Computational Optimization and Applications (2025) 91:1309–1338
https://doi.org/10.1007/s10589-025-00677-w

Adaptive dipole-like parameter calibration of complex black-box continuous processes

Avi Herbon[1] · **Mauro Gaggero**[2] · **Eugene Khmelnitsky**[3] · **Marcello Sanguineti**[4]

Received: 17 April 2024 / Accepted: 28 February 2025 / Published online: 29 March 2025

Abstract
Large-scale systems such as flexible manufacturing, chemical production facilities, and traffic networks aim to maximize measures related to profit, health and safety, throughput, and service level. Due to the complexity of such systems, the mechanism that connects the input parameters to performance outputs is often unavailable, and optimization methods based on convexity or even differentiability of the objective function may not be applicable. Since these systems are characterized by heavy costs per unit time, the system manager has to resort to black-box approaches for optimization, where a set of parameters is tuned in order to maximize an accumulated performance measure of the process. In this paper, a novel mechanism is proposed for real-time calibration of parameters in continuous search spaces. The developed algorithm seeks the global optimum by means of solution exploitation, and adapts dynamically according to environmental changes. The solution method builds a sequence of random pairs of trials, called "dipoles", which are used to adapt online the probability density function of the unknown parameters. The proposed method is characterized by the following advantages: (1) it does not depend on subjective coefficients setting; (2) solution exploitation starts from the first iteration; (3) the algorithm is effective also for systems with high dimensionality; (4) since sampling only involves two trials, exploitation is based on recent data rather than on data that extends far back in time. Several illustrative numerical examples are provided to show the applicability and efficiency of the proposed method.

Keywords Black-box optimization · Online parameter calibration · Random sampling · Global optimization

1 Introduction

Complex processes that operate in dynamic environments typically consume large amounts of capital and resources. Examples of such processes include tackling air and water pollution [53], monitoring traffic networks [37], operating call centers

Extended author information available on the last page of the article

[63], control of wildland fires [11], human-computer interaction [38], and controlling flexible manufacturing systems [44]. Such processes are often affected by random environmental perturbations, and may involve a large number of input parameters. Failure to control and operate such systems in an efficient way could have a negative impact on public health or firm or government reputation and profit. In addition, it could contribute to wasted resources. When decision makers are aware of the explicit operational mechanisms connecting input parameters with performance outputs, standard optimization methods can be efficiently applied to identify the best values of input parameters. However, due to the complexity of the processes described above, precise knowledge of the relationship that maps input parameters to performance outputs, or even the structural properties of this relationship, may not be available. In other words, such processes constitute black-box systems. Reaching an incontrovertible conclusion from each trial (as is the practice in many methods proposed in the literature) is likely to be much riskier than drawing a conclusion with only some level of likelihood. Real-life systems are often highly complex, and the full range of explanatory parameters is not always known. For example, the gross domestic product of a country or the mortality rate due to car accidents are the result of many factors, not all of which have been uncovered. In such applications, repeated "trials" would not ensure the same result. Furthermore, in many real-world systems, the number of parameters is large, making the task of handling the feasible parameter domain a significant challenge.

When dealing with complex, black-box processes, a set of input parameters has to be dynamically learnt and calibrated. Moreover, basic assumptions such as full consistency in obtaining the same outputs given repeated identical inputs in a discrete or continuous search space, or the convexity and differentiability of the objective, may not be guaranteed in real-life applications. Furthermore, it is often costly to implement multiple trials. Consequently, the problem of online tuning input parameters of complex processes has become an appealing research topic for both practitioners and theoreticians.

A large number of applications would stand to benefit from developments in the field of black-box parameter optimization. For example, in mobile robotics, wear and tear can change the system parameters by reducing the diameters of wheels or loosening belts. Such effects can introduce significant systematic errors into the robot odometry [54]. Golovin et al. [25] discuss a state-of-the-art system for black-box optimization developed within Google. Chen et al. [15] develop a Monte Carlo approach to facilitate portfolio selection using real data from the Shanghai-Shenzhen stock market. Abramson [2] analyzes a thermal insulation problem that involves optimizing a variable dictating the number of heat intercepts. In their review of black-box optimization applications, Alarie et al. [4] present 328 fields of application, as illustrated in Table 1 of their article. For example, the Gamma-ray monitoring device [16] utilizes a mesh adaptive direct search algorithm to monitor an apparatus that measures the attenuation of gamma rays emanating from the ground, and uses this information to infer the snow water equivalent in remote locations. Alarie et al. [3] illustrate parameter tuning for power system stabilizers while considering several generators simultaneously.

Gheribi et al. [23] present a case study that leads to an improvement in the mechanical properties of light metal alloys by maximizing the volume fraction of several specific rare-earth aluminum inter-metallics, under constraints on liquidus temperature, density and heat capacity.

In this paper, we propose an efficient method for parameter calibration, in which multiple parameters affecting the operation of a black-box process are tuned online in order to maximize a long-run performance index. We refer to the proposed approach as "dipole-like" method, as it is based on the idea of using samples consisting of only two trials, i.e., a dipole-like structure, as detailed in Sect. 3. We build upon the ideas of received rewards in Q-learning [17] and the online probabilistic algorithm presented by Arzi and Herbon [5], and modify them for a continuous search space associated with complex, large-scale processes. In several previous studies proposing black-box optimization methods, the goal has been to seek the global optimum, which, in many realistic situations, would lead to enormous costs over a very long horizon and might also never be reached. In contrast, we propose a new strategy that adapts dynamically to environmental changes and seeks to minimize the weighted, i.e., discounted over time, absolute gap between the transient optimum and the computed objective for a unit of time.

