is not necessarily correct in all scenarios. Some DE variants with micro-populations of a maximum of five individuals have been shown to perform well on a very large-scale problem with thousands of design variables [75], where exploration is partial under the fixed computational budget, and the decision to focus more on exploitation seems to yield better results. Similar results are obtained with other micro-population evolutionary and swarm intelligence algorithms [76, 77].

3.1. Micropopulations

As a general rule, the pMeta algorithms with $5 \leq N_P \leq 20$ can be referred to as μMeta algorithms. However, 20 solutions are often considered too many and are not used when the full benefit of having

a small population size, that is, rapid convergence, is sought. Potentially, less than five solutions can be tested, but only if the working logic of the algorithms allows for it or is not damaged. For example, a classic DE with *rand* mutation uses 4 individuals (the *target* plus three randomly selected individuals chosen from the population) to generate a new *offspring* solution (see [54] for details on DE). Hence, less than four individuals is not a feasible setting, and exactly four individuals would mean that they will be always involved in all the perturbations. At least 5 or 6 of them are preferred to be able to implement the benefit of having a population while increasing exploitation and minimising memory usage. Note that DE is specifically suitable for working with micro-populations as diversity can still be high in the initial phases of the search process [78]. An analysis of the effect of DE and PSO (Particle Swarm Optimisation) micropopulations on various problems with different characteristics is available at [79].

The first investigations of micro-populations date back to the introduction of the micro-Genetic Algorithm (μGA) [80] and its variants [81, 82]. Since then, several micro-population Evolutionary Algorithms (μ EAs) followed, such as, e.g., [83]. After understanding the potential of a classic micro-Differential Evolution (μDE) over large-scale problems, several μDE schemes, self-adaptive variants (such as μ JADE), and hybrid memetic alterations were proposed [75, 78, 84–91]. Analogously, Swarm Intelligence algorithms have been shown to have similar advantages when run with micro-populations. The results worth mentioning are those obtained with micro-Particle Swarm Optimisation (μPSO) algorithms [76, 92, 93]. Further successful examples are those of micro-Artificial Immune System (μAIS) [94], micro-Bacterial Foraging Algorithm (μBFA) [95], and other metaphor-led algorithms such as those in [77, 96, 97] (which are indeed very similar to the more established framework such as DE and PSO, thus returning similar results). Finally, other important roles played by μEAs are to perform a local search within memetic algorithms [98], and to act as micro-algorithms for multi-objective optimisation [99, 100].

4. Estimation of Distribution Algorithms (EDAs)

The EDAs family forms a significant subset of EC algorithms where the concept of population plays a different role compared to other pMeta algorithms. This family is in continuous evolution and investigation, with frameworks such as Bayesian Optimisation (BO), also known as efficient global optimisation [101], currently finding its place in several time-consuming optimisation contexts, while originally simply referred to as Probabilistic Model-Building Genetic Algorithms (PMBGAs) [102]. This is because the first algorithms of this kind were a modification of previous EAs to drive the search through probabilistic models to achieve better performance on those non-separable problems characterised by high epistasis [103, 104], which are challenging for many ES and SI strategies.

An EDA builds and samples promising candidate solutions from an *explicit* probabilistic model (which implicitly represents the population). The optimisation process is then the iterative evolution/update of the model, usually starting with an exploratory distribution and ending with one generating (near) optimal solutions. Some EDAs draw populations from the corresponding distributions (μ individuals are sampled per iteration), while others need fewer or a candidate solution to be drawn (as in most compact algorithms). Over the years, many algorithms appeared based on different models of all ranges of complexity, such as Population-Based Incremental Learning (PBIL) [105], Mutual Information Maximising Input Clustering (MIMIC) [106], Bivariate Marginal Distribution Algorithm (BMDA) [107], Factorised Distribution Algorithm [108], and many others such as several Extended Compact Genetic Algorithms [109–114]. Multivariate factorisation is also a

widely used method and since some evolution strategies incorporate multivariate normal models, these can be seen as EDAs. Among them, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [25, 115] is considered by many as a great example of EDA algorithm, see [116], which became soon a benchmark for optimisation due to its performances and properties, such as invariance to rotations of the problems and many other peculiarities which, according to the authors, do not necessarily match the key characteristic of a "pure" EDA. Indeed, CMA-ES estimates the distribution of expected steps, while EDAs model them, etc. (see [115] for more details). However, the CMA mechanism is based on a complex probabilistic model with time complexity $O(D^3)$ and memory complexity $O(D^2)$, which can be seen as a heavyweight computational requirement, as it can be relaxed by assuming that there is no cross-correlation (as done in compact optimisation). The resulting covariance matrix will prevent the normal distribution from rotating, while still allowing for adaptation along each coordinate axis if D variances are evolved (one per axis). Instead, if the same constant variance value - which acts as a sort of step size - is kept the same along all axes, one obtains a symmetric normal distribution, which can only move within the square space when its mean value gets updated. The latter is the simplest case, which is also less memory-expensive.

For an overview of the many models available for EDAs, we refer to [116–118]. Note that models and update rules for control parameters, such as, for example, variances, mean values, or other measures of central tendency and spread, can be very complex, with the simplest being those proposed for compact optimisation (details in Section 3.1).

5. Lightweight metaheuristics: a taxonomy

We propose a lightweight metaheuristic taxonomy structured as in Figure 2. This gives a graphical overview of the two main classes of algorithms we survey in this work, i.e., sMeta and cMeta, and offers a granularity level, further classifying relevant subfamilies and variants of the same framework. Milestone algorithms and their more recent variations are considered in the taxonomy, as well as a few examples of modern metaphor-led algorithms to comment on current practises. While doing this, we summarise their working logic and report relevant successful applications and application domains for such algorithms.

5.1. Single-solution optimisation algorithms for combinatorial problems

Hill Climbing (HC) [162], a.k.a. Iterative Descent, is a basic local search algorithm. Starting from an initial point, incremental perturbations are applied iteratively to enhance the value of the cost function. There are four main HC variants, namely the *iterative best improvement* method, the *iterative first improvement* method, the *randomised iterative improvement* method, and the *probabilistic iterative improvement* method; see [163] for details. Note that the first two strategies are greedy, while the others accept worsening moves, that is, candidate solutions with a worse objective function value than the current one. Applications of HC are abundant in the literature. An example of a timetabling problem is in [164].

Iterated Local Search (ILS), presented in [165], is a simple multi-start method that iteratively performs a *perturbation* step to explore a new starting point to then perform the *local search* step. When starting, the initial point must be provided or generated randomly, thus not requiring *perturbation*, and *local search* is immediately applied. These two iterated phases can be seen as alternating exploratory and exploitative searches. ILS is used successfully in other scheduling problems, such as the University Course Timetabling Problem [166]. Some notable ILS alterations are the hybrid adaptive ILS with Path-Relinking designed to solve the capacitated vehicle routing problem in [167],

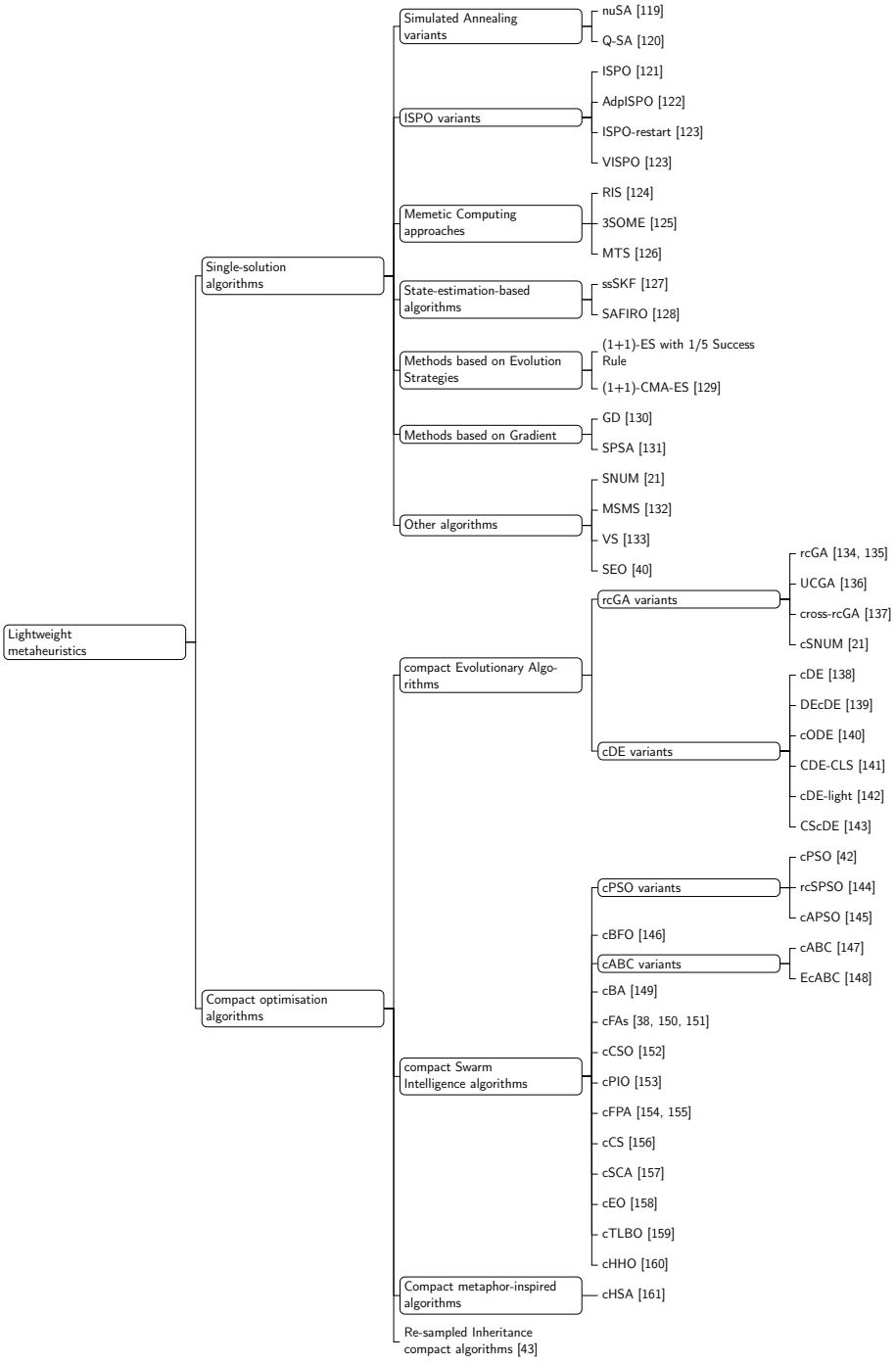

Figure 2: Overall taxonomy of lightweight metaheuristics for continuous optimisation.

the ILS with ejection chains for open vehicle routing problems with time windows [168], and the two-phase ILS for the Set-Union Knapsack Problem [169]. For a detailed review of this algorithm, we refer the reader to [170].

Breakout Local Search (BLS) [171] is a variant of ILS that merges the steepest descent local

search with adaptive perturbation strategies. BLS dynamically adjusts diversification by varying perturbation moves and types based on search history information. BLS is used successfully to address the Vertex Separator Problem (VSP) [171], the quadratic assignment problem [172], the maximum clique problem [173], and to solve the Assembly Sequence Planning Problem [174]. The hybrid BLS algorithm based on reinforcement learning from [175] shows improved performance over VSP.

Large Neighbourhood Search (LNS) [176] is a metaheuristic in which the two operators *repair* and *destroy* alternate to obtain a new solution in a neighbourhood of the candidate solution. The destroy operator is responsible for perturbing random components of the candidate solution, which then undergoes a feasibility check where the repair operator fixes the components to ensure that the new solution is in the socket space. Adaptive LNS algorithms for Vehicle Routing Problems (VRPs) can be found in [177, 178], and a hybrid adaptive LNS for the large-scale heterogeneous container loading problem is proposed in [179]. These are amongst the latest techniques proposed for these kinds of scheduling problems.

Great Deluge (GD) [180] operates by setting a threshold that acts as an upper limit for the admissible values of the objective function of the newly generated solution. Whenever the new candidate solution is accepted (i.e., in an imitation context, it must have a function value inferior to the upper limit), the upper limit value is decreased according to the adopted decay rate. In this way, most points are accepted at the beginning, but the algorithm becomes more selective after interaction. GD for a real-world examination timetabling problem [181]. In [182] an adaptive version is proposed, called the Flex-Deluge algorithm, to solve the timetabling problems of university exams. Other hybrid variants also populate the literature, such as those in [183] and [184], which are proposed to address VRP and task scheduling in grid computing, respectively.

Variable Neighbourhood Search (VNS) [185] is based on the idea of systematically changing the neighbourhood. This occurs in two phases, in a *local search phase* which chooses the best neighbour improving the current solution to find the local optima, and in a *shaking phase* to escape from the corresponding valley. Details on its extensions and applications in which VNS has proven to be very successful can be found in [186].

Greedy Randomised Adaptive Search (GRASP) [187] is an iterative process that combines a construction heuristic step with a sequential local search step. In the first step, a feasible solution is created using a randomised greedy heuristic. This solution serves as the starting point for the local search, which can be either a descent local search or a more advanced method. The best solution found is returned after the search process. Variations of GRASP and its applications are discussed in [188].

Similarly to the other algorithm, Guided Local Search (GLS) is built on top of the LS technique. To use GLS, one must first define a suitable set of features for the problem. Each feature has a cost and a penalty assigned by GLS. When the LS gets stuck at a local optimum, some features are selected and penalised. More details can be found in [189]. The authors in [190, 191] provide a list of GLS variants/extensions, guidelines on how to use this algorithm in practical applications, along with a variety of problems in which it was applied.

Descent-Based Local Search (DB-LS) [192] moves from the current solution to a neighbouring one according to a given neighbourhood structure in such a way that each movement leads to a better solution. This iterative process continues until no improvement is found, in which case the current solution corresponds to a local optimum. This technique was combined with the reinforcement learning technique and applied to graph colouring [192].

The Tabu Search (TS) method was used in a nontrivial number of combinatorial optimisation problems; see [193]. It was first presented in [194]. The TS algorithm explicitly leverages the history of the search not only to escape local optima but also to implement an exploration strategy. TS, like simulated annealing, allows for lower-quality solutions when a local optimum is discovered. A detailed presentation of this method and its fundamental concepts can be found in [193].

Simulated Annealing (SA) was proposed in [195]. SA accepts non-improvement solutions in order to increase the chance of exploring the search space and escape from local optima. The algorithm starts with an initial solution and generates a random neighbour using a predefined neighbourhood structure at each iteration. If the newly generated solution is better than the best, it is accepted; otherwise, a solution of poor quality is accepted with a probability specified by the Boltzmann distribution. In particular, although SA was introduced for combinatorial optimisation, it has also been used to tackle real-valued problems. An exhaustive review of the literature is provided in [196].

Threshold Accepting (TA) [197] follows the same principle as SA, but differs in the criterion used to accept candidate solutions. SA allows a non-improving solution only with a given probability, whereas TA accepts it if the degradation does not reach a progressively decreasing threshold.

All of these search methods have a similar structure. In addition, each has its own mechanism to diversify the exploration of the search space by escaping local optima. Other local searches are detailed in [6] and [41].

Finally, hyperheuristics are prominent in dealing with discrete optimisation and are widely adopted for combinatorial problems. As first described in [198], these can be seen as "heuristics for choosing heuristics". The selection method is arbitrary but often consists of using a learning mechanism to optimally activate the right operator. Therefore, once a set of heuristics/metaheuristics is provided, hyperheuristics work on the low-level operator space other than the solution space and can be defined in [199] as "a search method or learning mechanism to select or generate heuristics to solve computational search problems". For details on milestone methods and recent advances, the relevant sources are [200–202], from which it can be seen that this optimisation paradigm is highly recommended to solve scheduling, timetabling, and other discrete problems. Comprehensive lists of successful hyperheuristic applications are available in [200, 203]. Furthermore, one can see that most of the low-level operators involved in this framework are sMeta algorithms. This makes this framework very suitable for designing efficient single-solution memory-saving algorithms. Most importantly, single-solution optimisation plays a key role in designing hyperheuristics (even population-based ones), as the use of local searchers based, e.g., on hill climbing methods is very frequent [201, 202].

5.2. Single-solution optimisation algorithms for continuous optimisation

When it comes to metaheuristics, most people immediately think of pMeta algorithms because using a set of multiple candidate solutions, i.e., the so-called population, is currently a stable practice. This is particularly true in the continuous domain, where manipulating a population of points is always seen as beneficial, see, e.g., [14], while sMeta algorithms are considered to result in poorer performances [204]. However, in line with [9], this cannot always be the case, and some sMeta, such as Simultaneous Perturbation Stochastic Approximation (SPSA) methods [131], offered ideas and played an important role in applied sciences such as physics and engineering in the past. Other algorithms, obtained as degenerate variants of existing pMeta algorithms with some adjustments to have $N_P = 1$, also provided interesting results. Other ideas, such as memetic single solution algorithms and hyperheuristics for the continuous domain, displayed highly competitive performances. We report on all of these classes of algorithms, including relevant historical and modern methods.