In general, non-deterministic searches are able to escape from local maxima by means of randomness. Online optimization methods have been implemented in various real-world applications. The use of computerized methods such as genetic algorithms [29, 52], simulated annealing [1], tabu-search [24], cross-entropy [55], and ant colony algorithms [18, 20], have enabled near-optimal solutions to be found for a variety of contexts. Yet, random perturbations are reasonably common when operating real-time complex systems under dynamic environments. The method proposed in this paper consists of online learning from a series of trials, each one connecting a given set of parameters with the objective.

The proposed mechanism is not specific to any particular application and has been developed for general parameter calibration within the realm of online optimization and adaptive control. We apply the algorithm and demonstrate its efficiency in continuous parameter spaces, although it could also be implemented in discrete settings with trivial adaptation. The main features of the proposed approach are as follows:

1. Based on the knowledge of the recent operational behavior of the system at specific points within the parameter space, the proposed algorithm penalizes or rewards only a small portion of the domain around these points. Thus, the algorithm does not significantly penalize or reward points that are distant from the points under consideration, and detailed knowledge of the previous system behavior is not required.
2. It is capable of dealing with high-dimensional problems, i.e., with a large number of parameters.
3. It is independent of any specific application, i.e., it has general applicability.
4. It does not require subjective coefficients to be set.

5. Exploitation starts from the first iteration. In particular, we address an objective that accumulates costs from $t=0$ and does not focus solely on convergence.

The dipole-like method can be used also for the "offline tuning" stage, commonly required to improve given algorithms or metaheuristics. Automatic parameter tuning is a rapidly growing field. Recent methods for this topic can be found in the survey paper by Huang et al. [31]. We compare the "dipole-like" approach with another well-known optimization method based on random sampling through low-discrepancy sequences (LDSs). The method, referred to as "LDS-iterative" in the following, can be applied for several well-known functions without assuming any knowledge of the relationship that maps input parameters to performance outputs. The results show that, for almost all instances, the suggested method outperforms the benchmark. While the LDS-iterative approach consumes less CPU time, both methods are applicable for real-world applications.

The remaining of this paper is organized as follows. Section 2 provides a literature review and examples of potential applications. In Sect. 3, the optimization problem is defined, and the proposed dipole-like algorithm is presented. Section 4 studies properties of the proposed method. Section 5 presents a detailed numerical comparison between the dipole-like approach and the LDS-iterative method. Lastly, Sect. 6 concludes the paper.

2 Literature review and application examples

Machine-learning techniques, heuristics based on statistical data analysis, optimization, and computational search methods are common tools when tuning the parameters of black-box systems online. In particular, we focus on direct search methods and model-based methods, which are, according to Xi et al. [62], key classes of derivative-free methods. An overview of derivative-free optimization methods and some applications may be found in the textbooks of Conn et al. [19], Audet and Hare [7], and in the surveys of Larson et al. [42] and Xi et al. [62].

Direct search methods sample the objective function at a set of points with a certain predefined geometric pattern. Due to their simplicity, such algorithms were widely used for many decades to solve black-box optimization problems. However, they either converge relatively slowly or cannot guarantee global convergence. The generalized pattern search (GPS), the Nelder-Mead algorithm, and the mesh adaptive direct search (MADS) are common direct search methods. Lewis et al. [43] pointed out that direct search methods for unconstrained optimization make use of the relative ranks of function values instead of the numerical function values, that is, they depend on the objective function only through the relative ranks of a countable set of function values. These methods do not fully exploit the available information in the objective function, and therefore they need to carry out a relatively large number of function evaluations, making them very slow. The MADS algorithm [6] generalizes the GPS by removing the restriction that the local exploration of the variable space is confined to a fixed, finite set of directions. Gramacy and Le Digabel [26] extended the surrogate management framework—a method that incorporates

the use of surrogate functions into a pattern search methodology—to non-smooth optimization under general constraints. Audet et al. [8] introduced a modification to the MADS algorithm so that it is able to handle granular variables, i.e., variables with a controlled number of decimals. Razavi et al. [51] reviewed, analyzed, and categorized research efforts on surrogate modeling and its applications, with an emphasis on research in the water resources field. They reported that more than 85% of the applications using this methodology involve less than 20 decision variables. For a large number of parameters, several trials may need to be carried out at nearly the same point (i.e., the trust region), without exploitation, in order to identify the "correct" search direction. Even after finding the preferred search direction, i.e., one that results in a successful trial, the probability of finding a local minimum rather than a global one is high. Thus, response surface surrogate modeling becomes less attractive or even infeasible when the number of explanatory variables is large. Moreover, these methods require subjective constants. For example, the Nelder-Mead algorithm requires the definition of five constants prior to being able utilizing the method, all of which may affect effectiveness.

Model-based methods construct a model that replaces the unknown mechanism of the operating system, and an objective function is minimized over the trust region to derive the next point in the subsequent iteration. The model used to interpolate the objective is established based on sample points. Some of the better-known model-based algorithms include derivative-free optimization [19], NEWUOA [49] and BOBYQA [50]. A survey of surrogate-assisted evolutionary optimization techniques can be found in Jin [35]. The model functions used in model-based methods are usually smooth and easy to evaluate, with typical choices being linear and quadratic. However, these functions are not sufficiently complex to reflect real-life operating systems. The interested reader is referred to the book of Koziel and Leifsson [39] to learn more about surrogate-based modeling.