We remark that sMeta is not a synonym for simplicity or minimal material consumption. Let us consider the elegant single solution evolution algorithm Cholesky (1+1)-CMA-ES [129], which reduces the computational effort of CMA-ES from $O(D^3)$ to $O(D^2)$ and requires only one candidate solution (plus an additional one for a temporary new solution). Its working mechanism is theoretically sound and consists of manipulating a matrix $D \times D$, thus not belonging to the list of lightweight algorithms provided.

In the continuous domain, it is possible to take advantage of the notion of a gradient to guide the search. Gradient Descent (GD) [130] is a first-order single solution-based method that relies on the objective function differentiability for its proper functioning. Depending on the function to be optimised, it iteratively adjusts the current solution, moving it in the direction of the steepest ascent or descent. In the SPSA algorithm previously mentioned, this is done by approximating the classic finite-difference gradient methods stochastically. As in this case, if Hessian matrices are not required, the method can be quite efficient, in particular, if the problem is not highly multimodal, despite being memory-saving. Other methods, such as those deriving from the classic Hooke-Jeeves direct search method [205], do this indirectly by looking at objective function values in opposite orientated directions on the same line, per component/axis (we present moderate variations in the remainder of this section). This also resembles a continuous counterpart logic to the methods seen for the discrete domain based on neighbouring operators. In this case, the neighbourhood is obtained with either a fixed or an adaptive exploratory radius from the candidate solution. Solis and Wets [206] present a very simple randomised search of this kind.

Non-Uniform Simulated Annealing (nuSA) [119] is an improved version of SA for continuous optimisation. It uses a non-uniform mutation which gradually shrinks neighbourhood size during the search. Quaternion SA (Q-SA) [120], instead uses a quaternion representation of candidate solutions to improve neighbourhood exploration and prevent premature convergence by widening the initial search space. Q-SA explores the quaternion space rather than the Euclidean space and does not employ specific parameters to alter the neighbourhood range. Finally, the Single Non-Uniform Mutation-based (SNUM) algorithm [21] is a simplification of the non-uniform mutation strategy of nuSA. SNUM has only one parameter, which makes it quite easy to use. Its performance does not depend significantly on the value of this parameter.

There are also historical evolutionary sMeta algorithms specifically designed for the continuous domain. A worth mentioning one is the (1+1)-Evolution Strategy with 1/5 Success Rule [207], which decreases the standard deviation of the extended normal perturbation if the number of successful mutations is less than 1/5. Other methods from the EC and SI families are described in the following.

Intelligence Single Particle Optimiser (ISPO) [121], and its first formulation, referred to as IPO in [208], are simple sMeta variants of the Particle Swarm Optimisation (PSO) metaheuristic [209]. Its working logic operates per component, i.e., each design variable is perturbed a pre-fixed number of times sequentially to complete one iteration. Therefore, it is suitable for separable problems. ISPO adjusts the velocity vector depending on a learning factor based on the number of successful updates of the particle during the search. Four parameters are required in total, but performances depend mainly on two (namely the diversity factor and the descend factor), which are the only problem-dependent parameters according to [121, 208]. As tuning them can be challenging, the AdpISPO algorithm [122] is designed as a self-adaptive version of ISPO whose adaption logic has been shown to work well in many testbed problems. This outperforms ISPO on such problems and is free of problem-dependent parameters. This feature makes it suitable for hybridisation with other algorithms whose implementations depend on the parameter setting. Examples of Memetic Computing

(MC) algorithms that use PSO variants and AdpISPO to perform local computations are available in [210–212]. To improve performance, in particular, over multimodal and large-scale domains, two multi-start variants called ISPO-Restart and Very Intelligent Simple Particle Optimiser (VISPO) are presented in [123]. Both ISPO-Restart and VISPO perform a "jump" in the search space before restarting the search, but the newly generated starting point is first uniformly sampled within the domain and then mixed with the "elite" solution stored in memory by inheriting some of its promising design variables. This inheritance is obtained by binomial crossover (from DE [54]) with a low crossover rate to preserve randomness at the new starting point. ISPO-Restart and VISPO perform similarly, with VISPO being preferable to only a few benchmark problems tested in [123]. The only difference between the two is that the first variant restarts after a number of prefixed functional calls, while the second has a simple learning mechanism based on the number of successful continuous intervals per dimension to automatically decide when to restart.

Multiple Trajectory Search (MTS) [126] is another interesting lightweight sMeta that performs well on separable and large-scale problems (for which it was designed specifically). It can be seen as the coordination of three iterated local searchers algorithms where the first one perturbs all directions one at a time along one axis (as the Hook-Jeeves local searcher operator), the second one differs from the first one in that only searches one-fourth of the available dimensions, and the third one takes three small steps along each dimension according to find a candidate solution heuristically. The three operators are activated according to a grading system based on their successes, and if no improvement is registered, the search range is cut to one-half. The Multiple-Search Multi-Start (MSMS) framework in [132] is based on the simple idea of implementing a few search algorithms, but features a multi-start operator to keep changing the initial point in an attempt to approach premature convergence.

Three-Stage Optimal Meta-memetic Exploration (3SOME) [125] is a simple MC technique characterised by the activation of three operators (memes) that perturb a single solution. These three components, namely long (L), middle (M), and short (S) distance exploration, are arranged in a bottom-up structure and coordinated in such a way that the exploitation pressure increases as the algorithm converges to a promising area of the search space. Most of the calls to objective functions are used for the local search operator S [213], which implements the same strategy as the first local search of MTS. The coordination logic of the three memes is very simple but allows for competitive results when compared to established algorithms, including pMeta algorithms, and in particular on separable problems. Following these results, several variants have been proposed. An improved M operator dynamically narrowing the space around the solution in the attempt to provide a better quality starting point to S, is available at [214], and many modifications (not all necessarily preserving the memory-saving nature of 3SOME) proposed in [215] allow for handling non-separable problems/rotated problems. The analysis in [213] highlighted the importance of the coordination logic in the operator implementation itself and identified the least activated operations (and the most expensive in terms of the calls of the objective function) during several optimisation processes. This led to simplified variants, resulting in significantly different algorithms with operators making fewer functional calls before local refinement. A very simple one, having only two stages, is the Re-sampling Search algorithm [216] - which can be seen as a sort of multi-start ILS algorithm for continuous optimisation - which was subsequently improved in the Resampled Inheritance Search (RIS) [124] framework. Further variants have been designed to deal with specific real-world applications; see, e.g., [217].

RIS [124] is a simple MC approach that performs a restarted iterated local search with a low

level of inheritance of the previous best solution design variables after each restart. The S operator is run multiple times until a condition on the length of its exploratory radius is met (the option of fixing the number of S steps is also left available to the user). When a restart occurs, a point is drawn uniformly within the search space, and some of its design variables are crossed over to retain promising elite components. Both binomial and exponential DE crossover strategies are tested, see [54] for details, with exponential being the default choice. RIS is simple yet effective and competes (often outperforms) pMeta algorithms on several benchmark functions. It can be seen as an optimisation framework in which an algorithm, such as cross-over strategy and local search, can be replaced with more appropriate combinations depending on the problem if necessary. To obtain a more robust framework, the Parallel Memetic Structure (PMS) [218] followed as a general idea of having multiple searches performing complementing perturbations, thus increasing the diversity of possible moves within the search space. PMS maintains the restart mechanism with inheritance and runs two local searchers moving along the axis (S is used for this purpose) and diagonally in the search space (Rosenbrock is used). This framework was proposed with the idea of including an adaptation system that allocates more budget to the local community performing the most successful move dynamically during the search, see, e.g., [27, 219]. Note that the original implementation of PMS executes Rosenbrock to perform the diagonal move. Obviously, this adds a quadratic memory footprint to the algorithm as a whole. To have a memory-saving variant of PMS, this meme has to be replaced with a lightweight sMeta.

To reduce the number of parameters of the Simulated Kalman Filter (SKF) algorithm, the work in [127] proposes a single-solution SKF (ssSKF) version using only one agent. The working mechanism of ssSKF does not differ significantly from that of SKF, which goes through the three steps of prediction, measurement, and estimation. As these are performed by a single agent, working with a single solution, ssSKF is lightweight and easier to tune, which is an important aspect given the impact of setting parameters on SKF [220]. A similar idea led to the Single-Agent Finite Impulse Response Optimiser (SAFIRO) [128]. In this case, an agent is also responsible for measuring and estimating the optimal solution.

Several other nature-inspired sMeta algorithms have been proposed over the years for continuous optimisation. However, most methods proposed in the last years are not trying to approximate gradient or exploit any specific feature of the problem. Some examples such as the Social Engineering Optimiser (SEO) [40], the Vortex Search algorithm (VS) [133], and the Simulated Raindrop Algorithm (SRA) [221], are based on inspiring metaphors that are implemented heuristically.

5.3. Compact optimisation

The majority of the algorithms in this survey fall into this class. First, we comment on the methodology used to review the reported articles. Subsequently, we provide a comprehensive description of the compact optimisation literature.

5.3.1. Methodology & research questions

In scholarly writing within a specific domain, authors conventionally incorporate prevalent terminologies of that field into their research paper titles and keywords. This practice improves visibility and discoverability among a broad readership. When conducting a survey, employing the same keywords is intuitive for paper retrieval. Nevertheless, this approach may lack precision in the selection of pertinent papers. Hence, it is imperative to adopt a systematic methodology for the acquisition and meticulous selection of relevant literature.

After reviewing the Centre for Reviews and Dissemination (CRD) guidelines proposed in [222], we feel like we adhered sincerely and inadvertently to almost the same step-by-step procedure when compiling the research to ensure rigour and foster comprehensiveness. This approach facilitated the identification and evaluation of candidate publications, thus improving the quality and reliability of the survey results and their implications within the scientific community. There are also other guidelines for performing a bibliometric analysis, such as the one proposed in [223]. It elucidates the distinctions between systematic and bibliometric surveys, outlining the specific scenarios in which each method is applicable. This falls outside of the scope of this work; please refer to the previous reference for more details.

The survey we propose is motivated by the following Research Questions (RQs).

RQ1*: What characteristics of an algorithm can be used as useful classification criteria?*. In recent times there has been a strong emphasis on classifying heuristics based on their inspiring metaphor, but we argue that other characteristics, as we picked the number of processed solutions, are more useful in practice. This RQ is addressed in Sections 2, 3, 4 and 5.

RQ2*: How many kinds of "lightweight" algorithms are available in the literature?*. Answering this question is useful to show practitioners what a memory-saving algorithm is for which application domain a specific class of these algorithms can be used. This RQ is addressed in Sections 1 and 2.

RQ3*: What are the main characteristics of lightweight algorithms?*. We report a mathematical and algorithmic description of the general framework to use for obtaining most of the existing memory-saving variants. This RQ is addressed in Section 5.

RQ4*: What are the application domains of memory saving optimisation?*. There are numerous and obvious domains where these algorithms can be used, but we dig deeper and indicate specific cases to show which strategies and variants have been more successful. This RQ is addressed in Section 6.

RQ5*: What potential future impact can lightweight algorithms have?*. There are open challenges to face, for example, in combining machine/reinforcement learning, where the use of fats and memory-saving algorithms can be preferred. This RQ is addressed in Sections 7 and 8.

The primary keywords used to review the literature and address the RQs are "memory-saving metaheuristics", "lightweight metaheuristics", "single solution metaheuristics", "compact metaheuristics", and all combinations where "metaheuristics" are replaced by "optimisation" or "algorithms". The results are refined by incorporating the names of the seminal algorithms, e.g., "cGA" and "rcGA", in the search, as well as relevant authors such as, e.g., "Harik" or "Minnino". It is highly improbable for a new paper on compact optimisation (discrete or continuous) to be published without referencing any of these influential works. This is known as the forward snowball technique to expand the search. We then looked at the references in these papers to find other relevant pieces of research. the latter approach is known as backward snowballing.

We produced a comprehensive list of articles published in the proceedings of established conferences and prestigious publishers. This list includes some old milestone methods and several recently proposed algorithms. Each article was evaluated for alignment with the research questions of this survey, leading to its inclusion or exclusion in the article. In particular, the participation of a single researcher in this process can introduce bias, oversight, and inconsistencies. To address these

concerns, the validity of the study was protected through a secondary review conducted by a second author. Although predefining the survey scope and managing paper selection could appear subjective, all conclusions and recommendations stem from the latest insights to ensure clarity of the scope of the paper scope and effective communication of its core message.

Toward the inclusion and exclusion criteria that guided the selection process in accurately categorising lightweight algorithms, we set the following ones. Papers published in languages other than English were excluded. The full text of the paper was scanned in many cases, not just the title and/or abstract. That is because the latter was not enough since it is not rare to find a paper that talks about compact algorithm/method/approach, but after reading the paper, we realise that it is about something different from the definition of "compactness" adopted in this survey. Similarly, algorithms that operate on single solutions may not necessarily be memory savings. In addition, duplicate instances of the retrieved papers were omitted, as well as other papers that have been published at a conference (with limited experimental setup), and then an extended version appears in a journal paper (for example, with cTLBO and cBAT). Lastly, lightweight algorithms hybridised with other components that lead to improvement but do not preserve the concept of lightweight were also removed or ultimately retained due to some considerations. Another case appeared concerning a particular algorithm, the Mean-Variance Mapping Optimisation (MVMO), in which we had different points of view: it was considered by one author as sMeta but not by another. Since we encountered difficulty in making a conclusive decision, we decided to exclude it from the taxonomy. This thorough review filtered out and pruned down the number of articles to a more manageable one. The resulting compilation was deemed representative of presenting a diverse range of lightweight algorithms serving as the basis for the proposed taxonomy.

5.3.2. Compact optimisation algorithms

Compact algorithms are among the simplest expressions of EDA algorithms, see Section 4, thus featuring fewer memory and computationally onerous algorithmic structure. For this reason, they have become popular since the publication of the compact Genetic Algorithm (cGA) [224], which was shown to perform similarly to the popular Simple GA with uniform crossover over discrete domains.

After cGA, counterparts for the continuous domain appeared in [134] and [135], where the real compact Genetic Algorithm (rcGA) is presented. This concept was then expanded to obtain other EC approaches, such as compact DE (cDE) [138], compact PSO (cPSO) [42] and, in a more general sense, a compact optimisation framework [31]. A general template that illustrates the structure of a compact algorithm is depicted in Algorithm 1.

5.4. Probabilistic models

The binary and Gaussian probabilistic models in [224] and [135] are the most widely used in compact optimisation depending on the discrete or continuous/real-valued nature of the search space. Taking into account the original notation in [224], compact algorithms require a "Probability Vector" **PV** to probability values. However, this is true for the original binary case. In the continuous domain, **PV** has been kept as a legacy variable, but contains (Gaussian) distribution parameters (mean values and standard deviation), thus being a two-dimensional array. Note that a "virtual" population size N_P has to be indicated. This is used within the module to mimic the coverage behaviour that larger or smaller populations have in the search space.

Note that other models have been proposed in the literature. We describe them in the following subsections.

Algorithm 1 High-level template of compact optimisation algorithms.

input: probabilistic model $\boldsymbol{PV}$, problem size D
output: best solution $\boldsymbol{elite}$
sample $\boldsymbol{elite}$ by means of $\boldsymbol{PV}$
while termination criterion is not met **do**
  sample a candidate solution $\boldsymbol{x}$ by means of $\boldsymbol{PV}$
  compare fitness of $\boldsymbol{elite}$ and $\boldsymbol{x}$
  update $\boldsymbol{PV}$
  if $\boldsymbol{elite}$ condition replacement is satisfied **then**
    $\boldsymbol{elite} \leftarrow \boldsymbol{x}$
  end if
end while

5.4.1. Binary model

In this model, $\boldsymbol{PV}$ is a vector of length equal to the dimension of problem D, where each i^{th} element is the probability of sampling 1 for that design variable. All elements of $\boldsymbol{PV}$ are initialised with a value of 0.5 to have an initial uniformly distributed solution. Relevant examples of algorithms using this model are, e.g., [37, 225–229].

5.4.2. Gaussian model

In the Gaussian model $\boldsymbol{PV}$ is the $2\times D$ matrix $\boldsymbol{PV} = [\boldsymbol{\mu}, \boldsymbol{\sigma}]$ where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation vectors of an uncorrelated and truncated Gaussian Probability Distribution Function (PDF) [135] with domain $[-1,+1]$. Mathematically, this is formulated as in Equation 1,

$$\mathrm{PDF}(x_i,\mu_i,\sigma_i) = \frac{e^{\left(\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right)}\sqrt{\frac{2}{\pi}}}{\sigma_i\left(\mathrm{erf}\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right)-\mathrm{erf}\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)\right),} \tag{1}$$

where μ_i and σ_i are the mean and standard deviation along the i^{th} asxis, and $erf(\cdot)$ is the error function.

When the algorithm is initialised, these parameters are initialised so that $\mu_i = 0$ and $\sigma_i = \lambda$ ($i = 1,2,3,\ldots,D$) where λ is a positive constant (usually $\lambda = 10$) large enough to approximate a uniform distribution in $[-1,1]$.