We now describe some examples of large-scale systems in which online parameter tuning is required and where the method proposed in this paper might be advantageous. Many software programs in operations research depend on parameters that must be set before the program is executed. To this end, Baz et al. [9] present a method of tuning software parameters using ideas from software testing and machine learning. Schaul [56] discusses several applications for black-box optimization, one of which is the development of a method for speaker identification [14]. Other applications mentioned by Schaul [56] include optimization techniques for matching computed tomography scans to ultrasound images [61] and for forensic identification [32]. In chemistry, black-box optimization is used for chromatography [34] and to find stable crystalline structures [59].

Meta-heuristic procedures, such as evolutionary algorithms, simulated annealing, cross-entropy, tabu-search, and Bayesian techniques, combine a probabilistic search with optimization. These methods usually suffer from several drawbacks that reduce their suitability for efficient, online estimation of the parameters of complex, black-box processes. In particular, in order to operate effectively, simulated annealing requires several coefficients to be set, such as the initial temperature and the temperature stage duration, which makes it case-based [33]. Furthermore, its reliance on the initial state and its requirement that a large number of trials have already been

executed make it slow and difficult to be applied for large-scale systems. According to Elhaddad (2012), simulated annealing only uses one candidate solution, thus, the method may be unable to provide an overall view of the search space. Genetic search algorithms were conceived as an optimization method to solve problems with discrete variables [30]. Yet, they often suffer from slow convergence due to dependency on the initial population sample. The tabu-search algorithm improves local search performance by incorporating a memory bank containing the visited solutions into the search process. Specifically, this is used to prevent the algorithm from visiting solutions stored in the list at a subsequent iteration. In high-dimensional problems, however, the amount of required memory and the computational time may render the technique infeasible. The cross-entropy method translates a deterministic problem into an associated stochastic one. It adaptively estimates the probability of rare events, thus making it a global search method for combinatorial optimization problems. The algorithm is unlikely to become trapped in local optimal points. However, similarly to genetic algorithms, the cross-entropy method requires samples of points, which may be problematic for complex processes where each trial tends to be very expensive. Another approach for online parameter estimation is based on Q-learning, which is an efficient method for learning and optimizing systems when the dynamics can be described by a Markov decision process [60]. The main drawback of this approach, shared by some of the other available meta-heuristics, is the need to store previous rewards. Thus, Q-learning and meta-heuristics are regularly used when the system is characterized by a discrete state space. However, often the above-mentioned approaches cannot be efficiently implemented in real-world applications, since no explicit model for the performance measure exists. Therefore, it may be costly and time-consuming to compute a performance measure for a given set of parameters. An efficient way of addressing the aforementioned problems is to use probabilistic approaches based on random search algorithms with adapted steps. Arzi and Herbon [5] adopted this perspective and modeled the scenario where a decision maker controls a multi-cell flexible manufacturing system consisting of machines that operate in a highly variable, produce-to-order environment subject to failures.

The approach proposed in this paper belongs to a class of algorithms referred to as estimation-of-distribution algorithms (EDAs), which store a probabilistic model over the solution space. EDAs have been used very successfully in real-world applications (see, e.g., [27, 41, 45, 46]). The interested reader is also referred to Chapter 4 in Bengoetxea [10], which introduces several algorithms and codes for carrying out these methods. According to Krejca and Witt [40], optimization in a noisy setting (or, more generally, optimization under uncertainty) is a scenario in which EDAs may outperform classical evolutionary algorithms. In contrast to the common practice of storing a probabilistic model over the entire multidimensional solution space, we suggest a heuristic approach that is free of coefficients and stores the updated probability distribution for each variable separately, thus enabling higher-level control of the parameters. Instead of using sample sets (often referred to as *populations*) to revise the probability distribution, we suggest using samples consisting of only two trials (i.e., a dipole-like structure). In the next section, we will refer to this idea as "dipole-like" method.

3 Problem formulation and solution methodology

We consider a system that depends on N different parameters. After setting their initial values (initial values can be set randomly or by exploiting possible a-priori estimates that may be available), the system starts working and the chosen performance measure (e.g., profit, throughput, service level, etc.) is recorded during its operation, usually after a predetermined delay. Having obtained the performance measure of the system with the current parameter values, we want to select the optimal values for the next working cycle. Thus, we can consider the parameters as time dependent. The search for the optimal values of the parameters is limited to a compact, continuous set in the N-dimensional space. The goal is to develop a policy that minimizes the transient-time cost originating from the gap between the maximal value of the function modeling the performance measure under study and the current value of the function. In the remainder of this section, we define the problem and outline the proposed solution method.

3.1 Problem formulation

Let $F(p_1(t), p_2(t), ..., p_N(t))$ be a scalar function of time t, which depends on N operational parameters $p_1, p_2, ..., p_N$ assumed to be time-variant, as discussed before. We also assume that all parameters are independent. The set of parameters $p_1, p_2, ..., p_N$ is considered independent if the decision maker can change each parameter while keeping all the others fixed. For example, consider the objective of minimizing the accumulated cost of car accidents. The strategy required to control this real-time transportation system might involve modifying the number of cell phones in cars and modifying the number of cars on the road. Obviously, decreasing the number of cars also decreases the number of cell phones, meaning that the two variables are interdependent and one of them is not needed. Formally speaking, one cannot freely determine any chosen value of one of the variables and keep the other fixed. In the case where the set of parameters is dependent, it is always possible to decrease the number of parameters to a level where those influencing the performance are independent, that is, to reduce the degree of freedom to the minimal possible value.

Our model is grounded on the assumption that some understanding of the parameters (e.g., the search domain boundaries of each parameter) and the relationship between them is required; however, we do not require knowledge of the specific function that relates the inputs to the output. The function F is assumed to map the input parameters to the output performance of a certain complex process. The time runs over discrete steps (for instance, days), i.e., $t = 1, 2, ..., T$, and accordingly, the parameter values are tuned at each discrete step t. For this reason, as previously pointed out, they can be considered time-variant. The parameters are independently defined within bounded intervals, that is, $p_n(t) \in [a_n, b_n]$, $n = 1, 2, ..., N$, for all t. We look for a strategy that maximizes F in the long run, i.e., the solution of an optimization problem that consists in minimizing the difference between the values of F corresponding to the current combination of

parameters and the maximal value of F. More formally, the following optimization problem is solved:

$$\begin{aligned} &\min_{p_n(t),\ n=1,...,N} \frac{\sum_{t=1}^{T} e^{-rt}\left|F^*(t)-F(p_1(t),...,p_N(t))\right|}{\sum_{t=1}^{T} e^{-rt}} \\ &s.t. \\ &a_n \le p_n(t) \le b_n \quad n=1,2,...,N,\ t=1,2,...,T \end{aligned}, \tag{1.1}$$

where $F^*(t) = \max_{p_n(t),n=1,...,N} F(p_1(t),p_2(t),...,p_N(t))$ for $t=1,2,...,T$, T is the number of steps over which the system will operate (chosen a priori), and r is a discounting coefficient. By definition, $F^*(t) \ge F(p_1(t),...,p_N(t))$. Thus, problem (1.1) is equivalent to:

$$\begin{aligned} &\max_{p_n(t),\ n=1,...,N} \sum_{t=1}^{T} e^{-rt} F(p_1(t),p_2(t),...,p_N(t)) \\ &s.t. \\ &a_n \le p_n(t) \le b_n \quad n=1,2,...,N,\ t=1,2,...,T \end{aligned}. \tag{1.2}$$

Below we report the main assumptions of the problem under investigation.

A1. The search domain (i.e., an N-dimensional box) is finite and known.

A2. All parameters to be calibrated are continuous within their domain.

A3. Each parameter to be calibrated can be changed while all other parameters remain fixed (i.e., the parameters are independent).

A4. The scalar function $F(p_1,p_2,...,p_N)$ is continuous with respect to each parameter.

3.2 Outline of the solution method

Let $v := (p_1,p_2,...,p_N)$ denote the set of parameters and let $v_t := (p_1(t),p_2(t),...,p_N(t))$ denote the set of parameter values at time step t, referred to as a "sample". Since it is impossible to store all historical data, especially for a large number of parameters, the proposed method builds a sequence of random samples, where each one is drawn from a certain probability distribution function $\varphi_t(v)$. The sample generation function $\varphi_t(v)$ is updated at each time step t after obtaining the performance measures for two successive samples v_{t-1} and v_t. To simplify the notation, herein we drop the dependence on the index t, thus denoting by v^+ and v^- the sample with the higher and the lower performance measure, respectively, among the two randomly-extracted sets of parameters, that is, $F(v^+) \ge F(v^-)$. As explained below, the generator $\varphi(v)$ is updated in such a way that it significantly increases (decreases) within a neighborhood of $v^+(v^-)$, whereas in the rest of the domain, it is barely affected.

The concept of updating the probability distribution based on two samples is inspired by the structure of the electric field of a dipole, i.e., two charges equal in magnitude but opposite in sign. The absolute magnitude of the electric field falls away much faster in the vicinity of the poles than in other parts of the field. Figure 1 illustrates a one-dimensional dipole-like field with the charges located at the points $p = 1$ and p =-1.

The most interesting features of a dipole-like update are as follows:

1. It implicitly learns the function $F(v)$ by assigning higher (lower) probabilities, in terms of the generator $\varphi(v)$, to the areas where F is higher (lower).
2. Learning of the objective function $F(v)$ makes use of both samples. This represents a contrast with optimization methods that neglect the information associated with less successful samples.
3. The distribution function is updated over the entire domain, and not only at the two sample points.
4. Due to the dipole-like feature, the probability distribution in areas far from the two samples is changed only insignificantly, in order to reflect the lack of information required for such an update.

Since the procedure is initialized when no information about the function F is available, the initial $\varphi(v)$ is set to be a uniform distribution.

3.3 Discretization of the search space and learning of the objective function

In order to update the distribution function $\varphi(v)$, we discretize the interval $[a_n, b_n]$ into D_n sub-intervals of equal length. The vector of points corresponding to the y_n-th subinterval, denoted by p_{yn}, is given by

$$p_{yn} = a_n + y_n \frac{b_n - a_n}{D_n}, \quad y_n = 0, 1, ..., D_n, \quad n = 1, 2, ..., N. \tag{2}$$

Thus, the search in the parameter space $S = \prod_{n=1}^{N} [a_n, b_n]$ is carried out over a discrete space S^d of N-dimensional vector $v = (y_1, ..., y_N)$, $y_n = 0, 1, ..., D_n$. Note that the replacement of the absolute values of the parameters p_{yn} with the indexes y_n ensures that the method can always combine different types of parameters, even though each one has its own units and range.