For the Cumulative Distribution Function (CDF), this is formulated in Equation 2.

$$CDF(x_i,\mu_i,\sigma_i) = \int_{-1}^{1} PDF(x_i,\mu_i,\sigma_i)dx_i. \tag{2}$$

To compute this CDF, which does not have a closed analytical expression, it is approximated using Chebyshev polynomials [230] to be used within the algorithm.

Sampling

The polynomial approximation of the CDF is used to generate solutions within the search space. To do this, a uniform random number $r = rand(0,1)$ is first generated.

Subsequently, the inverse function of the CDF of each i^{th} axis must be computed to be evaluated at r. This returns the value of the i^{th} design variable that forms the new candidate solution $x_i = CDF^{-1}(r,\mu_i,\sigma_i) \in [-1,1]$.

Note that this model operates in the $[-1,1]$ domain. Therefore, when dealing with a generic $[a_i,b_i]$ domain, the obtained value of x_i has to be scaled back to the original decision space simply by performing $x'_i = \frac{(x_i+1)}{2}(b_i-a_i)+a_i$. Conversely, this also means that before feeding a solution to the algorithm, one should normalise it within $[-1,1]$ - unless that is the domain of the original problem.

Selection and update

When the newly generated individual competes with the current individual, the fittest (i.e., the one with a lower objective function value in a minimisation context) is referred to as the **winner**, while the other is declared as **loser**. The winner influences $\boldsymbol{PV}$ as its design variables are used in the update rule of both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ as shown in Equations 3 and 4, respectively, where j indicates the iteration counter and $winner_i$ and $loser_i$ are the i^{th} components of **winner** and **looser**, respectively.

$$\mu_i^{j+1} = \mu_i^{it} + \frac{1}{N_P}(winner_i - loser_i) \tag{3}$$

$$\sigma_i^{j+1} = \sqrt{(\sigma_i^{it})^2 + (\mu_i^{it})^2 - (\mu_i^{j+1})^2 + \frac{1}{N_P}(winner_i^2 - loser_i^2)} \tag{4}$$

For more details, see [135].

Algorithms using this model are, e.g., the real-valued compact algorithms rcGA [135], cDE [138], cPSO [42], cABC [147], etc. - see Section 5.6 for more details.

5.4.3. Enhanced Gaussian model

An improved model using two PDFs that share the same parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ is proposed in [148].

When this model is used, the normalised search space is seen as $\bigtimes_{i=1}^{D}[-1,\mu_i]\cup[\mu_i,+1]$, and two PDFs, namely $Q-PDF(x_i)$ and $R-PDF(x_i)$, are defined as

$$Q-PDF(x_i,\mu_i,\sigma_i) = \frac{-\sqrt{\frac{2}{\pi}}}{\sigma_i erf(\frac{\mu_i+1}{\sqrt{2}\sigma_i})} e^{(-\frac{(x_i-\mu_i)^2}{2\sigma_i^2})} \quad \text{for} \quad -1 \le x_i \le \mu_i, \tag{5}$$

$$R-PDF(x_i,\mu_i,\sigma_i) = \frac{-\sqrt{\frac{2}{\pi}}}{\sigma_i erf(\frac{\mu_i-1}{\sqrt{2}\sigma_i})} e^{(-\frac{(x_i-\mu_i)^2}{2\sigma_i^2})} \quad \text{for} \quad \mu_i \le x_i \le 1. \tag{6}$$

Algorithm 2 shows the sampling mechanism, where the parameter ξ controls the probability of employing Equation 5 other than Equation 6.

If ξ is set to 0.5, then this model is expected to perform as the original Gaussian model [148]. Recent studies using this model reported, such as [148, 149, 154].

5.4.4. Uniform model

The work in [231] proposes a model based on the Uniform PDFs defined below in Equation 7:

$$U-PDF(x_i) = \begin{cases} \frac{1}{b_i-a_i} & \text{if } x_i \in [a_i,b_i] \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Algorithm 2 Sampling mechanism used with the enhanced Gaussian model.

input: the vectors $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ and parameter ξ
output: a new trial solution $\boldsymbol{x}$
for $i \in \{1,2,\ldots,D\}$ **do**
  generate a random number $r \in [0,1]$ according to the uniform distribution
  if $r \leq \xi$ **then**
    generate $x_i \in$ [-1,μ_i] according to *Q-PDF* described in Eq. (5)
  else
    generate $x_i \in [\mu_i$,1] according to *R-PDF* described in Eq. (6)
  end if
end for

where a_i and b_i are the lower and upper bounds of the uniform distribution (note that these are different from the search space bounds and change during the optimisation process). By integrating U-PDF, one can easily obtain the following.

$$U-CDF(x_i) = \begin{cases} \frac{1}{b_i-a_i}x_i - \frac{a_i}{b_i-a_i} & \text{if } x_i \in [a_i,b_i] \\ 1 & \text{if } x_i > b_i \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

The inverse $U-CDF^{-1}$ is given by:

$$U-CDF^{-1}(r) = (b_i-a_i)r + a_i. \tag{9}$$

Unlike the Gaussian model, the bounds a_i and b_i of *U-PDF* vary if the mean and standard deviation vary, according to Equation 10 and Equation 11.

$$a_i = \sqrt{3}\sigma_i + \mu_i \tag{10}$$

$$b_i = \mu_i - \sqrt{3}\sigma_i. \tag{11}$$

After calculating a_i and b_i, a uniform random number $r \in [0,1]$ must be generated to obtain a generic design variable x_i through Equation 9.

Examples of algorithms that employ these models are those in [38, 136].

5.5. Binary/discrete compact optimisation algorithms

The first compact algorithm, i.e., cGA, was designed for discrete optimisation. Since then, several other cMeta algorithms appeared in the literature for solving discrete problems. Alternative mutation operators for cGA are proposed in [37, 225, 228, 232], and an evolutionary strategy for the survival of the offspring is proposed in [233]. A study on elitism for GAs in [226] led to two variants with strong and weak elitism that outperformed the original non-elitist cGA and the popular (1+1)-ES. Elitism has also been used in [37, 228, 234, 235]. It should be noted that selection pressure can also be ensured by using a larger tournament selection [224].

Other studies try to improve the update process of $\boldsymbol{PV}$. A moving average strategy is presented in [227] and weights are used in [236]. Learning mechanisms for choosing among multiple evolved probability vectors are also available [237, 238]. Note that by doing this, the algorithm might achieve better performance, but would require more memory usage to store multiples PV. Hence, these

might no longer be considered lightweight according to the classification of this survey. However, if such memory is available on the device, these strategies can be used, as well as some μ-population algorithms that might have the same memory footprint. Unlike [239], some degree of inheritance is included in the probabilistic model, while the study [235] replaces $\boldsymbol{PV}$ with the belief vector $\boldsymbol{BV}$ to store probability values belonging to a Gaussian distribution with a given mean and variance. As in the continuous case, $\boldsymbol{BV}$ is not a real vector and is more memory-intensive than the original model. We would like to highlight that these advances available in the literature often perform better than the original methods. However, we argue that they are all based on adding complexity. This often leads to higher memory footprints and/or higher time overheads. This is a well-known problem in stochastic optimisation, where a trade-off between performances and other criteria (asymptotic complexity, overheads, smallest use of memory slots, etc.) must be taken according to the optimisation scenario. For the sake of completeness, we report a wide range of studies in this survey and provide these considerations to the reader.

Adaptation is an interesting feature of an optimisation algorithm. In [229, 240] adaptation to the problem is obtained by adding information on the frequencies and continuity of the update probabilities in the update rules. Most importantly, parameter adaptation schemes are proposed in [241] to have parameter-less cGAs capable of tuning the population size to remove unfavourable implications of genetic drift.

Multi-start schemes also help in compact optimisation. After each restart, the initial point changes, and usually $\boldsymbol{PV}$ is re-initialised. In [242], the restart occurs if no improvement in the objective function values is registered within a fixed number of consecutive generations. This can be seen as an enhanced exploration phase, where the cMeta is used to refine the new initial point. This can also be done with an opposite approach where the cMeta provides solutions that are refined with a local searcher as, e.g., steepest descent [243], or problem-specific local search operators [242, 244, 245].

For the sake of completeness, we report on the use of combinatorial cMeta used in systems that allow for parallelisation. Here, the problem of keeping the number of memory slots at a minimum level is less evident, while the simplicity of the models is important to a feasible and error-free implementation in devices such as FPGAs. The possibility of constructing parallel versions of cGA is discussed in [246–250], as well as in [251] (which considers multi-FPGA partitioning), while a memetic variant of cGA was presented in [252], along with a mechanism for fine-grained parallelism.

Other works employed cGA on various hardware devices [37, 253–257]. More recently, a GPU-enabled implementation of cGA was presented in [258], to solve a "seriously" large-scale (up to 10 million variables) Integer Linear Programming problem taken from [259], as well as continuous and discrete versions of the OneMax benchmark problem of up to one billion variables.

The use of cGA was also analysed in the context of noisy optimisation in [260]. This study showed that cGA can handle noise efficiently by adjusting its step size according to the level of noise. This method was called *graceful* noise scaling.

From a theoretical perspective, the runtime of a discrete cGA is studied in multiple works. In particular, it was analysed on the jump functions in [261–263]. In [264], lower and upper cGA runtime bounds have been derived for pseudo-Boolean functions, such as OneMax. Other studies have investigated how the size of the virtual population influences the performance of cGA [265, 266].

On a historical note, it is worth mentioning the Selfish Gene Algorithm (SGA) [267], which is very similar to cGA and was presented almost contemporaneously. SGA (not to be confused with the Simple GA) is based on the "selfish gene" theory in biological evolution [268]. Similarly to cGA, SGA

evolves a pool of genes that is updated by means of a virtual population. Recently, a new variant of SGA dubbed the "replacement and never penalising" SGA, was proposed in [269]. Instead of penalising the genes of the loser, this variant replaces them with those of the winner. This algorithm was applied to optimise the gymnastic movements of a humanoid robot. Two elitist variants (with persistent and non-persistent elitism) of this algorithm are presented in [270]. These seem to significantly outperform the original SGA. For details on SGAs, we refer to [271].

Finally, we should stress that there are several compact binary variants of other metaheuristics besides cGA and SGA. An algorithm known as cBinDE, which stands for compact Binary Differential Evolution, was introduced in [272]. This algorithm follows the same principle as cGA, but it uses the binary versions of mutation and crossover of the Differential Evolution algorithm combined with a simple local search. This algorithm was successfully used to maximise the functional coverage percentage in the verification of digital systems. Binary cDE was also studied in [273, 274]. Other works instead investigated binary versions of compact PSO [275], compact Firefly Algorithm [33] and compact Co-Firefly Algorithm [276, 277], compact Memetic Algorithms [278], and other kinds of compact EAs [279, 280]. We argue that potentially all metaheuristics can be made "compact". However, finding the most usable or suitable solution for a problem is a real challenge. This either requires a time-consuming empirical phase, or a more informed approach, which can be possible only in some cases. This is a fundamental research question in the field to be prioritised in the future.

The most interesting areas of application of compact optimisation in the discrete domain include: the Travelling Salesman Problem (TSP) [244, 281]; determining minimum set primers in Polymerase Chain Reaction (PCR) [282]; task scheduling in grid computing environments [283]; protein folding [284]; object recognition [285, 286]; soft decision decoding [287, 288]; minimising the number of coding operations required in multi-cast based on network coding [242]; estimating the parameters of the maximum log-likelihood function of a first-order moving average model $MA(1)$ [289] and a mixed model $ARMA(1,1)$ [243]; optimising the aggregation of multiple similarity measures to obtain a single similarity metric for ontology matching [290]; optimising ontology alignment [291]; designing multiple input multiple output wireless communication systems [292].

5.6. Real-valued Compact optimisation algorithms

In Sections 5.6.1 and 5.6.2, we report on EC and SI cMeta for the continuous domain, respectively.

5.6.1. Compact evolutionary algorithms (cEAs)

The real-valued compact Genetic Algorithm [135] is the first compact algorithm for continuous optimisation. It uses the Gaussian model described in Section 5.4.2 and only requires storing the elite individual, a temporary solution, and **PV** to perform the search. A similar variant is proposed in [137], which, despite being named a compact PSO algorithm, displays the same working mechanism of rcGA (we will refer to it as cross-rcGA). The peculiarity of this variant is that it performs a decomposition of the problem into three sub-problems. For each sub-problem, a local best solution is needed, as well as a local **PV** (in this context, it is similar to a PSO). We point out that this algorithm is based on an interesting idea but ends up requiring three local best solutions, three temp solutions, a global best slot, and three **PV** matrices, thus having a similar memory footprint of pMeta with a small population size. Another variant is presented in [21], where rcGA is hybridised with SNUM and is called cSNUM. Here, after generating an offspring solution with the rcGA mechanism, the SNUM operator is applied to a randomly selected design variable. cSNUM deals well with separable problems of different dimensionalities. Similarly, the Single/Multi Non-Uniform Mutation (cSM) algorithm [44] is a hybrid algorithm that combines an rcGA-like structure with the non-uniform

mutation (NUM) operator. This is very similar to cSNUM but perturbs all variables instead of just one. An interesting solution is the Uniform compact Genetic Algorithm (UcGA) [136], which is based on the uniform model and features a virtual population size that decreases linearly. Furthermore, it employs a local search operator.

The cDE algorithm [138] generates new trial solutions using the fundamental logic of DE, but rather than selecting them from a population, it samples them from a probabilistic model. Potentially, it can be used with all possible DE mutation strategies, crossover operators, and elitism schemes. However, depending on the use of a specific mutation operator, one may need to sample more individuals, thus requiring more memory. The simplest mutation, i.e., "rand/1", requires sampling three points to generate the so-called mutant vector. Compared to rcGA, a performance gain is recorded in most benchmark problems [138]. This might be due to the fact that DE is designed for continuous optimisation, and, therefore, cDE maintains the very same encoding and working logic as DE. This is not the case for GA, which is usually used over discrete domains and requires a real population to perform selection mechanisms such as fitness-proportionate or tournament selection (which can only be used with a size of two individuals in the memory-saving context). For these reasons, rcGA ends up performing worse than its population-based counterpart in many cases, particularly for mid- and high-dimensional problems (> 10), while cDE is comparable to its population-based counterpart. Moreover, in the continuous domain, cDE usually outperforms rcGA (but requires at least 3 individuals for the mutation, on top of the elite solution and a temporary vector). Similar considerations also apply to other cMeta algorithms, see [31] for details, as population-based algorithms that perform selection based on pairwise comparisons can be successfully and straightforwardly encoded into a compact scheme, while the other might display substantial performance degradation. In this light, there are many cDE-based algorithms in the literature. We remark that some of them might require the same amount of memory slots of population-based algorithms with small populations, but most are still characterised by simple and memory-cheap algorithmic structures. For example, the Disturbed Exploitation compact Differential Evolution (DEcDE) algorithm in [139] is a simple memetic approach based on a cDE algorithm that employs two DE exploitative search strategies. The first is the classic DE/rad/1/exp configuration. The second configuration instead has the trigonometric mutation (see [54] for details on DE). These exploitative DE operators are counterbalanced by a periodic stochastic alteration of the virtual population, which is meant to introduce exploration elements in the search. Despite its simplicity, the algorithm outperforms other compact algorithms in the benchmark functions tested. Similar results are obtained by using generalised Opposition-Based Learning (OBL) within cDE. The resulting cODE (compact Opposition-Based DE) [140] is competitive with its population-based counterpart. Differently, [141] proposes a memory-saving solution called Concise DE-based Chaotic Local Search (CDE-CLS) where a local searcher is added purposely to achieve fast convergence on a real-world problem. An adaptive version is instead the Compound Sinusoidal cDE (CScDE) proposed in [143]. Here, the compound sinusoidal heuristic is used to self-adapt the crossover rate and the mutation scale factor. CScDE outperforms most state-of-the-art compact algorithms on various benchmark problems. Other methods to improve upon exploration include the use of multiple cDE running together, as, e.g., [293]. However, these methods are not memory-saving, as they end up requiring a similar amount of memory slots to a small population-based algorithm, which may be preferred.

From the memory point of view, the cheapest compact DE framework is the compact Differential Evolution light (cDE-light) algorithm proposed in [142]. This is a fast approach, as it requires sampling only one candidate solution per iteration instead of the three required to perform the classic

DE/rand/exp. Furthermore, it does not require loops to implement the exponential crossover operator, which is replaced with a counterpart of this operator capable of predicting the number of design variables to be exchanged. The main idea to reduce the number of individuals in the rand/1 mutation (which is a linear combination of three randomly selected individuals) is to exploit the property of the Gaussian distribution from which these individuals need to be drawn [3]. Indeed, under the reasonable assumption of having statistically independent individuals, the corresponding three Gaussian distributions can be linearly combined to model the Gaussian model of the resulting mutant vector. Without having to sample an individual to generate the mutant, this can simply be obtained from his Gaussian model. As for the crossover "light", the derivation of the formula predicting the number of variables to be exchanged without requiring a loop through two individuals is provided in [142]. This algorithm is based on interesting design ideas, and, despite the assumptions and approximations, it performs well and behaves similarly to cDE. As seen for a single-solution algorithm, in this case, a performance gain is recorded when it is equipped with the restart with the inheritance mechanism described in Section 5.2. The study in [43] shows that most compact algorithms can benefit from this scheme by benchmarking restart variants of compact DE, PSO, and other algorithms whose compact version is introduced in the next section.