Another key feature of the approach is that the learning of the objective function $F(v)$ through the generator $\varphi(v)$ is split into N one-dimensional procedures. Thus, instead of storing and updating the joint distribution function $\varphi(v)$, which would

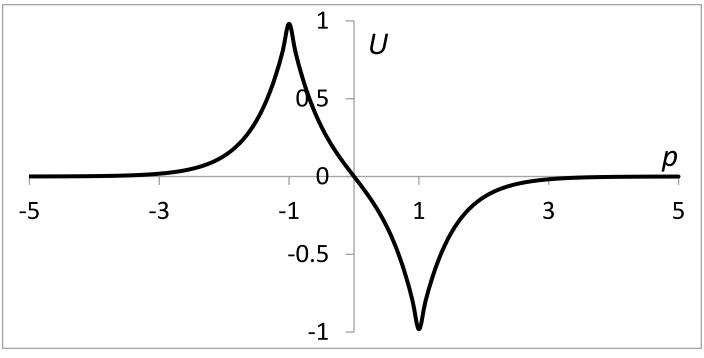

Fig. 1 The dipole-like potential function U, in a one-dimensional parameter space

be impossible for systems depending on a large number of parameters, we propose updating each of its components $\varphi_n(y)$ independently. Such a decomposition reduces the complexity of the method to be linear in N, thus extending its applicability to large-scale optimization problems. The alternative of storing the entire multi-dimensional distribution would be infeasible even for moderate problem sizes. The proposed decomposition technique supports the axiomatic approach under which the decision maker is familiar, a priori, with the potential interdependency of the parameters, which reduce the number of parameters such that, eventually, can manage the operating system only with independent control variables.

To evaluate the progress of the method and the rate of convergence to the transient optimum, we make use of the entropy measure E_n of the one-dimensional distribution functions $\varphi_n(y)$, defined as

$$E_n = -\frac{\sum_{y_n=0}^{D_n} \phi_n(y) \ln(\phi_n(y))}{\ln\left(D_n + 1\right)}, \quad n = 1, 2, ..., N. \tag{3}$$

The numerator is the well-known Shannon entropy formula [57]. As previously pointed out, the method starts with $\varphi_n(y)$ being a uniform distribution, i.e., we have $E_n = 1$ for all $n = 1, 2, ..., N$. When approaching the optimal solution, the uncertainty on $\varphi_n(y)$ diminishes and E_n tends to zero. Note that the use of alternative measures to the one stated above (for instance, a measure based on F itself) was not considered because F is a black-box function from the perspective of the decision maker.

3.4 Dipole-like update of $\varphi_n(y)$

The probability functions $\varphi_n(y)$ are updated after obtaining two successive samples, $v^+ = (y_1^+, y_2^+, ..., y_N^+)$ and $v^- = (y_1^-, y_2^-, ..., y_N^-)$, such that $F(v^+) \geq F(v^-)$. To this end, we follow the analogy with the electric dipole and define the magnitude of the dipole charge as the difference in the performance measures, $\Delta F = F(v^+) - F(v^-)$, and the dipole moment along the axis n (i.e., the product of the magnitude of the dipole charge and the distance between the charges projected onto axis n), as follows:

$$\Delta F_n = \frac{\Delta F \left|\Delta y_n\right|}{\sum_{n=1}^{N} \left|\Delta y_n\right|}, \; n = 1, 2, ..., N, \tag{4}$$

where $\Delta y_n = y_n^+ - y_n^-$, $y_n^+ \in \{0, 1, ..., D_n\}$, $y_n^- \in \{0, 1, ..., D_n\}$, and consequently $\Delta F = \sum_{n=1}^{N} \Delta F_n$. Without loss of generality, we make use of exponential functions (rather than the power functions used in physics) to define the field generated by the dipole. Hence, given the points y_n^+ and y_n^-, the dipole-like potential at any point y_n in S^d is given by

$$U(y_n) = \Delta F_n U^0(y_n), \quad y_n = 0, 1, ..., D_n, \quad n = 1, 2, ..., N, \tag{5}$$

where

$$U^0(y_n) = e^{-\gamma_n \frac{\left|y_n^+ - y_n\right|}{D_n+1}} - e^{-\gamma_n \frac{|y_n^- - y_n|}{D_n+1}}, \quad y_n = 0, 1, \ldots D_n, \qquad n = 1, 2, ..., N. \tag{5.1}$$

The coefficient $\gamma > 0$ determines the extent to which the probability of each feasible point is affected by the dipole. The smaller the value of γ_n, the larger the spread of the field across the feasible domain. On the contrary, the higher the value of γ_n, the more the field is concentrated in the neighborhood of y_n^+ and y_n^-. This factor is similar to the variable cooling factor in simulated annealing schedules, which are inspired by the metallurgical process of heating up a solid and then slowly cooling it until it crystallizes [47]. In line with Peprah et al. [47], we believe that the cooling factor should be a variable $\gamma_n(t) > 0$ and not a constant as proposed by many other authors, such as in the case of simulated annealing (see, for example, [28]). In particular, we choose to represent it by an intrinsic factor—named "entropy"—scaled between 0 and 1, as this is suited to our application, that is,

$$\gamma_n(t) = E_n(t), \qquad n = 1, 2, ..., N, \quad \forall t. \tag{6}$$

The above description of the role of coefficients $\gamma_n > 0$ is analogous to the concept of entropy in physics. The atoms of the heated solid material gain high energy (i.e., high entropy) at very high temperatures, giving them a great deal of freedom in their ability to restructure themselves. As the temperature is reduced, the energy of these atoms decreases until a state of minimum energy is achieved.

The overall potential (5) is given by the superposition of the effect of the positive charge (the point with a higher objective) and the negative charge (the point with a lower objective), similar to the electrical potential due to a dipole in physics. Since the potential $U(y_n)$ in (5) can take on positive as well as negative values (see Fig. 1) within the range $\left[-\Delta F_n,\ \Delta F_n\right]$, we transform it into a positive potential $W(y_n)$ as follows:

$$W(y_n) = U(y_n) + \Delta F_n, \quad y_n = 0, 1, ..., D_n, \quad n = 1, 2, ..., N. \tag{7}$$

Then, the dipole-like method updates the probability $\varphi_n(y)$ and obtains the distribution function $\varphi_n^{new}(y_n)$ at the next iteration (i.e., the next time step), as follows:

$$\phi_n^{new}(y_n) = \begin{cases} \frac{\phi_n(y_n) \cdot W(y_n)}{\sum_{y_n=0}^{D_n} \phi_n(y_n) \cdot W(y_n)}, & if\ \Delta F_n > 0 \\ \phi_n^{new}(y_n) = \phi_n(y_n), & if \Delta F_n = 0 \end{cases} \quad n = 1, 2, ..., N,\ y_n = 0, 1, ..., D_n \tag{8}$$

where the sum of $\varphi_n^{new}(y_n)$ is normalized to 1. The positive potential $W(y_n)$ increases the probability of the points in the neighborhood of v^+ and decreases the probability of the points in the neighborhood of v^-. The magnitude of the increase (or decrease) in probability of a certain point strongly depends on the dipole moment, i.e., it depends on the charge ΔF_n and on the distance between the point and the charges.

3.5 Compression of the search space for a continuous parameter domain

In Sect. 3.3, we described the discretization of the search space S where the domain of the n-th parameter is divided into D_n equal-length sub-intervals,

$n = 1, 2, ..., N$. In order to better approximate the continuous search space S by means of a discrete space S^d, the value of D_n can be dynamically adjusted while running the algorithm. In particular, after each iteration, we propose increasing D_n by one unit if the entropy measure E_n of the distribution function $\varphi_n(y)$ has increased (corresponding to a deterioration). An increase in entropy along the n-th axis can also result from insufficient accuracy of the representation of the continuous objective function F by the discrete grid of D_n points. Adjusting the value of D_n (using the method described below) achieves a better balance between the complexity of the method and its accuracy. In the following, we refer to such a procedure as "compression". We point out that the use of the compression step is optional.

When D_n is increased by one unit and a new grid of points y_n is created according to (2), the probability distribution $\varphi_n(y)$ has to be spread over the new grid. The distribution over the finer grid is determined as a linear combination of the distribution values in the old grid:

$$\phi_n^{new\ grid}(y_n) = \sum_{z=0}^{D_n-1} \phi_n^{old\ grid}(z) e^{-\alpha_n |z-y_n|}, \quad n = 1, 2, ..., N, \quad y_n = 0, 1, ..., D_n \tag{9}$$

where z encompasses the points of the old grid, i.e.,$z \in \{0, 1, ..., D_n - 1\}$, and α_n is such that the new distribution is normalized to 1:

$$\sum_{y_n=0}^{D_n} \phi_n^{new\ grid}(y_n) = \sum_{y_n=0}^{D_n} \sum_{z=0}^{D_n-1} \phi_n^{old\ grid}(z) e^{-\alpha_n |z-y_n|} = 1, \quad n = 1, 2, ..., N. \tag{10}$$

Proposition 2 in Sect. 4 proves the existence and uniqueness of the parameter $\alpha_n > 0$.

4 Mathematical analysis

The strategy is implemented by defining an empirical probability distribution function of the unknown parameters over the feasible search region based on historical performance records, and dynamically updating it using the last two trials. The rationale for this approach is that exploitation based on extremely large samples, as is commonly carried out in other black-box optimization methods, might be unreliable. This is due to possible environmental changes that decrease the reliability of consistency of data acquired from trials that span a long-time interval. From the methodological and technical viewpoints, our method is analogous to the dipole phenomenon. Specifically, we make use of the form of the electrical field created by a dipole: the values of the objective function obtained for the last two trials are considered as a dipole, and the electrical field created by the dipole is used to update the probability density function of the parameters. The use of the dipole configuration makes it possible,

to a large extent, to confine the update to the region of the search space that is nearest to the poles, i.e., the trust region.

4.1 The algorithm

The proposed algorithm for parameter calibration iteratively draws a dipole from the current distribution function $\varphi_n(y)$, evaluates the performance measure, and updates $\varphi_n(y)$, $n = 1, 2, ..., N$. The steps of the algorithm are as follows, where we have explicitly included the iteration t (i.e., time step) as a superscript in all quantities.

Step 0 (initialization). Set the initial number of sub-intervals D_n and generate a uniform probability distribution $\varphi_n^{(0)}(y) = \frac{1}{D_n+1}$ for $y_n = 0, 1, ..., D_n$, $n = 1, 2, ..., N$. Let $t = 0$.

Step 1 (compute entropy). Compute the entropy $E_n^{(t)}$ of $\varphi_n^{(t)}(y)$ according to (3).

Step 2 (obtain two samples). Randomly extract two samples v^+ and v^- from the probability distribution function $\varphi_n^{(t)}(y)$ such that $F(v^+) \geq F(v^-)$, for all $n = 1, 2, ..., N$. Denote by v_t^{best} the point that has achieved the best performance up to time t.