In the literature, there are examples of compact Evolution Strategy (cES) algorithms, such as c(1+1)-ES and the c(μ, λ)-ES [294]. From the experimental analysis in [294] the c(1+1)-ES algorithm appeared to be as effective as the original (1+1)-ES. Conversely, c(μ, λ)-ES appeared to perform worse than its population-based (μ, λ)-ES counterpart, especially for high values of λ.

5.6.2. Compact swarm intelligence algorithms

Compact Particle Swarm Optimisation (cPSO) [42] is the compact counterpart of the PSO algorithm for the continuous domain. This is simply obtained by using the Gaussian model to generate a new particle $\boldsymbol{x}$, which is perturbed by the velocity vector $\boldsymbol{v}$ as in the original PSO. However, to avoid sampling the swarm, some adjustments are needed; the concept of the local best particle cannot be replicated if a single solution is employed at a time to maintain a memory-saving framework. For this reason, the PSO update formula for $\boldsymbol{v}$ only takes into account the actual global best solution $\boldsymbol{x}_{gb}$, while the local best solution $\boldsymbol{x}_{lb}$ is drawn from the Gaussian model with the current **PV** values. Note that there are variants of this algorithm using different distributions for the model, as in the real-parameter compact supervision for PSO (rcSPSO) [144], where a combination of Cauchy and Gaussian distributions are used. Self-adaptive variants, like the one in [145], have also been proposed.

The compact Bacterial Foraging Optimisation (cBFO) algorithm [146] also employs the same chemotaxis scheme of population-based BFO, but models the population with the Gaussian model of section 5.4.2. Similarly to BFO, a new solution is generated from the model at each chemotactic step, and a mix of tumble/swim moves is attempted. When generating new offspring (either using the sampling mechanism or via a tumble/swim), its fitness value is compared to that of the current best solution. The compact implementation of the reproduction and elimination/dispersal steps is a bit different. Instead of preserving and replicating the best $S/2$ bacteria as BFO does, cBFO moves the PDF in favour of the elite and shrinks over it. As a result, forcing a PDF update is an approximation of the sexual reproduction step. Finally, the injection of new randomly produced bacteria into the swarm is modelled using a perturbation of $\boldsymbol{PV}$ in the elimination/dispersal step.

[3] Note that this is an approximation as the distribution is truncated and not a theoretical Gaussian function. However, the results are satisfactory

The list of compact algorithms is large. As many population-based algorithms can be made compact, the literature keeps offering examples of a new compact version to be used mainly in applied contexts. A compact Artificial Bee Colony (cABC) is proposed in [147] and an Enhanced cABC (EcABC) variant is proposed in [148]. A parallel structure, abbreviated as pcABC, is presented in [295]. However, the latter is meant for hardware systems with multiple cores/processors which do not suffer from memory or computational limitations. Some compact Firefly Algorithms (cFAs) [38] are also widely used. Note that these are obtained with some simplifications of the original strategy, which makes the compact version (in particular, the persistent variant using the Gaussian distribution) follow the same steps of rcGA except for an extra step before updating **PV**. This is required to direct the loser toward the winner to adhere to the original framework. Based on the method for updating the elite solution, which can require using L'evy flight movement, Opposition-Based Learning, and the use of Gaussian or uniform distribution, 12 cFA variants can be obtained. More are presented in [150, 151]. In this light, one can see that making an algorithm compact can be simple. However, the current trend of simplifying existing algorithms just to present a new optimisation framework does not necessarily help to progress in understanding what good practices are in the algorithmic design phase. This is particularly true when the design is driven only by inspiring metaphors, which often results in new algorithms whose working mechanism is either unclear or similar to other existing heuristics. These metaphor-led compact algorithms are now abundant in the literature for solving real-world applications. We do report some one of these applied scenarios solved with, e.g., compact Cat Swarm Optimisation (cCSO) [152], compact Teaching-Learning-Based Optimisation (cTLBO) [159], compact Harris Hawks Optimisation (cHHO) algorithm [160], compact Bat Algorithms [149], compact Flower Pollination Algorithms [154, 155], compact Pigeon-Inspired Optimisation (cPIO) [153], the compact Sine Cosine Algorithm (cSCA/pcSCA) [157, 296] and McSCA with Multi-group and Multi-strategy (based on different DE mutations) [297], the compact Equilibrium Optimiser algorithm (cEO/pcEO) [158], the compact Cuckoo Search (cCS) [156], compact Harmony Search Algorithms (cHSA) [161] and many others. We direct the reader to these studies and argue that while the application domain is interesting, it is difficult to understand what the contribution of proposing such algorithms to solve such problems is. In most cases, these are similar to existing methods or just present insufficiently motivated combinations of operators. Although we believe that progress in the algorithmic design must be kept alive within the community, by surveying the recent literature, we call for more emphasis on analysing the algorithms to have a more informed design phase in the future.

6. Lightweight metaheuristics applications

The main scenarios and motivations for using lightweight algorithms are summarised below.

Embedded systems. Lightweight algorithms can be used in several low-cost, resource-limited computing devices that are used nowadays in a wide range of miniaturised commercially available devices, such as those used, for example, in wet laboratories [32, 33], humanoid robots [38], flying robots (micro-aerial vehicles) [298], and mobile robots [22].

Real-time optimisation. When applications require a real-time optimisation problem to be solved, sMeta can be a logical choice (as well as most compact algorithms as long as the complex model is not used). In engineering, such situations are abundant and in most cases do not require an optimal solution but rather a solution of satisfactory quality within some precision thresholds. Examples

of these scenarios are nonlinear optimal control, receding horizon control, and moving horizon estimation [299]. Other applications might involve solving large-scale optimisation problems, such as optimising the parameters of black-box models (e.g., a deep neural network or a hidden Markov model), or solving inverse problems [217] on board an embedded system.

Hybrid optimisation algorithms. Lightweight algorithms, and in particular single-solution metaheuristics, are useful "building blocks" for hybrid algorithms [136, 141]. Even in non-memory-saving contexts, this is evident when dealing with hyper-heuristics and memetic computing approaches.

Other situations. A universal metaheuristic does not exist, and in many real-world scenarios, simple algorithms perform better than more complex ones. Based on examples in, for example, optimal control in industrial plants [42], neural network training [159], and ontology mapping [300], we recommend taking them into account as they might be able to provide satisfactory results while keeping implementation difficulties relatively low.

Relevant application contexts where lightweight algorithms are used are listed below.

1. Wireless Sensor Networks (WSNs), e.g., Optimised deployment [301], Minimised energy depletion [302], Base station locations [303], Topology control scheme [304], Clustering formation [301] [149, 302] and Node Localisation [145, 296];
2. Embedded control systems [39, 124, 135, 138, 139, 305];
3. Robotics, e.g., industrial robots [39, 139, 141, 142, 305, 306], mobile robots [22], humanoid robots [38, 150, 151, 161], unmanned aerial vehicles [124, 156];
4. Electronic design, e.g., of magnetic field sensors [217], printed circuit boards [307], digital signal processing elements [125, 308–311], and antennas [312, 313];
5. Computer vision, e.g., image segmentation [152], clustering [132], face recognition [122, 210];
6. Power/energy systems [42, 314, 315] and renewable energy systems [153, 158, 316];
7. Transportation [157, 297] and civil engineering [317, 318];
8. Design of engineering/mechanical structures [160, 319];
9. Natural language processing [320];
10. Machine learning [141, 159, 321–323];
11. Ontology engineering [136, 137];
12. Cloud computing security [324];
13. Bioinformatics [211, 212].

Table 1 contains the aforementioned applications, related articles, and specific algorithms that are being used.

7. Discussion and open issues

Based on the survey of the literature on lightweight metaheuristics reported before, we can now draw the following conclusions:

- Compact and single-solution algorithms are commonly expected to be outperformed by population-based algorithms in terms of solution quality. However, this is not always the case. Furthermore, there might be applied contexts where these are the only choices because of memory constraints or just preferred for speed gain and simplicity in their implementation.

- Some well-known drawbacks of population-based optimisation are premature convergence or stagnation. When the first occurs, the population loses diversity, and the algorithm is stuck in a local optimum. In the second case, even though the population is still diverse, the search may stagnate, meaning that the operators cannot create offspring that outperform their parent solutions. Premature convergence also plagues lightweight algorithms. Interestingly, the latter can be used to help population-based algorithms overcome stagnation by providing additional movement in the search space [78] and premature convergence by adding the population superfit individuals.

- In relation to the previous point, most compact optimisation methods have an intrinsic limitation when dealing with multimodal functions [21, 143]. Indeed, since they lack an actual population of candidate solutions, they cannot provide a sufficient degree of diversity, particularly in the long term after the Gaussian model has converged. As a result, unless certain restart methods are introduced, these types of algorithms excel at exploitation but fall short of exploration, which is required to handle multimodal functions effectively. Indeed, after the Gaussian model converges, new solutions are sampled from a relatively tiny subset of the search space, resulting in a local search.

- Due to the fact that compact optimisation algorithms by nature handle each variable independently, they perform exceptionally well on separable problems. This is especially true for some of the most recent algorithms, e.g., CScDE, SNUM, and cSNUM [21, 143]. However, it is possible to endow both cMeta and sMeta with algorithmic moves that handle multiple variables at the same time. A concrete example is that recently proposed in [44], where the cSM approach was shown to be successful, particularly in multimodal problems.

- Among compact algorithms, some (e.g., rcGA) work well in lower dimensionalities, while others (e.g., cPSO) perform especially well in larger dimensionalities (a similar observation can be made for single-solution algorithms). This seems not only to be a consequence of the inherent search logic underlying each algorithm. Another possible explanation has been provided in a recent study provided in [325], which showed that the correlation between pairs of variables appears to continuously decrease as the size of the problem increases, from the point of view of a stochastic search algorithm. As a consequence, large-scale non-separable high-dimensional problems can be approached as if they are separable. According to the same authors, this effect arises from the fact that in high dimensionalities only a very limited portion of the decision space can be explored with a reasonable computational budget. Therefore, exploration should be performed with any improvement along each variable, which is consistent with simple methods that use a limited computational budget to focus mainly on exploitation. This is also compatible with most multi-start lightweight algorithms.

8. Future research directions

Promising research lines to improve upon the current state-of-the-art in light-weight optimisation algorithms are summarised below.

- **Probabilistic models**: Nearly the totality of compact algorithms employ the Gaussian distribution, which is simple to work with, but alternative models might be more suitable and should be investigated.
- **Multimodality**: Poor performance in this class of problems suggests the need to focus more on exploratory operators or mechanisms in the algorithmic design phase.
- **Scalability**: The problem of dealing with increasing dimensionality values without deterioration of performance needs further attention, as truly large-scale problems are becoming increasingly frequent, even in low-resource devices.
- **Hybridisation**: It will be interesting to explore different combinations of multiple compact logics.
- **Adaptation and learning**: Lightweight algorithms with increased intelligence will be needed, that is, algorithms that can self-adapt their parameters to the problem at hand. Another possibility would be to put multiple lightweight algorithms together, using, for instance, a scheme similar to that introduced in [326]: in this scheme, 6 algorithms are arranged into three bags, each with two algorithms, depending on the number of solutions used in the optimisers. Once a bag is selected, its optimisers are run according to a reinforcement learning process based on its performance. As a result, if the considered optimiser performs well (based on the fitness improvement), it will be rewarded, while the others will be penalised. The best-performing optimiser will earn more probability to run in the next iterations, while the other optimisers will be activated less frequently, eventually until their corresponding probability is null.
- **Noisy/dynamic optimisation**: It will be useful to investigate new compact optimisation schemes (for instance, with new distributions or new sampling mechanisms) to deal with noisy functions or Dynamic Optimisation Problems (DOPs) [327, 328] (i.e., problems where the search space changes over time). Concerning noisy optimisation, apart from some works on cGA [260, 329–331], the only few works dealing with noise are based on rcGA [332], cDE [333] and a compact EDA framework [334]. In terms of dynamic optimisation, a Hooke-Jeeves-based Memetic Algorithm (HJMA) was presented in [335], where experiments have been conducted on the Moving Peaks (MP) problem[4]. To address the need to continuously adapt to landscape changes, some improvements in cGA have been proposed in [340], based on techniques of hyper-mutation and random immigrants. The results of a modified version of the Moving Peaks Benchmark indicate that both strategies improve the algorithm performance for dynamic environments. Instead, the compact adaptive mutation

[4]This benchmark is defined in [336] as an artificial multidimensional landscape comprised of multiple peaks, each of which has its height, width, and position slightly altered whenever a change occurs in the environment. Its complexity can be raised by increasing the number of dimensions/peaks and by adding noise over the whole landscape. A review of the approaches that have been tested in the dynamic MP problem can be found in [337]. More recently, some authors proposed the Deterministic Distortion and Rotation Benchmark (DDRB) [338], a method to generate Deterministic Dynamic Multimodal Optimisation Problems (DMMOP) considering both dynamic and multimodal characteristics, which can simulate more diverse sets of challenges. In the same context, another set of benchmark problems, as well as an optimisation framework, called PopDMMO, containing several population-based algorithms, was designed in [339].

genetic algorithm (amcGA) presented in [341] is based on an adaptive mechanism where the mutation scheme is directly linked to a change detection scheme so that the change detection scheme regulates the mutation rate (i.e., the degree of change determines the probability of mutation). This method was tested in [342] using a real-world dynamic optimisation problem that includes designing and optimising a PID controller for a torsional mass-spring-damper system in a dynamic environment. Other variants of amcGA can also be found in [343]. However, more research in this direction is needed.

- **Constrained and multi-objective problems:** Bound-constrained single-objective optimisation has been the focus of most research on lightweight metaheuristics. In many application scenarios, however, one needs to handle multiple objectives at the same time and/or handle a set of equality/inequality constraints. Therefore, future research on lightweight algorithms for multi-objective and/or constrained optimisation applications may be interesting. Regarding sMeta, some multi-objective variants of SEO have been adapted, for example, to solve the problem of home healthcare routing and scheduling [344], to optimise an integrated system of water supply and waste collection [345], and to optimise a municipal solid waste problem [346]. Instead, in [347] a multi-objective variant of the VS algorithm is introduced. Regarding cMeta, there are already some studies that go in this direction already exist; see, e.g., [348, 349]. These research papers show that Multi-Objective compact Differential Evolution (MOcDE) and Multi-Objective compact Particle Swarm Optimiser (MOcPSO), respectively, can be used successfully for solving unconstrained, continuous multi-objective optimisation problems. In [277, 350], the authors solve the ontology alignment problem using the compact multi-objective Co-Firefly algorithm and cPSO, respectively. One main drawback of cMeta and sMeta is that, without an actual population or an archive, they cannot keep a Pareto front in memory. Lack of diversity is also an issue for multi-objective optimisation. Therefore, more research is needed in this direction. Regarding constrained optimisation, an Improved SA (ISA) is introduced in [351], which is capable of dealing with only linear constraints. ISA is characterised by changing only one component of the current solution at each iteration, without penalty function. Another similar work [352] proposed two variants based on a hybrid Simulated Annealing-Hill Climbing algorithm, to solve constrained optimisation problems. The first version incorporates penalty methods for constraint handling, whereas the second one eliminates the need for imposing penalties in the objective function by tracing feasible and infeasible solution sequences independently. Also, more research seems to be needed in this area.

- The trend in designing novel algorithms solely by following an inspiring metaphor and making their compact versions available would not help to understand how these simple methods work and can be improved in the future. Even if good results can be obtained with such algorithms over some application domains, this does not seem to lead to any progress in explaining the algorithmic behaviour and the well-known drawbacks of metaheuristics. Also from the novelty point of view, this approach is arguable [65, 66, 353]. Therefore, we call for more fundamental research in the direction of overcoming drawbacks to obtain an improved self-adaptive algorithmic structure.

Data Availability

Original data were not collected or generated to support this study. Instead, we have included comments on the references surveyed in our article that contain links to data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Souheila Khalfi: conceptualisation, investigation, writing - original draft, writing - review & editing, visualisation; *Fabio Caraffini*: conceptualisation, writing - original draft, writing - review & editing, visualisation; *Giovanni Iacca*: conceptualisation, investigation, writing - original draft.

References

[1] S. E. Thompson, S. Parthasarathy, Moore's law: the future of SI microelectronics, Materials today 9 (2006) 20–25.

[2] F. Pisani, F. M. C. de Oliveira, E. S. Gama, R. Immich, L. F. Bittencourt, E. Borin, Fog computing on constrained devices: paving the way for the future iot, Advances in Edge Computing: Massive Parallel Processing and Applications 35 (2020) 22–60.

[3] L. P. Kaelbling, Learning in embedded systems, MIT press, 1993.

[4] S. Schiaffino, A. Amandi, Intelligent user profiling, in: Artificial Intelligence An International Perspective, Springer, 2009, pp. 193–216.

[5] F. W. Glover, G. A. Kochenberger, Handbook of metaheuristics, Springer Science & Business Media, 2006.

[6] E.-G. Talbi, Metaheuristics: from design to implementation, John Wiley & Sons, 2009.