Step 3 (update the distribution). Compute the dipole moment of the samples and its projections onto the N axes, according to (4). Then, compute the positive potentials of the points according to (5) and (6), respectively. Lastly, compute the new distribution $\varphi_n^{(t+1)}(y)$ according to (8) for all $n = 1, 2, ..., N$. Repeat Step 3 for each of the following:

(a) Replace v^+ with the point v_t^{best} and replace v^- with the point v^+.

(b) Replace v^+ with the point v_t^{best}.

Step 4 (compression of the search space). Compute the updated entropy measure $E_n^{(t+1)}$ for all $n = 1, 2, ..., N$. If $E_n^{(t+1)} > E_n^{(t)}$, increase D_n by one unit and spread the distribution over the new grid according to (9) and (10).

Step 5 (incrementation). Set $t \leftarrow t + 1$ and go back to Step 1.

Since real-time black-box systems evolve dynamically, we do not set any stopping rule for the above algorithm. In particular, as long as the system under consideration continues to operate (i.e., $t \leq T$), the algorithm seeks a better set of parameters.

4.2 Theoretical results

In this section, we prove two properties that justify Steps 3 and 4 of the method presented in Sect. 4.1. Proposition 1 shows that the updated probability density function increases in the neighborhood of y_n^+ and decreases in the neighborhood of y_n^-. As a result, the correlation between the distribution $\varphi(y)$ and the objective function F increases with the number of iterations of the method.

Proposition 1 *Denote by r_n the mean potential defined w.r.t. the current probability distribution, that is,*

$$r_n = \sum_{y_n=0}^{D_n} \phi_n^{(t)}(y_n) U^0(y_n), n = 1,2, \dots, N. \tag{11}$$

For a generic iteration t and for all $n = 1, 2, ..., N$, the inequality

$$\phi_n^{(t+1)}(y_n) \geq \phi_n^{(t)}(y_n)$$

holds for all y_n satisfying

$$\left|y_n - y_n^+\right| \leq -\frac{D_n + 1}{\gamma_n} ln\left(\sqrt{\frac{r_n^2}{4} + e^{\frac{-\gamma_n|\Delta y_n|}{D_n+1}}} + \frac{r_n}{2}\right);$$

and the inequality

$$\phi_n^{(t+1)}(y_n) \leq \phi_n^{(t)}(y_n)$$

holds for all y_n satisfying

$$\left|y_n - y_n^-\right| \leq -\frac{D_n + 1}{\gamma_n} ln\left(\sqrt{\frac{r_n^2}{4} + e^{\frac{-\gamma_n|\Delta y_n|}{D_n+1}}} - \frac{r_n}{2}\right).$$

Proof Denote by h_n the mean weight defined w.r.t. the current probability distribution, that is,

$$h_n = \sum_{y_n=0}^{D_n} \phi_n^{(t)}(y_n) W(y_n), n = 1,2, \dots, N, \tag{12}$$

Then, from (8) it follows that
$\phi_n^{(t+1)}(y_n) \geq \phi_n^{(t)}(y_n)$ iff $W(y_n) \geq h_n$, and $\phi_n^{(t+1)}(y_n) \leq \phi_n^{(t)}(y_n)$ iff $W(y_n) \leq h_n$.
From (7) we know that $W(y_n) = \Delta F_n\left(1 + U^0(y_n)\right)$ and, as a result, $h_n = \Delta F_n\left(1 + r_n\right)$.
Therefore, the inequality $W(y_n) \geq h_n$ holds iff

$$U^0\left(y_n\right) \geq r_n. \tag{13}$$

By substituting (5.1) into (13) we obtain

$$e^{\frac{-\gamma_n\left|y_n^+ - y_n\right|}{D_n+1}} - e^{\frac{-\gamma_n\left|y_n^- - y_n\right|}{D_n+1}} \geq r_n. \tag{13.1}$$

Consider first a y_n located between y_n^+ and y_n^- so that

$$\left|y_n^- - y_n\right| = \left|\Delta y_n\right| - \left|y_n^+ - y_n\right|.$$

Then, (13.1) writes

$$e^{\frac{-\gamma_n\left|y_n^+-y_n\right|}{D_n+1}}-e^{\frac{-\gamma_n\left(|\Delta y_n|-\left|y_n^+-y_n\right|\right)}{D_n+1}}\geq r_n. \tag{13.2}$$

We denote $e^{\frac{-\gamma_n\left|y_n^+-y_n\right|}{D_n+1}}$ by s_n and re-write (13.2) as the quadratic inequality,

$$\left(s_n\right)^2-e^{\frac{-\gamma_n|\Delta y_n|}{D_n+1}}\geq r_n s_n, \tag{13.3}$$

whose solution is

$$\left|y_n-y_n^+\right|\leq-\frac{D_n+1}{\gamma_n}ln\left(\sqrt{\frac{r_n^2}{4}+e^{\frac{-\gamma_n|\Delta y_n|}{D_n+1}}}+\frac{r_n}{2}\right). \tag{13.4}$$

Consider y_n located on the other side of y_n^+ so that

$$\left|y_n^--y_n\right|=|\Delta y_n|+\left|y_n^+-y_n\right|.$$

Then, the solution of (13.1), which takes the form of the linear inequality

$$s_n\left(1-e^{\frac{-\gamma_n|\Delta y_n|}{D_n+1}}\right)\geq r_n, \tag{13.5}$$

is

$$\left|y_n-y_n^+\right|\leq-\frac{D_n+1}{\gamma_n}ln\left(\frac{r_n}{1-e^{\frac{-\gamma_n|\Delta y_n|}{D_n+1}}}\right). \tag{13.6}$$