[7] K. Miettinen, Evolutionary algorithms in engineering and computer science: recent advances in genetic algorithms, evolution strategies, evolutionary programming, GE, John Wiley & Sons, Inc., 1999.

[8] R. Chiong, F. Neri, R. I. McKay, Nature that breeds solutions, International Journal of Signs and Semiotic Systems (IJSSS) 2 (2012) 23–44.

[9] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE transactions on evolutionary computation 1 (1997) 67–82.

[10] A. Tzanetos, G. Dounias, Nature inspired optimization algorithms or simply variations of metaheuristics?, Artificial Intelligence Review 54 (2021) 1841–1862.

[11] T. Ting, X.-S. Yang, S. Cheng, K. Huang, Hybrid metaheuristic algorithms: past, present, and future, in: Recent Advances in Swarm Intelligence and Evolutionary Computation, Springer, 2015, pp. 71–83.

[12] A. P. Piotrowski, J. J. Napiorkowski, Some metaheuristics should be simplified, Information Sciences 427 (2018) 32–62.

[13] F. Wortmann, K. Flüchter, Internet of things, Business & Information Systems Engineering 57 (2015) 221–224.

[14] A. Prügel-Bennett, Benefits of a population: Five mechanisms that advantage population-based algorithms, IEEE Transactions on Evolutionary Computation 14 (2010) 500–517.

[15] D. R. da S. Medeiros, M. F. Torquato, M. A. Fernandes, Embedded genetic algorithm for low-power, low-cost, and low-size-memory devices, Engineering Reports 2 (2020) e12231.

[16] I. Fister, G. Vrbancic, T. Hozjan, V. Podgorelec, Performance study of bat algorithm running on embedded hardware, in: 2019 23rd International Conference Electronics, IEEE, 2019, pp. 1–4.

[17] M. El-Shafei, I. Ahmad, M. G. Alfailakawi, Implementation of harmony search on embedded platform, Microprocessors and Microsystems 45 (2016) 187–197.

[18] A. Hassanein, M. El-Abd, I. Damaj, H. U. Rehman, Parallel hardware implementation of the brain storm optimization algorithm using fpgas, Microprocessors and Microsystems 74 (2020) 103005.

[19] A. Ortiz, E. Mendez, D. Balderas, P. Ponce, I. Macias, A. Molina, Hardware implementation of metaheuristics through labview fpga, Applied Soft Computing 113 (2021) 107908.

[20] P. H. O. Santos, G. L. Soares, T. M. Machado-Coelho, B. A. G. de Oliveira, P. I. Ekel, F. M. F. Ferreira, C. A. P. da Silva Martins, Multi-objective genetic algorithm implemented on a stm32f microcontroller, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1–7.

[21] S. Khalfi, G. Iacca, A. Draa, On the use of single non-uniform mutation in lightweight metaheuristics, Soft Computing 26 (2022) 2259–2275.

[22] G. Iacca, F. Caraffini, F. Neri, Memory-saving memetic computing for path-following mobile robots, Applied Soft Computing 13 (2013) 2003–2016.

[23] H. Rosenbrock, An automatic method for finding the greatest or least value of a function, The computer journal 3 (1960) 175–184.

[24] M. J. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, The computer journal 7 (1964) 155–162.
[25] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, in: Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE, 1996, pp. 312–317.
[26] S.-M. Guo, C.-C. Yang, Enhancing differential evolution utilizing eigenvector-based crossover operator, IEEE Transactions on Evolutionary Computation 19 (2014) 31–49.
[27] F. Caraffini, F. Neri, L. Picinali, An analysis on separability for memetic computing automatic design, Information Sciences 265 (2014) 1–22.
[28] J. A. Nelder, R. Mead, A simplex method for function minimization, The computer journal 7 (1965) 308–313.
[29] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions i. binary parameters, in: International conference on parallel problem solving from nature, Springer, 1996, pp. 178–187.
[30] P. Larrañaga, J. A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation, Springer Science & Business Media, 2001.
[31] F. Neri, G. Iacca, E. Mininno, Compact optimization, in: Handbook of optimization, Springer, 2013, pp. 337–364.
[32] A. Soares, T. De Lima, F. Soares, C. Coelho, F. Federson, A. Delbem, J. Van Baalen, Mutation-based compact genetic algorithm for spectroscopy variable selection in determining protein concentration in wheat grain, Electronics Letters 50 (2014) 932–934.
[33] L. C. de Paula, H. V. Nogueira, A. S. Soares, T. W. de Lima, C. J. Coelho, A compact firefly algorithm for the variable selection problem in pharmaceutical ingredient determination, in: 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 3832–3838.
[34] L. Dey, A. Mukhopadhyay, Compact genetic algorithm-based feature selection for sequence-based prediction of dengue-human protein interactions, IEEE/ACM Transactions on Computational Biology and Bioinformatics 19 (2021) 2137–2148.
[35] Soniya, S. Paul, L. Singh, Simultaneous structure and parameter learning of convolutional neural network, in: N. K. Verma, A. K. Ghosh (Eds.), Computational Intelligence: Theories, Applications and Future Directions - Volume II, Springer Singapore, Singapore, 2019, pp. 93–104. doi:10.1007/978-981-13-1135-2_8.
[36] Soniya, S. Paul, L. Singh, Application and need-based architecture design of deep neural networks, International Journal of Pattern Recognition and Artificial Intelligence 34 (2020) 2052014.
[37] J. C. Gallagher, S. Vigraham, G. Kramer, A family of compact genetic algorithms for intrinsic evolvable hardware, IEEE Transactions on evolutionary computation 8 (2004) 111–126.
[38] L. Tighzert, C. Fonlupt, B. Mendil, A set of new compact firefly algorithms, Swarm and Evolutionary Computation 40 (2018) 92–115.
[39] G. Iacca, F. Caraffini, F. Neri, E. Mininno, Robot base disturbance optimization with compact differential evolution light, in: European Conference on the Applications of Evolutionary Computation, Springer, 2012, pp. 285–294.
[40] A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, R. Tavakkoli-Moghaddam, The social engineering optimizer (seo), Engineering Applications of Artificial Intelligence 72 (2018) 267–293.
[41] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information sciences 237 (2013) 82–117.
[42] F. Neri, E. Mininno, G. Iacca, Compact particle swarm optimization, Information Sciences 239 (2013) 96–121.
[43] G. Iacca, F. Caraffini, Re-sampled inheritance compact optimization, Knowledge-Based Systems 208 (2020) 106416.
[44] S. Khalfi, , G. Iacca, A. Draa, A single-solution–compact hybrid algorithm for continuous optimization, Memetic Computing 15 (2023) 155–204.
[45] T. Back, D. B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, IOP Publishing Ltd., 1997.
[46] A. E. Eiben, J. E. Smith, Introduction to evolutionary computing, Springer, 2003.
[47] M. Clerc, Particle swarm optimization, John Wiley & Sons, 2010.
[48] W. Tang, Q. Wu, Biologically inspired optimization: a review, Transactions of the Institute of Measurement and Control 31 (2009) 495–515.
[49] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, D. Fister, A brief review of nature-inspired algorithms for optimization, Elektrotehniski vestnik 80 (2013) 116–122.
[50] F. Yang, P. Wang, Y. Zhang, L. Zheng, J. Lu, Survey of swarm intelligence optimization algorithms, in: 2017 IEEE International Conference on Unmanned Systems (ICUS), IEEE, 2017, pp. 544–549.
[51] S. Katoch, S. S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimedia Tools and Applications 80 (2021) 8091–8126.
[52] R. D. Al-Dabbagh, F. Neri, N. Idris, M. S. Baba, Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy, Swarm and Evolutionary Computation 43 (2018) 284–311.

[53] E. H. Houssein, A. G. Gad, K. Hussain, P. N. Suganthan, Major advances in particle swarm optimization: Theory, analysis, and application, Swarm and Evolutionary Computation 63 (2021) 100868.
[54] S. Das, S. S. Mullick, P. N. Suganthan, Recent advances in differential evolution – an updated survey, Swarm and Evolutionary Computation 27 (2016) 1–30.
[55] E.-G. Talbi, Machine learning into metaheuristics: A survey and taxonomy, ACM Computing Surveys (CSUR) 54 (2021) 1–32.
[56] E.-G. Talbi, Automated design of deep neural networks: A survey and unified taxonomy, ACM Computing Surveys (CSUR) 54 (2021) 1–37.
[57] S. Almufti, R. Marqas, V. Ashqi, Taxonomy of bio-inspired optimization algorithms, Journal Of Advanced Computer Science & Technology 8 (2019) 23.
[58] R. A. Zitar, M. A. Al-Betar, M. A. Awadallah, I. A. Doush, K. Assaleh, An intensive and comprehensive overview of jaya algorithm, its versions and applications, Archives of Computational Methods in Engineering 29 (2022) 763–792.
[59] L. Abualigah, M. Shehab, M. Alshinwan, H. Alabool, Salp swarm algorithm: a comprehensive survey, Neural Computing and Applications 32 (2020) 11195–11215.
[60] R. Solgi, H. A. Loáiciga, Bee-inspired metaheuristics for global optimization: a performance comparison, Artificial Intelligence Review 54 (2021) 4967–4996.
[61] A. LaTorre, D. Molina, E. Osaba, J. Poyatos, J. Del Ser, F. Herrera, A prescription of methodological guidelines for comparing bio-inspired optimization algorithms, Swarm and Evolutionary Computation 67 (2021) 100973.
[62] E. Osaba, E. Villar-Rodriguez, J. Del Ser, A. J. Nebro, D. Molina, A. LaTorre, P. N. Suganthan, C. A. Coello Coello, F. Herrera, A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems, Swarm and Evolutionary Computation 64 (2021) 100888.
[63] A. H. Halim, I. Ismail, S. Das, Performance assessment of the metaheuristic optimization algorithms: an exhaustive review, Artificial Intelligence Review 54 (2021) 2323–2409.
[64] A. E. Ezugwu, A. K. Shukla, R. Nath, A. A. Akinyelu, J. O. Agushaka, H. Chiroma, P. K. Muhuri, Metaheuristics: a comprehensive overview and classification along with bibliometric analysis, Artificial Intelligence Review 54 (2021) 4237–4316.
[65] K. Sörensen, Metaheuristics—the metaphor exposed, International Transactions in Operational Research 22 (2015) 3–18.
[66] C. L. Camacho-Villalón, M. Dorigo, T. Stützle, Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors, International Transactions in Operational Research 30 (2022) 2945–2971.
[67] T. Achary, S. Pillay, S. M. Pillai, M. Mqadi, E. Genders, A. E. Ezugwu, A performance study of meta-heuristic approaches for quadratic assignment problem, Concurrency and Computation: Practice and Experience 33 (2021) e6321.
[68] P. Agrawal, H. F. Abutarboush, T. Ganesh, A. W. Mohamed, Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019), IEEE Access 9 (2021) 26766–26791.
[69] A. Singh, S. Sharma, J. Singh, Nature-inspired algorithms for wireless sensor networks: A comprehensive survey, Computer Science Review 39 (2021) 100342.
[70] Y. Wu, A survey on population-based meta-heuristic algorithms for motion planning of aircraft, Swarm and Evolutionary Computation 62 (2021) 100844.
[71] O. Zedadra, A. Guerrieri, N. Jouandeau, G. Spezzano, H. Seridi, G. Fortino, Swarm intelligence-based algorithms within iot-based systems: A review, Journal of Parallel and Distributed Computing 122 (2018) 173–187.
[72] F. Héliodore, A. Nakib, B. Ismail, S. Ouchraa, L. Schmitt, Metaheuristics for intelligent electrical networks, Wiley Online Library, 2017.
[73] C. Huang, Y. Li, X. Yao, A survey of automatic parameter tuning methods for metaheuristics, IEEE transactions on evolutionary computation 24 (2019) 201–216.
[74] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, in: Proceedings of the 17th IEEE region 10 international conference on computer, communications, control and power engineering, IEEE, 2002, pp. 606–611.
[75] K. E. Parsopoulos, Cooperative micro-differential evolution for high-dimensional problems, in: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, Association for Computing Machinery, 2009, pp. 531–538.
[76] K. E. Parsopoulos, Parallel cooperative micro-particle swarm optimization: A master–slave model, Applied Soft Computing 12 (2012) 3552–3579.
[77] A. Rajasekhar, S. Das, S. Das, μabc: a micro artificial bee colony algorithm for large scale global optimization, in: Proceedings of the 14th annual conference companion on Genetic and evolutionary computation, Association for

Computing Machinery, 2012, pp. 1399–1400.
[78] F. Caraffini, F. Neri, I. Poikolainen, Micro-differential evolution with extra moves along the axes, in: 2013 IEEE Symposium on Differential Evolution (SDE), IEEE, 2013, pp. 46–53.
[79] F. Viveros Jiménez, E. Mezura Montes, A. Gelbukh, Empirical analysis of a micro-evolutionary algorithm for numerical optimization, International Journal of Physical Sciences 7 (8) (2012) 1235–1258.
[80] D. E. Goldberg, Sizing populations for serial and parallel genetic algorithms, in: Proceedings of the 3rd international conference on genetic algorithms, Morgan Kaufmann Publishers Inc., 1989, pp. 70–79.
[81] K. Krishnakumar, Micro-genetic algorithms for stationary and non-stationary function optimization, in: Intelligent Control and Adaptive Systems, International Society for Optics and Photonics, 1990, pp. 289–296.
[82] V. W. Tam, K.-Y. Cheng, K.-S. Lui, Using micro-genetic algorithms to improve localization in wireless sensor networks., Journal of communications 1 (2006) 1–10.
[83] F. Viveros-Jiménez, E. Mezura-Montes, A. Gelbukh, Elitistic evolution: a novel micro-population approach for global optimization problems, in: 2009 Eighth Mexican International Conference on Artificial Intelligence, IEEE, 2009, pp. 15–20.
[84] M. Olguín-Carbajal, J. C. Herrera-Lozada, J. Sandoval-Gutierrez, J. I. Vasquez-Gomez, J. F. Serrano-Talamantes, F. Chavez-Estrada, I. Rivera-Zarate, M. Hernandez-Boláos, A micro-differential evolution algorithm for continuous complex functions, IEEE Access 7 (2019) 172783–172795.
[85] S. Nesmachnow, H. Cancela, E. Alba, A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling, Applied Soft Computing 12 (2012) 626–639.
[86] H. Salehinejad, S. Rahnamayan, H. R. Tizhoosh, S. Y. Chen, Micro-differential evolution with vectorized random mutation factor, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 2055–2062.
[87] S. Rahnamayan, H. R. Tizhoosh, Image thresholding using micro opposition-based differential evolution (micro-ode), in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1409–1416.
[88] M. A. Sotelo-Figueroa, H. J. P. Soberanes, J. M. Carpio, H. J. F. Huacuja, L. C. Reyes, J. A. S. Alcaraz, Evolving bin packing heuristic using micro-differential evolution with indirect representation, in: Recent Advances on Hybrid Intelligent Systems, Springer, 2013, pp. 349–359.
[89] M. Olguin-Carbajal, J. C. Herrera-Lozada, J. Arellano-Verdejo, R. Barron-Fernandez, H. Taud, Micro differential evolution performance empirical study for high dimensional optimization problems, in: International Conference on Large-Scale Scientific Computing, Springer, 2013, pp. 281–288.
[90] H. Salehinejad, S. Rahnamayan, H. R. Tizhoosh, Opposition-based ensemble micro-differential evolution, in: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2017, pp. 1–8.
[91] C. Brown, Y. Jin, M. Leach, M. Hodgson, μjade: adaptive differential evolution with a small population, Soft computing 20 (2016) 4111–4120.
[92] J. C. F. Cabrera, C. A. Coello Coello, Handling constraints in particle swarm optimization using a small population size, in: Mexican International Conference on Artificial Intelligence, Springer, 2007, pp. 41–51.
[93] T. Huang, A. S. Mohan, Micro-particle swarm optimizer for solving high dimensional optimization problems (μpso for high dimensional optimization problems), Applied Mathematics and Computation 181 (2006) 1148–1154.
[94] J. C. Herrera-Lozada, H. Calvo, H. Taud, A micro artificial immune system, Polibits 43 (2011) 107–111.
[95] S. Dasgupta, A. Biswas, S. Das, B. K. Panigrahi, A. Abraham, A micro-bacterial foraging algorithm for high-dimensional optimization, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 785–792.
[96] A. Rajasekhar, S. Das, P. N. Suganthan, Design of fractional order controller for a servohydraulic positioning system with micro artificial bee colony algorithm, in: 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.
[97] A. O. Topal, O. Altun, Y. E. Yildiz, Micro bat algorithm for high dimensional optimization problems, International Journal of Computer Applications 122 (12).
[98] S. A. Kazarlis, S. E. Papadakis, J. Theocharis, V. Petridis, Microgenetic algorithms as generalized hill-climbing operators for ga optimization, IEEE Transactions on Evolutionary Computation 5 (2001) 204–217.
[99] C. A. Coello Coello, G. T. Pulido, Multiobjective optimization using a micro-genetic algorithm, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc, 2001, pp. 274–282.
[100] Q. Lin, J. Chen, A novel micro-population immune multiobjective optimization algorithm, Computers & operations research 40 (2013) 1590–1601.
[101] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global optimization 13 (1998) 455–492.
[102] M. Pelikan, Probabilistic model-building genetic algorithms, in: Hierarchical Bayesian Optimization Algorithm:

Toward a new Generation of Evolutionary Algorithms, Springer Berlin Heidelberg, 2005, pp. 13–30.
[103] Y. Davidor, Epistasis variance: Suitability of a representation to genetic algorithms, Complex Systems 4 (1990) 369–383.
[104] F. Caraffini, F. Neri, A study on rotation invariance in differential evolution, Swarm and Evolutionary Computation 50 (2019) 100436.
[105] S. Baluja, Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science (1994).
[106] J. De Bonet, C. Isbell, P. Viola, Mimic: Finding optima by estimating probability densities, Advances in neural information processing systems 9 (1996) 424–430.
[107] M. Pelikan, H. Mühlenbein, The bivariate marginal distribution algorithm, in: Advances in Soft Computing, Springer, 1999, pp. 521–535.
[108] H. Mühlenbein, T. Mahnig, Fda-a scalable evolutionary algorithm for the optimization of additively decomposed functions, Evolutionary computation 7 (1999) 353–376.
[109] G. R. Harik, F. G. Lobo, K. Sastry, Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ecga), in: Scalable optimization via probabilistic modeling, Springer, 2006, pp. 39–61.
[110] P.-C. Hung, Y.-P. Chen, Iecga: Integer extended compact genetic algorithm, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation, Association for Computing Machinery, 2006, pp. 1415–1416.
[111] L. Fossati, P. L. Lanzi, K. Sastry, D. E. Goldberg, O. Gomez, A simple real-coded extended compact genetic algorithm, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 342–348.
[112] P. L. Lanzi, L. Nichetti, K. Sastry, D. Voltini, D. E. Goldberg, Real-coded extended compact genetic algorithm based on mixtures of models, in: Linkage in evolutionary computation, Springer, 2008, pp. 335–358.
[113] Y.-p. Chen, C.-H. Chen, Enabling the extended compact genetic algorithm for real-parameter optimization by using adaptive discretization, Evolutionary Computation 18 (2010) 199–228.
[114] C.-Y. Chuang, S. F. Smith, Diversity allocation for dynamic optimization using the extended compact genetic algorithm, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1540–1547.
[115] N. Hansen, The cma evolution strategy: a comparing review, in: Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms, Springer, 2006, pp. 75–102.
[116] J. A. Lozano, P. Larrañaga, E. Bengoetxea, I. Inza, Towards a new evolutionary computation: advances on estimation of distribution algorithms, Springer Science & Business Media, 2006.
[117] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, Swarm and Evolutionary Computation 1 (2011) 111–128.
[118] M. Pelikan, M. W. Hauschild, F. G. Lobo, Estimation of distribution algorithms, in: Springer Handbook of Computational Intelligence, Springer Berlin Heidelberg, 2015, pp. 899–928.
[119] Z. Xinchao, Simulated annealing algorithm with adaptive neighborhood, Applied Soft Computing 11 (2011) 1827–1836.
[120] A. El Afia, M. Lalaoui, E.-g. Talbi, Quaternion simulated annealing, in: Heuristics for Optimization and Learning, Springer, 2021, pp. 299–314.
[121] Z. Ji, J.-R. Zhou, H.-L. Liao, Q.-H. Wu, A novel intelligent single particle optimizer, Chinese Journal of computers 33 (2010) 556–561.
[122] J. Zhou, Z. Ji, W. Huang, T. Tian, Face recognition using gabor wavelet and self-adaptive intelligent single particle optimizer, in: 2010 Chinese Conference on Pattern Recognition (CCPR), IEEE, 2010, pp. 1–5.
[123] G. Iacca, F. Caraffini, F. Neri, E. Mininno, Single particle algorithms for continuous optimization, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1610–1617.
[124] F. Caraffini, F. Neri, B. N. Passow, G. Iacca, Re-sampled inheritance search: high performance despite the simplicity, Soft Computing 17 (2013) 2235–2256.
[125] G. Iacca, F. Neri, E. Mininno, Y.-S. Ong, M.-H. Lim, Ockham's razor in memetic computing: three stage optimal memetic exploration, Information Sciences 188 (2012) 17–43.
[126] L.-Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 3052–3059.
[127] N. H. A. Aziz, Z. Ibrahim, N. A. Ab Aziz, M. S. Mohamad, J. Watada, Single-solution simulated kalman filter algorithm for global optimisation problems, Sādhanā 43 (2018) 103.
[128] T. Ab Rahman, Z. Ibrahim, N. A. A. Aziz, S. Zhao, N. H. A. Aziz, Single-agent finite impulse response optimizer for numerical optimization problems, IEEE Access 6 (2018) 9358–9374.
[129] C. Igel, T. Suttorp, N. Hansen, A computational efficient covariance matrix update and a (1+ 1)-cma for evolution strategies, in: Proceedings of the 8th annual conference on Genetic and evolutionary computation, Association for Computing Machinery, 2006, pp. 453–460.

[130] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747.
[131] J. C. Spall, An overview of the simultaneous perturbation method for efficient optimization, Johns Hopkins apl technical digest 19 (1998) 482–492.
[132] K.-C. Hu, C.-W. Tsai, M.-C. Chiang, A multiple-search multi-start framework for metaheuristics for clustering problems, IEEE Access 8 (2020) 96173–96183.
[133] B. Doğan, T. Ölmez, A new metaheuristic for numerical function optimization: Vortex search algorithm, Information Sciences 293 (2015) 125–145.
[134] G. Shi, Q. Ren, Research on compact genetic algorithm in continuous domain, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 793–800.
[135] E. Mininno, F. Cupertino, D. Naso, Real-valued compact genetic algorithms for embedded microcontroller optimization, IEEE Transactions on Evolutionary Computation 12 (2008) 203–219.
[136] C. Jiang, X. Xue, A uniform compact genetic algorithm for matching bibliographic ontologies, Applied Intelligence 51 (2021) 7517–7532.
[137] Y. Wang, H. Yao, L. Wan, H. Li, J. Jiang, Y. Zhang, F. Wu, J. Chen, X. Xue, C. Dai, Optimizing hydrography ontology alignment through compact particle swarm optimization algorithm, in: International Conference on Swarm Intelligence, Springer, 2020, pp. 151–162.
[138] E. Mininno, F. Neri, F. Cupertino, D. Naso, Compact differential evolution, IEEE Transactions on Evolutionary Computation 15 (2011) 32–54.
[139] F. Neri, G. Iacca, E. Mininno, Disturbed exploitation compact differential evolution for limited memory optimization problems, Information Sciences 181 (2011) 2469–2487.
[140] G. Iacca, F. Neri, E. Mininno, Opposition-based learning in compact differential evolution, in: European Conference on the Applications of Evolutionary Computation, Springer, 2011, pp. 264–273.
[141] X. Wang, G. Xu, Robot path planning based on chaos concise differential evolution and rfnn control, The Open Automation and Control Systems Journal 6 (2014) 69–76.
[142] G. Iacca, F. Caraffini, F. Neri, Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead, Journal of Computer Science and technology 27 (2012) 1056–1076.
[143] S. Khalfi, A. Draa, G. Iacca, A compact compound sinusoidal differential evolution algorithm for solving optimisation problems in memory-constrained environments, Expert Systems with Applications 186 (2021) 115705.
[144] S. Khosravi, T. M.-R. Akbarzadeh, Real-parameter compact supervision for the particle swarm optimization (rcspso), in: 2014 Iranian Conference on Intelligent Systems (ICIS), IEEE, 2014, pp. 1–6.
[145] W.-M. Zheng, N. Liu, Q.-W. Chai, S.-C. Chu, A compact adaptive particle swarm optimization algorithm in the application of the mobile sensor localization, Wireless Communications and Mobile Computing 2021 (2021) 1–15.
[146] G. Iacca, F. Neri, E. Mininno, Compact bacterial foraging optimization, in: Swarm and Evolutionary Computation, Springer, 2012, pp. 84–92.
[147] T.-K. Dao, S.-C. Chu, C.-S. Shieh, M.-F. Horng, Compact artificial bee colony, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2014, pp. 96–105.
[148] A. Banitalebi, M. I. A. Aziz, A. Bahar, Z. A. Aziz, Enhanced compact artificial bee colony, Information Sciences 298 (2015) 491–511.
[149] J.-S. Pan, T.-K. Dao, A compact bat algorithm for unequal clustering in wireless sensor networks, Applied Sciences 9 (2019) 1973.
[150] L. Tighzert, B. Mendil, Cfo: A new compact swarm intelligent algorithm for global optimization and optimal bipedal robots walking, in: 2016 8th International Conference on Modelling, Identification and Control (ICMIC), IEEE, 2016, pp. 487–492.
[151] L. Tighzert, C. Fonlupt, B. Mendil, Towards compact swarm intelligence: a new compact firefly optimisation technique, International journal of computer applications in technology 60 (2019) 108–123.
[152] M. Zhao, A novel compact cat swarm optimization based on differential method, Enterprise Information Systems 14 (2020) 196–220.
[153] A.-Q. Tian, S.-C. Chu, J.-S. Pan, H. Cui, W.-M. Zheng, A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station, Sustainability 12 (2020) 767.
[154] T.-K. Dao, T.-S. Pan, T.-T. Nguyen, S.-C. Chu, J.-S. Pan, A compact flower pollination algorithm optimization, in: 2016 Third International Conference on Computing Measurement Control and Sensor Network (CMCSN), IEEE, 2016, pp. 76–79.
[155] J.-S. Pan, T.-K. Dao, T.-S. Pan, T.-T. Nguyen, S.-C. Chu, J. F. Roddick, An improvement of flower pollination algorithm for node localization optimization in wsn., J. Inf. Hiding Multim. Signal Process. 8 (2017) 486–499.
[156] J.-S. Pan, P.-C. Song, S.-C. Chu, Y.-J. Peng, Improved compact cuckoo search algorithm applied to location of drone logistics hub, Mathematics 8 (2020) 333.

[157] J.-S. Pan, Q.-y. Yang, S.-C. Chu, K.-C. Chang, Compact sine cosine algorithm applied in vehicle routing problem with time window, Telecommunication Systems 78 (2021) 609–628.
[158] X. W. Xu, T. S. Pan, P. C. Song, C. C. Hu, S. C. Chu, Multi-cluster based equilibrium optimizer algorithm with compact approach for power system network, Journal of Network Intelligence 6 (2021) 117–142.
[159] Z. Yang, K. Li, Y. Guo, H. Ma, M. Zheng, Compact real-valued teaching-learning based optimization with the applications to neural network training, Knowledge-Based Systems 159 (2018) 51–62.
[160] Z. Yu, J. Du, G. Li, Compact harris hawks optimization algorithm, in: 2021 40th Chinese Control Conference (CCC), IEEE, 2021, pp. 1925–1930.
[161] F. Lachouri, A. Khelifi, L. Tighzert, T. Aguercif, B. Mendil, Self-stunding up of humanoid robot using a new intelligent algorithm, in: 2016 8th International Conference on Modelling, Identification and Control (ICMIC), IEEE, 2016, pp. 903–908.
[162] D. S. Johnson, C. H. Papadimitriou, M. Yannakakis, How easy is local search?, Journal of computer and system sciences 37 (1988) 79–100.
[163] B. Chopard, M. Tomassini, An introduction to metaheuristics for optimization, Springer, 2018.
[164] M. A. Al-Betar, A β-hill climbing optimizer for examination timetabling problem, Journal of Ambient Intelligence and Humanized Computing 12 (2021) 653–666.
[165] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search, in: Handbook of metaheuristics, Springer, 2003, pp. 320–353.
[166] T. Song, S. Liu, X. Tang, X. Peng, M. Chen, An iterated local search algorithm for the university course timetabling problem, Applied Soft Computing 68 (2018) 597–608.
[167] V. R. Máximo, M. C. Nascimento, A hybrid adaptive iterated local search with diversification control to the capacitated vehicle routing problem, European Journal of Operational Research 294 (2021) 1108–1119.
[168] J. Brandão, Iterated local search algorithm with ejection chains for the open vehicle routing problem with time windows, Computers & Industrial Engineering 120 (2018) 146–159.
[169] Z. Wei, J.-K. Hao, Iterated two-phase local search for the set-union knapsack problem, Future Generation Computer Systems 101 (2019) 1005–1017.
[170] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search: Framework and applications, in: Handbook of metaheuristics, Springer, 2019, pp. 129–168.
[171] U. Benlic, J.-K. Hao, Breakout local search for the vertex separator problem, in: Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI/AAAI, 2013, pp. 461–467.
[172] U. Benlic, J.-K. Hao, Breakout local search for the quadratic assignment problem, Applied Mathematics and Computation 219 (2013) 4800–4815.
[173] U. Benlic, J.-K. Hao, Breakout local search for maximum clique problems, Computers & Operations Research 40 (2013) 192–206.
[174] S. Ghandi, E. Masehian, A breakout local search (bls) method for solving the assembly sequence planning problem, Engineering Applications of Artificial Intelligence 39 (2015) 245–266.
[175] U. Benlic, M. G. Epitropakis, E. K. Burke, A hybrid breakout local search and reinforcement learning approach to the vertex separator problem, European Journal of Operational Research 261 (2017) 803–818.
[176] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in: International conference on principles and practice of constraint programming, Springer, 1998, pp. 417–431.
[177] D. Sacramento, D. Pisinger, S. Ropke, An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones, Transportation Research Part C: Emerging Technologies 102 (2019) 289–315.
[178] C. Friedrich, R. Elbert, Adaptive large neighborhood search for vehicle routing problems with transshipment facilities arising in city logistics, Computers & Operations Research 137 (2022) 105491.
[179] Y. Li, M. Chen, J. Huo, A hybrid adaptive large neighborhood search algorithm for the large-scale heterogeneous container loading problem, Expert Systems with Applications 189 (2022) 115909.
[180] G. Dueck, New optimization heuristics: The great deluge algorithm and the record-to-record travel, Journal of Computational physics 104 (1993) 86–92.
[181] M. Mohmad Kahar, G. Kendall, A great deluge algorithm for a real-world examination timetabling problem, Journal of the Operational Research Society 66 (2015) 116–133.
[182] E. K. Burke, Y. Bykov, An adaptive flex-deluge approach to university exam timetabling, INFORMS Journal on Computing 28 (2016) 781–794.
[183] E. T. Yassen, M. Ayob, M. Z. A. Nazri, N. R. Sabar, An adaptive hybrid algorithm for vehicle routing problems with time windows, Computers & Industrial Engineering 113 (2017) 382–391.
[184] K. Eng, A. Muhammed, M. A. Mohamed, S. Hasan, A hybrid heuristic of variable neighbourhood descent and great deluge algorithm for efficient task scheduling in grid computing, European Journal of Operational Research 284

(2020) 75–86.
[185] N. Mladenović, P. Hansen, Variable neighborhood search, Computers & operations research 24 (1997) 1097–1100.
[186] P. Hansen, N. Mladenović, J. A. M. Pérez, Variable neighbourhood search: methods and applications, Annals of Operations Research 175 (2010) 367–407.
[187] T. A. Feo, M. G. Resende, Greedy randomized adaptive search procedures, Journal of global optimization 6 (1995) 109–133.
[188] P. Festa, M. G. Resende, Grasp: basic components and enhancements, Telecommunication Systems 46 (2011) 253–271.
[189] C. Voudouris, Guided local search for combinatorial optimisation problems., Ph.D. thesis, University of Essex (1997).
[190] C. Voudouris, E. P. Tsang, A. Alsheddy, Guided local search, in: Handbook of metaheuristics, Springer, 2010, pp. 321–361.
[191] A. Alsheddy, C. Voudouris, E. P. K. Tsang, A. Alhindi, Guided local search, in: Handbook of Heuristics, Springer, 2016, pp. 1–37.
[192] Y. Zhou, J.-K. Hao, B. Duval, Reinforcement learning based local search for grouping problems: A case study on graph coloring, Expert Systems with Applications 64 (2016) 412–422.
[193] A. Amuthan, K. D. Thilak, Survey on tabu search meta-heuristic optimization, in: 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), IEEE, 2016, pp. 1539–1543.
[194] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers operations research 13 (1986) 533–549.
[195] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, science 220 (1983) 671–680.
[196] K. A. Dowsland, J. Thompson, Simulated annealing, in: Handbook of Natural Computing, Springer, 2012, pp. 1623–1655.
[197] G. Dueck, T. Scheuer, T.-o. Accepting, Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing, Journal of Computation Physics 90 (1990) 161–175.
[198] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: International conference on the practice and theory of automated timetabling, Springer, 2000, pp. 176–190.
[199] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J. R. Woodward, A classification of hyper-heuristic approaches, in: Handbook of metaheuristics, Springer, 2010, pp. 449–468.
[200] N. Pillay, R. Qu, Hyper-heuristics: theory and applications, Springer, 2018.
[201] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, J. R. Woodward, A classification of hyper-heuristic approaches: revisited, in: Handbook of metaheuristics, Springer, 2019, pp. 453–477.
[202] J. H. Drake, A. Kheiri, E. Özcan, E. K. Burke, Recent advances in selection hyper-heuristics, European Journal of Operational Research 285 (2020) 405–428.
[203] M. Sánchez, J. M. Cruz-Duarte, J. carlos Ortíz-Bayliss, H. Ceballos, H. Terashima-Marin, I. Amaya, A systematic review of hyper-heuristics on combinatorial optimization problems, IEEE Access 8 (2020) 128068–128095.
[204] N. S. Jaddi, S. Abdullah, Global search in single-solution-based metaheuristics, Data Technologies and Applications 54 (2020) 275–296.
[205] R. Hooke, T. A. Jeeves, "direct search"solution of numerical and statistical problems, Journal of the ACM (JACM) 8 (1961) 212–229.
[206] F. J. Solis, R. J.-B. Wets, Minimization by random search techniques, Mathematics of Operations Research 6 (1981) 19–30.
[207] A. Auger, Benchmarking the (1+ 1) evolution strategy with one-fifth success rule on the bbob-2009 function testbed, in: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, Association for Computing Machinery, 2009, pp. 2447–2452.
[208] Z. Ji, H. Liao, Y. Wang, Q. Wu, A novel intelligent particle optimizer for global optimization of multimodal functions, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 3272–3275.
[209] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.
[210] J. Zhou, Z. Ji, L. Shen, Z. Zhu, S. Chen, Pso based memetic algorithm for face recognition gabor filters selection, in: 2011 IEEE Workshop on Memetic Computing (MC), IEEE, 2011, pp. 1–6.
[211] Z. Zhu, J. Zhou, Z. Ji, Y.-H. Shi, Dna sequence compression using adaptive particle swarm optimization-based memetic algorithm, IEEE Transactions on Evolutionary Computation 15 (2011) 643–658.
[212] Z. Ji, J. Zhou, Z. Zhu, S. Chen, Self-configuration single particle optimizer for dna sequence compression, Soft Computing 17 (2013) 675–682.
[213] F. Caraffini, G. Iacca, F. Neri, E. Mininno, The importance of being structured: a comparative study on multi stage memetic approaches, in: 2012 12th UK Workshop on Computational Intelligence (UKCI), IEEE, 2012, pp. 1–8.

[214] I. Poikolainen, G. Iacca, F. Caraffini, F. Neri, Focusing the search: a progressively shrinking memetic computing framework, International Journal of Innovative Computing and Applications 5 5 (2013) 127–142.
[215] F. Caraffini, G. Iacca, F. Neri, E. Mininno, Three variants of three stage optimal memetic exploration for handling non-separable fitness landscapes, in: 2012 12th UK Workshop on Computational Intelligence (UKCI), IEEE, 2012, pp. 1–8.
[216] F. Caraffini, F. Neri, M. Gongora, B. N. Passow, Re-sampling search: A seriously simple memetic approach with a high performance, in: 2013 IEEE Workshop on Memetic Computing (MC), IEEE, 2013, pp. 52–59.
[217] G. Iacca, F. L. Bakker, H. Wörtche, Real-time magnetic dipole detection with single particle optimization, Applied soft computing 23 (2014) 460–473.
[218] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Information Sciences 227 (2013) 60–82.
[219] F. Caraffini, F. Neri, M. Epitropakis, Hyperspam: A study on hyper-heuristic coordination strategies in the continuous domain, Information Sciences 477 (2019) 186–202.
[220] N. H. A. Aziz, Z. Ibrahim, N. A. Ab Aziz, B. Muhammad, T. Ab Rahman, M. S. Mohamad, S. A. Rahmad, Parameter tuning in the single-solution simulated kalman filter optimizer, in: Symposium on Intelligent Manufacturing and Mechatronics, Springer, 2019, pp. 48–56.
[221] A. Ibrahim, S. Rahnamayan, M. V. Martin, Simulated raindrop algorithm for global optimization, in: 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, 2014, pp. 1–8.
[222] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering–a systematic literature review, Information and software technology 51 (2009) 7–15.
[223] N. Donthu, S. Kumar, D. Mukherjee, N. Pandey, W. M. Lim, How to conduct a bibliometric analysis: An overview and guidelines, Journal of business research 133 (2021) 285–296.
[224] G. R. Harik, F. G. Lobo, D. E. Goldberg, The compact genetic algorithm, IEEE transactions on evolutionary computation 3 (1999) 287–297.
[225] C. Zhou, K. Meng, Z. Qiu, Compact genetic algorithm mutated by bit, in: Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527), IEEE, 2002, pp. 1836–1839.
[226] C. W. Ahn, R. S. Ramakrishna, Elitism-based compact genetic algorithms, IEEE Transactions on Evolutionary Computation 7 (2003) 367–385.
[227] S. Rimcharoen, D. Sutivong, P. Chongstitvatana, Updating strategy in compact genetic algorithm using moving average approach, in: 2006 IEEE Conference on Cybernetics and Intelligent Systems, IEEE, 2006, pp. 1–6.
[228] R. R. Silva, H. S. Lopes, C. R. E. Lima, A new mutation operator for the elitism-based compact genetic algorithm, in: International Conference on Adaptive and Natural Computing Algorithms, Springer, 2007, pp. 159–166.
[229] S. Phiromlap, S. Rimcharoen, A frequency-based updating strategy in compact genetic algorithm, in: 2013 International Computer Science and Engineering Conference (ICSEC), IEEE, 2013, pp. 207–211.
[230] W. J. Cody, Rational chebyshev approximations for the error function, Mathematics of Computation 23 (1969) 631–637.
[231] L. Tighzert, C. Fonlupt, O. Bruneau, B. Mendil, A new uniform compact evolutionary algorithms, in: 2017 5th International Conference on Electrical Engineering-Boumerdes (ICEE-B), IEEE, 2017, pp. 1–6.
[232] C. W. Ahn, R. S. Ramakrishna, Augmented compact genetic algorithm, in: International Conference on Parallel Processing and Applied Mathematics, Springer, 2003, pp. 560–565.
[233] J.-H. Seok, J.-J. Lee, A novel compact genetic algorithm using offspring survival evolutionary strategy, Artificial Life and Robotics 14 (2009) 489–493.
[234] F. Cupertino, E. Mininno, D. Naso, Elitist compact genetic algorithms for induction motor self-tuning control, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 3057–3063.
[235] J.-Y. Lee, M.-S. Kim, J.-J. Lee, Compact genetic algorithms using belief vectors, Applied Soft Computing 11 (2011) 3385–3401.
[236] Q.-b. Zhang, T.-h. Wu, B. Liu, A weight based compact genetic algorithm, in: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, ACM, 2009, pp. 1057–1060.
[237] B. Van Ha, R. Zich, M. Mussetta, P. Pirinoli, C. N. Dao, Improved compact genetic algorithm for em complex system design, in: 2012 Fourth International Conference on Communications and Electronics (ICCE), IEEE, 2012, pp. 389–392.
[238] B. Ha, M. Mussetta, P. Pirinoli, R. Zich, Modified compact genetic algorithm for thinned array synthesis, IEEE Antennas and Wireless Propagation Letters 15 (2015) 1105–1108.
[239] Z. Han, Y. Zhu, S. Lin, A dynamic co-evolution compact genetic algorithm for e/t problem, IFAC-PapersOnLine 48 (2015) 1439–1443.
[240] S. Rimcharoen, S. Phiromlap, N. Leelathakul, Analysis of frequency-based compact genetic algorithm (fb-cga), Maejo International Journal of Science and Technology 9 (2015) 121–135.

[241] B. Doerr, W. Zheng, From understanding genetic drift to a smart-restart parameter-less compact genetic algorithm, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, Association for Computing Machinery, 2020, pp. 805–813.
[242] H. Xing, R. Qu, A compact genetic algorithm for the network coding based resource minimization problem, Applied Intelligence 36 (2012) 809–823.
[243] R. D. Al-Dabbagh, A. Kinsheel, M. S. Baba, S. Mekhilef, A combined compact genetic algorithm and local search method for optimizing the arma (1, 1) model of a likelihood estimator, Sci. Asia 40 (2014) 78–86.
[244] R. Baraglia, J. I. Hidalgo, R. Perego, A hybrid heuristic for the traveling salesman problem, IEEE Transactions on evolutionary computation 5 (2001) 613–622.
[245] J. I. Hidalgo, J. Lanchares, A. Ibarra, R. Hermida, A hybrid evolutionary algorithm for multi-fpga systems design, in: Proceedings Euromicro Symposium on Digital System Design. Architectures, Methods and Tools, IEEE, 2002, pp. 60–67.
[246] J. I. Hidalgo, M. Prieto, J. Lanchares, R. Baraglia, F. Tirado, O. Garnica, Hybrid parallelization of a compact genetic algorithm, in: Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2003. Proceedings., IEEE, 2003, pp. 449–455.
[247] Y. Jewajinda, P. Chongstitvatana, A cooperative approach to compact genetic algorithm for evolvable hardware, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 2779–2786.
[248] K. Sastry, D. E. Goldberg, X. Llora, Towards billion-bit optimization via a parallel estimation of distribution algorithm, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, Association for Computing Machinery, 2007, pp. 577–584.
[249] K. E. Duncan, S. K. Boddhu, M. Sam, J. C. Gallagher, Islands of fitness compact genetic algorithm for rapid in-flight control learning in a flapping-wing micro air vehicle: A search space reduction approach, in: 2014 IEEE International Conference on Evolvable Systems, IEEE, 2014, pp. 219–226.
[250] J. C. Gallagher, An islands-of-fitness compact genetic algorithm approach to improving learning time in swarms of flapping-wing micro air vehicles, in: Robot Intelligence Technology and Applications 2012, Springer, 2013, pp. 855–862.
[251] J. I. Hidalgo, R. Baraglia, R. Perego, J. Lanchares, F. Tirado, A parallel compact genetic algorithm for multi-fpga partitioning, in: Proceedings Ninth Euromicro Workshop on Parallel and Distributed Processing, IEEE, 2001, pp. 113–120.
[252] F. G. Lobo, C. F. Lima, H. Mártires, An architecture for massive parallelization of the compact genetic algorithm, in: Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 412–413.
[253] C. Aporntewan, P. Chongstitvatana, A hardware implementation of the compact genetic algorithm, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), IEEE, 2001, pp. 624–629.
[254] Y. Jewajinda, P. Chongstitvatana, Fpga implementation of a cellular compact genetic algorithm, in: 2008 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE, 2008, pp. 385–390.
[255] Y. Jewajinda, P. Chongstitvatana, Cellular compact genetic algorithm for evolvable hardware, in: 2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, IEEE, 2008, pp. 1–4.
[256] M. A. Moreno-Armendáriz, N. Cruz-Cortés, A. León-Javier, A novel hardware implementation of the compact genetic algorithm, in: 2010 International Conference on Reconfigurable Computing and FPGAs, IEEE, 2010, pp. 156–161.
[257] M. A. Moreno-Armendáriz, N. Cruz-Cortés, C. A. Duchanoy, A. León-Javier, R. Quintero, Hardware implementation of the elitist compact genetic algorithm using cellular automata pseudo-random number generator, Computers & Electrical Engineering 39 (2013) 1367–1379.
[258] A. Ferigo, G. Iacca, A GPU-enabled compact genetic algorithm for very large-scale optimization problems, Mathematics 8 (2020) 758.
[259] K. Deb, C. Myburgh, Breaking the billion-variable barrier in real-world optimization using a customized evolutionary algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, Association for Computing Machinery, 2016, pp. 653–660.
[260] T. Friedrich, T. Kötzing, M. S. Krejca, A. M. Sutton, The compact genetic algorithm is efficient under extreme Gaussian noise, IEEE Transactions on Evolutionary Computation 21 (2016) 477–490.
[261] V. Hasenöhrl, A. M. Sutton, On the runtime dynamics of the compact genetic algorithm on jump functions, in: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, 2018, pp. 967–974.
[262] B. Doerr, The runtime of the compact genetic algorithm on jump functions, Algorithmica 83 (2021) 3059–3107.
[263] B. Doerr, An exponential lower bound for the runtime of the compact genetic algorithm on jump functions, in: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, Association for Computing

Machinery, 2019, pp. 25–33.
[264] S. Droste, A rigorous analysis of the compact genetic algorithm for linear functions, Natural Computing 5 (2006) 257–283.
[265] J. Lengler, D. Sudholt, C. Witt, Medium step sizes are harmful for the compact genetic algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, 2018, pp. 1499–1506.
[266] J. Lengler, D. Sudholt, C. Witt, The complex parameter landscape of the compact genetic algorithm, Algorithmica 83 (2021) 1096–1137.
[267] F. Corno, M. S. Reorda, G. Squillero, The selfish gene algorithm: a new evolutionary optimization strategy, in: Proceedings of the 1998 ACM symposium on Applied Computing, Association for Computing Machinery, 1998, pp. 349–355.
[268] R. Dawkins, The selfish gene, Oxford University Press, 1976.
[269] L. Tighzert, B. Mendil, Realization of gymnastic movements on the bar by humanoid robot using a new selfish gene algorithm, in: Modelling and Implementation of Complex Systems, Springer, 2016, pp. 49–65.
[270] L. Tighzert, T. Aguercif, C. Fonlupt, B. Mendil, Intelligent trajectory planning and control of a humanoid robot using a new elitism-based selfish gene algorithm, in: 2017 6th International Conference on Systems and Control (ICSC), IEEE, 2017, pp. 514–519.
[271] N. M. Ariff, N. E. A. Khalid, R. Hashim, N. M. Noor, Selfish gene algorithm vs genetic algorithm: A review, in: IOP Conference Series: Materials Science and Engineering, IOP Publishing, 2016, p. 012098.
[272] A. M. Cruz, R. B. Fernández, H. M. Lozano, M. A. R. Salinas, L. A. V. Vargas, Automated functional test generation for digital systems through a compact binary differential evolution algorithm, Journal of Electronic Testing 31 (2015) 361–380.
[273] X. Xue, J. Chen, Matching biomedical ontologies through compact differential evolution algorithm, Systems Science & Control Engineering 7 (2019) 85–89.
[274] X. Xue, J. Chen, Matching biomedical ontologies through compact differential evolution algorithm with compact adaption schemes on control parameters, Neurocomputing 458 (2021) 526–534.
[275] Y. Huang, X. Xue, C. Jiang, Semantic integration of sensor knowledge on artificial internet of things, Wireless Communications and Mobile Computing 2020 (2020) 1–8.
[276] X. Xue, J. Chen, Optimizing sensor ontology alignment through compact co-firefly algorithm, Sensors 20 (2020) 2056.
[277] X. Xue, J. Chen, A compact co-firefly algorithm for matching ontologies, in: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2019, pp. 2629–2632.
[278] X. Xue, P.-W. Tsai, J. Wang, Using compact memetic algorithm for optimizing ontology alignment, ICIC Express Lett 11 (2017) 53–58.
[279] X. Xue, J.-S. Pan, A compact co-evolutionary algorithm for sensor ontology meta-matching, Knowledge and Information Systems 56 (2018) 335–353.
[280] S.-C. Chu, X. Xue, J.-S. Pan, X. Wu, Optimizing ontology alignment in vector space, Journal of Internet Technology 21 (2020) 15–22.
[281] K. Suksen, P. Chongstitvatana, Exploiting building blocks in hard problems with modified compact genetic algorithm, in: 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, 2018, pp. 1–6.
[282] Y.-C. Huang, C.-F. Chang, C.-h. Chan, T.-J. Yeh, Y.-C. Chang, C.-C. Chen, C.-Y. Kao, Integrated minimum-set primers and unique probe design algorithms for differential detection on symptom-related pathogens, Bioinformatics 21 (2005) 4330–4337.
[283] P. Kumar Singh, N. Sahli, Task scheduling in grid computing environment using compact genetic algorithm, Int. J. Sci. Eng. Technol. Res.(IJSETR) 3 (2014) 107–110.
[284] A. Badr, I. M. Aref, B. M. Hussien, Y. Eman, Solving protein folding problem using elitism-based compact genetic algorithm, Journal of Computer Science 4 (2008) 525–529.
[285] R. R. Silva, H. S. Lopes, C. R. E. Lima, A compact genetic algorithm with elitism and mutation applied to image recognition, in: International Conference on Intelligent Computing, Springer, 2008, pp. 1109–1116.
[286] R. R. Da Silva, C. R. ERIG LIMA, H. S. LOPES, Template matching in digital images using a compact genetic algorithm with elitism and mutation, Journal of Circuits, Systems, and Computers 19 (2010) 91–106.
[287] H. Boualame, N. Tahiri, I. Chana, A. Azouaoui, M. Belkasmi, An efficient soft decision decoding algorithm using cyclic permutations and compact genetic algorithm, in: 2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS), IEEE, 2016, pp. 1–6.
[288] A. Berkani, M. Belkasmi, A reduced complexity decoder using compact genetic algorithm for linear block codes, in:

2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS), IEEE, 2016, pp. 1–6.
[289] R. D. Al-Dabbagh, M. S. Baba, S. Mekhilef, A. Kinsheel, The compact genetic algorithm for likelihood estimator of first order moving average model, in: 2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), IEEE, 2012, pp. 474–481.
[290] X. Xue, J. Liu, P.-W. Tsai, X. Zhan, A. Ren, Optimizing ontology alignment by using compact genetic algorithm, in: 2015 11th International Conference on Computational Intelligence and Security (CIS), IEEE, 2015, pp. 231–234.
[291] X. Xue, X. Wu, J. Chen, Optimizing ontology alignment through an interactive compact genetic algorithm, ACM Transactions on Management Information Systems (TMIS) 12 (2021) 1–17.
[292] N. Tahiri, A. Azouaoui, M. Belkasmi, A novel detector based on the compact genetic algorithm for mimo systems, in: 2018 International Conference on Advanced Communication Technologies and Networking (CommNet), IEEE, 2018, pp. 1–6.
[293] G. Iacca, R. Mallipeddi, E. Mininno, F. Neri, P. N. Suganthan, Global supervision for compact differential evolution, in: 2011 IEEE Symposium on Differential Evolution (SDE), IEEE, 2011, pp. 1–8.
[294] A. Sergio, S. Carvalho, R. Marco, On the use of compact approaches in evolution strategies, ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal 3 (2014) 13–23.
[295] X. Cheng, Y. Jiang, D. Li, Z. Zhu, N. Wu, Optimal operation with parallel compact bee colony algorithm for cascade hydropower plants, Journal of Network Intelligence 6 (2021) 440–452.
[296] S. Zhang, F. Fan, W. Li, S.-C. Chu, J.-S. Pan, A parallel compact sine cosine algorithm for tdoa localization of wireless sensor network, Telecommunication Systems 78 (2021) 213–223.
[297] M. Zhu, S.-C. Chu, Q. Yang, W. Li, J.-S. Pan, Compact sine cosine algorithm with multigroup and multistrategy for dispatching system of public transit vehicles, Journal of Advanced Transportation 2021 (2021) 1–16.
[298] K. M. Timmerman, A hardware compact genetic algorithm for hover improvement in an insect-scale flapping-wing micro air vehicle, Master's thesis, Wright State University, USA (2012).
[299] H. J. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. L. Jerez, G. Stathopoulos, C. Jones, Embedded optimization methods for industrial automatic control, IFAC-PapersOnLine 50 (2017) 13194–13209.
[300] X. Xue, J. Liu, Collaborative ontology matching based on compact interactive evolutionary algorithm, Knowledge-Based Systems 137 (2017) 94–103.
[301] T.-T. Nguyen, S.-C. Chu, T.-K. Dao, T.-D. Nguyen, T.-G. Ngo, An optimal deployment wireless sensor network based on compact differential evolution., J. Netw. Intell. 2 (2017) 263–274.
[302] T.-S. Pan, T.-T. Nguyen, T.-K. Dao, S.-C. Chu, An optimal clustering formation for wireless sensor network based on compact genetic algorithm, in: 2015 Third International Conference on Robot, Vision and Signal Processing (RVSP), IEEE, 2015, pp. 294–299.
[303] J.-S. Pan, T.-K. Dao, T.-T. Nguyen, T.-S. Pan, Compact particle swarm optimization for optimal location of base station in wireless sensor network, in: Genetic and Evolutionary Computing: Proceedings of the Tenth International Conference on Genetic and Evolutionary Computing, November 7-9, 2016 Fuzhou City, Fujian Province, China 10, Springer, 2017, pp. 54–62.
[304] T.-K. Dao, T.-S. Pan, T.-T. Nguyen, S.-C. Chu, A compact artificial bee colony optimization for topology control scheme in wireless sensor networks, Journal of Information Hiding and Multimedia Signal Processing 6 (2015) 297–310.
[305] F. Neri, E. Mininno, Memetic compact differential evolution for cartesian robot control, IEEE Computational Intelligence Magazine 5 (2010) 54–65.
[306] M. Toz, Chaos-based vortex search algorithm for solving inverse kinematics problem of serial robot manipulators with offset wrist, Applied Soft Computing 89 (2020) 106074.
[307] N. H. A. Aziz, Z. Ibrahim, N. A. Ab Aziz, Z. M. Yusof, M. S. Mohamad, Single-solution simulated kalman filter algorithm for routing in printed circuit board drilling process, in: Intelligent Manufacturing & Mechatronics, Springer, 2018, pp. 649–655.
[308] B. Doğan, A. Yüksel, Analog filter group delay optimization using the vortex search algorithm, in: 2015 23nd Signal Processing and Communications Applications Conference (SIU), IEEE, 2015, pp. 288–291.
[309] B. Doğan, T. Ölmez, Vortex search algorithm for the analog active filter component selection problem, AEU-International Journal of Electronics and Communications 69 (2015) 1243–1253.
[310] M. Zhao, J.-S. Pan, S.-T. Chen, Optimal snr of audio watermarking by wavelet and compact pso methods., J. Inf. Hiding Multim. Signal Process. 6 (2015) 833–846.
[311] M. Zhao, J.-S. Pan, S.-T. Chen, Entropy-based audio watermarking via the point of view on the compact particle swarm optimization, Journal of Internet Technology 16 (2015) 485–493.
[312] H. Li, X. Mou, Z. Ji, H. Yu, Y. Li, L. Jiang, Miniature rfid tri-band cpw-fed antenna optimised using ispo algorithm, Electronics Letters 47 (2011) 161–162.

[313] H. Li, X. Mou, Z. Ji, H. Yu, Y. Li, L. Jiang, A novel miniature four-band cpw-fed antenna optimized using ispo algorithm, in: International Wireless Internet Conference, Springer, 2011, pp. 576–581.
[314] O. D. Montoya, W. Gil-González, L. F. Grisales-Noreña, Vortex search algorithm for optimal power flow analysis in dc resistive networks with cpls, IEEE Transactions on Circuits and Systems II: Express Briefs 67 (2019) 1439–1443.
[315] W. Ali, M. A. Qyyum, K. Qadeer, M. Lee, Energy optimization for single mixed refrigerant natural gas liquefaction process using the metaheuristic vortex search algorithm, Applied Thermal Engineering 129 (2018) 782–791.
[316] A. Fathy, M. Abd Elaziz, A. G. Alharbi, A novel approach based on hybrid vortex search algorithm and differential evolution for identifying the optimal parameters of pem fuel cell, Renewable Energy 146 (2020) 1833–1845.
[317] A. Zimmer, A. Schmidt, A. Ostfeld, B. Minsker, Evolutionary algorithm enhancement for model predictive control and real-time decision support, Environmental Modelling & Software 69 (2015) 330–341.
[318] H. Fauzi, U. Batool, A three-bar truss design using single-solution simulated kalman filter optimizer, Mekatronika 1 (2019) 98–102.
[319] C. Millán-Páramo, E. Millán-Romero, F. J. Wilches, Truss optimization with natural frequency constraints using modified social engineering optimizer, International Journal of Engineering Research and Technology 13 (2020) 3950–3963.
[320] F. S. Gharehchopogh, I. Maleki, Z. A. Dizaji, Chaotic vortex search algorithm: metaheuristic algorithm for feature selection, Evolutionary Intelligence 15 (2021) 1–32.
[321] S. Iliya, E. Goodyer, J. Gow, J. Shell, M. Gongora, Application of artificial neural network and support vector regression in cognitive radio networks for rf power prediction using compact differential evolution algorithm, in: 2015 federated conference on computer science and information systems (FedCSIS), IEEE, 2015, pp. 55–66.
[322] J. Zhou, Z. Ji, L. Shen, Simplified intelligence single particle optimization based neural network for digit recognition, in: 2008 Chinese Conference on Pattern Recognition, IEEE, 2008, pp. 1–5.
[323] X. Li, P. Niu, J. Liu, Combustion optimization of a boiler based on the chaos and levy flight vortex search algorithm, Applied Mathematical Modelling 58 (2018) 3–18.
[324] S. Bhagat, S. K. Pasupuleti, Simulated raindrop algorithm to mitigate ddos attacks in cloud computing, in: Proceedings of the Sixth International Conference on Computer and Communication Technology 2015, Association for Computing Machinery, 2015, pp. 412–418.
[325] F. Caraffini, F. Neri, G. Iacca, Large scale problems in practice: The effect of dimensionality on the interaction among variables, in: European Conference on the Applications of Evolutionary Computation, Springer, 2017, pp. 636–652.
[326] Z. Laboudi, A. Moudjari, A. Saighi, A. Draa, S. Hadjadj, An adaptive context-aware optimization framework for multimedia adaptation service selection, Neural Computing and Applications 34 (2022) 14239–14251.
[327] C. Cruz, J. R. González, D. A. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, Soft Computing 15 (2011) 1427–1448.
[328] T. Macias-Escobar, B. Dorronsoro, L. Cruz-Reyes, N. Rangel-Valdez, C. Gómez-Santillán, A survey of hyper-heuristics for dynamic optimization problems, in: Intuitionistic and type-2 fuzzy logic enhancements in neural and optimization algorithms: Theory and applications, Springer, 2020, pp. 463–477.
[329] D. E. Goldberg, K. Sastry, X. Llorà, Toward routine billion-variable optimization using genetic algorithms, Complexity 12 (2007) 27–29.
[330] S. Iturriaga, S. Nesmachnow, Solving very large optimization problems (up to one billion variables) with a parallel evolutionary algorithm in cpu and gpu, in: 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, IEEE, 2012, pp. 267–272.
[331] J. E. Rowe, The benefits and limitations of voting mechanisms in evolutionary optimisation, in: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, Association for Computing Machinery, 2019, pp. 34–42.
[332] F. Neri, E. Mininno, T. Kärkkäinen, Noise analysis compact genetic algorithm, in: European Conference on the Applications of Evolutionary Computation, Springer, 2010, pp. 602–611.
[333] G. Iacca, F. Neri, E. Mininno, Noise analysis compact differential evolution, International Journal of Systems Science 43 (2012) 1248–1267.
[334] S. Rojas-Galeano, N. Rodriguez, A memory efficient and continuous-valued compact eda for large scale problems, in: Proceedings of the 14th annual conference on Genetic and evolutionary computation, Association for Computing Machinery, 2012, pp. 281–288.
[335] I. Moser, R. Chiong, A hooke-jeeves based memetic algorithm for solving dynamic optimisation problems, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2009, pp. 301–309.
[336] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), IEEE, 1999, pp. 1875–1882.

[337] I. Moser, R. Chiong, Dynamic function optimization: the moving peaks benchmark, in: Metaheuristics for Dynamic Optimization, Springer, 2013, pp. 35–59.
[338] A. Ahrari, S. Elsayed, R. Sarker, D. Essam, C. A. Coello Coello, A novel parametric benchmark generator for dynamic multimodal optimization, Swarm and Evolutionary Computation 65 (2021) 100924.
[339] X. Lin, W. Luo, P. Xu, Y. Qiao, S. Yang, Popdmmo: A general framework of population-based stochastic search algorithms for dynamic multimodal optimization, Swarm and Evolutionary Computation 68 (2022) 101011.
[340] G. R. Kramer, J. C. Gallagher, Improvements to the *CGA enabling online intrinsic evolution in compact EH devices, in: NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings., IEEE, 2003, pp. 225–231.
[341] C. J. Uzor, M. Gongora, S. Coupland, B. N. Passow, Adaptive mutation in dynamic environments, in: 2014 14th UK Workshop on Computational Intelligence (UKCI), IEEE, 2014, pp. 1–7.
[342] C. J. Uzor, M. Gongora, S. Coupland, B. N. Passow, Real-world dynamic optimization using an adaptive-mutation compact genetic algorithm, in: 2014 IEEE Symposium on computational intelligence in dynamic and uncertain environments (CIDUE), IEEE, 2014, pp. 17–23.
[343] C. J. Uzor, M. Gongora, S. Coupland, B. N. Passow, Adaptive-mutation compact genetic algorithm for dynamic environments, Soft Computing 20 (2016) 3097–3115.
[344] A. M. Fathollahi-Fard, A. Ahmadi, F. Goodarzian, N. Cheikhrouhou, A bi-objective home healthcare routing and scheduling problem considering patients' satisfaction in a fuzzy environment, Applied soft computing 93 (2020) 106385.
[345] A. M. Fathollahi-Fard, A. Ahmadi, S. M. Al-e Hashem, Sustainable closed-loop supply chain network for an integrated water supply and wastewater collection system under uncertainty, Journal of Environmental Management 275 (2020) 111277.
[346] M. Mojtahedi, A. M. Fathollahi-Fard, R. Tavakkoli-Moghaddam, S. Newton, Sustainable vehicle routing problem for coordinated solid waste management, Journal of Industrial Information Integration 23 (2021) 100220.
[347] A. Özkış, A. Babalık, A novel metaheuristic for multi-objective optimization problems: The multi-objective vortex search algorithm, Information Sciences 402 (2017) 124–148.
[348] J. M. O. Velazquez, C. A. Coello Coello, A. Arias-Montano, Multi-objective compact differential evolution, in: 2014 IEEE Symposium on Differential Evolution (SDE), IEEE, 2014, pp. 1–8.
[349] J. J. Montiel, C. A. Coello Coello, M. G. C. Tapia, A proposal of a multi-objective compact particle swarm optimizer, in: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2019, pp. 2269–2278.
[350] X. Xue, X. Wu, J. Chen, Optimizing biomedical ontology alignment through a compact multiobjective particle swarm optimization algorithm driven by knee solution, Discrete Dynamics in Nature and Society 2020 (2020) 1–10.
[351] M. Ji, Z. Jin, H. Tang, An improved simulated annealing for solving the linear constrained optimization problems, Applied mathematics and computation 183 (2006) 251–259.
[352] C. S. Pedamallu, L. Ozdamar, Investigating a hybrid simulated annealing and local search algorithm for constrained optimization, European Journal of Operational Research 185 (2008) 1230–1245.
[353] C. Villalón, T. Stützle, M. Dorigo, Cuckoo search≡(μ+ λ)–evolution strategy, Tech. rep., IRIDIA (2021).
[354] M. A. Azzam, U. Batool, H. Fauzi, Design of an helical spring using single-solution simulated kalman filter optimizer, MEKATRONIKA 1 (2019) 93–97.

Table 1: Application fields of lightweight algorithms

Application	**Study**	**Algorithm(s)**
Training neural network for digit recognition	[322]	SISPO (Simplified ISPO which is equivalent to AdpISPO)
Optimal antennas for Radio Frequency IDentification (RFID) applications	[312, 313]	ISPO
DNA sequence compression	[211, 212]	AdpISPO & hybrid CLPSO + AdpISPO
Optimal Gabor features for face recognition	[122, 210]	
Optimal control system for a helicopter robot	[124]	RIS
Digital Infinite Impulse Response filter design	[125]	3SOME
Online PID tuning on path-following robot	[22]	3SOME
Real-time detection of a magnetic dipole	[217]	Single particle optimization with Restarted Exploration (SPORE)
Routing in printed circuit hole drilling	[307]	ssSKF
Three-bar truss design problem	[318]	ssSKF
Helical spring design problem	[354]	ssSKF
Optimisation of the analog filter group delay	[308]	VS
Analogue active filter component selection	[309]	
Energy efficient refrigerant gas liquefaction	[315]	
Optimal power flow in resistive networks with constant power loads	[314]	A hybrid master-slave VS algorithm with a power flow method
Optimal exchange membrane fuel cell	[316]	VSDE
Text author identification	[320]	Ten chaos theory enhanced VS algorithms
Inverse kinematics problem solution	[306]	
Optimised fast learning network	[323]	Three VS variants
Truss frequency constrained optimisation	[319]	A modified SEO algorithm
Distributed denial of services mitigation in cloud computing	[324]	SRD / SRD
Clustering and codebook problems	[132]	MSMS
Online optimal anti-windup PI speed controller	[135]	(nonpersistent elitist) rcGA
WSNs minimised energy depletion	[302]	
Model predictive control for sewer system	[317]	(non elitist) rcGA
Optimising hydrography ontology alignment	[137]	cross-rcGA variant
On-line training of neural controller	[138]	cDE
Optimised deployment of WSNs	[301]	
Optimal regression models in radio networks	[321]	
Industrial Cartesian robot optimal control	[305]	A memetic cDE algorithm
Space robot disturbance minimisation	[39, 139]	DEcDE, cDE-light
Path following problem	[142]	cDE-light
Fuzzy Neural optimal control for mobile robot	[141]	CDE-CLS
On-line control design of a boiler	[42]	cPSO
Base station locations in WSNs	[303]	
Optimal SNR in audio watermarking	[310]	
Optimisation-based audio watermarking	[311]	
Mobile sensor localisation	[145]	cAPSO
Topology control scheme in WSNs	[304]	cABC
Self-Standing Humanoid Robot	[161]	cHSA
Train neural and radial models	[159]	cTLBO
Optimal movement in humanoid robots	[38, 151]	cFA
Optimal bipedal robots walking	[150]	
Clustering of WSNs	[149]	cBA
Node Localisation in WSNs	[155]	cFPA
Gray image segmentation	[152]	cCSO
Optimise hydroelectric power generation	[153]	cPIO
Location of drone logistics hub	[156]	An improved cCS
Optimal allocation of distributed generation	[158]	pcEO and cEO variants
Vehicle routing problem with time window	[157]	cSCA
Public transit vehicles dispatch	[297]	McSCA
Node localisation in 3D actual terrain	[296]	pcCSA
Optimal bibliographic ontology matching	[136]	UcGA
Optimal thrust allocation (in Cybership III)	[160]	cHHO