Since both (13.4) and (13.6) should hold in the neighborhood of y_n^+ and since the right-hand side of (13.4) is greater than the right-hand side in (13.6), we obtain

$$\left|y_n-y_n^+\right|\leq-\frac{D_n+1}{\gamma_n}ln\left(\sqrt{\frac{r_n^2}{4}+e^{\frac{-\gamma_n|\Delta y_n|}{D_n+1}}}+\frac{r_n}{2}\right),$$

which proves the first part of the proposition related to the neighborhood of y_n^+. By using similar arguments, one can prove also the second part of the proposition related to the neighborhood of y_n^-. □

The feasibility of Step 4 of the algorithm is proved in the following proposition.

Proposition 2 *Recalling that $\phi_n^{old\ grid}$ represents the probability distribution $\varphi_n(y)$ of the old grid, then, for all $n=1,2,...,N$, there exists a unique $\alpha_n>0$ such that*

$$\sum_{y_n=0}^{D_n}\sum_{z=0}^{D_n-1}\varphi_n^{old\ grid}(z)e^{-\alpha_n|z-y_n|}=1,\quad n=1,2,...,N. \tag{14}$$

Proof Denote by $G_n(\alpha_n)$ the left-hand side of (14) as a function of α_n. By differentiating α_n w.r.t. α_n, we obtain

$\frac{d}{d\alpha_n} G_n(\alpha_n) = -\sum_{y_n=0}^{D_n} \sum_{z=0}^{D_n-1} \varphi_n^{old\ grid}(z)|z - y_n| e^{-\alpha_n|z-y_n|} < 0,$

that is, $G_n(\alpha_n)$ monotonically decreases as a function of α_n. From the monotonicity of $G_n(\alpha_n)$ and the facts that $G_n(0) = D_n$ and $G_n(\infty) = 0$, it follows that the equation $G_n(\alpha_n) = 1$ must have a unique solution. Efficient numerical procedures (e.g., a one-dimensional golden section search [36]) can be implemented to locate the root of the equation $G_n(\alpha_n) = 1$. □

5 Simulation results

5.1 Background and benchmark method

This section reports the results of numerical simulations performed to show the effectiveness of the proposed dipole-like method for parameter calibration. We present results from eight different cases, summarized in Tables 1, 2, 3, 4, 5, 6, 7 and 8, denoted by "Case A", "Case B", …, Case "H", respectively. The examples were chosen in such a way as to investigate possible issues related to continuity of the search domain, convexity, and problem size. The availability of an optimal solution allows one to evaluate the effectiveness of the proposed approach in terms of accuracy. Clearly, when applying the proposed approach, we assume no knowledge of the true optimal solution or of the properties of the problem, such as convexity or separability of the variables. Since the initial number D_n of sub-intervals was chosen to be relatively high (we fixed $D_n = 10{,}000$), the algorithm in Sect. 4.1 was run without applying the compression step (Step 4) in Case A up to Case G. The application of the compression step with such high levels of D_n would add significant CPU time with only insignificant performance improvements. In fact, according to (9), ND_n^2 computations are needed as well as solving Eq. (10) needs even more than ND_n^2 computations when compression step is completed. In the case decision makers are limited with time budget, it is likely they choose a smaller initial number D_n of sub-intervals (to enable its feasibility); the compression step will be applied to accelerate performance improvement with additional CPU time. To have a complete assessment of the effectiveness of the proposed approach, we evaluated the role of compression in Case H.

The results provided by the dipole-like approach were compared with another optimization method. Among the various alternatives, we selected a technique based on random search since it shares with the proposed dipole approach: (a) the assumption that the mapping F is a black-box and (b) the strategy of finding an approximate solution through a sequence of randomly-extracted samples. Furthermore, it requires a very limited number of constants in comparison with other methods described in Sects. 1 and 2. The basic scheme selected as a benchmark is based on random sampling of the parameter space by using low-discrepancy sequences (LDSs). These are a type of deterministic sequence commonly employed in the

fields of number-theoretic methods, statistics, and quasi-Monte Carlo integration. The main motivation for using low-discrepancy sampling is that the resulting sets of points provide efficient, uniform coverage of the state space even under high-dimensional settings. For this reason, The LDS approach has been widely used also in the contexts of approximate minimization and dynamic programming (see, e.g., [12, 13, 12–13]).

Low-discrepancy sampling consists of sampling the N-dimensional space of the feasible parameters using LDSs made up of G' points. The function F is then evaluated at all the sample points, and the optimal value is chosen to be the sample that maximizes (or minimizes)F. The LDS method usually selects large values of G' (corresponding to a large exploration of the domain space prior to reaching a conclusion about the preferred areas). However, there are two main drawbacks associated with this method. The first is that choosing a large value of G' means that the operating system runs for a very long period of time on the basis of randomly selected parameters rather than preferred parameters. The second drawback arises from the fact that the operating system is assumed static, and therefore the objective $F(p_1, ..., p_N)$ behaves identically throughout this time. These drawbacks could represent a significant obstacle when applying the method to complex systems operating in real time.

Since in black-box processes it may not be possible for the operator to wait until all G' trials have been conducted, we considered an iterative version of the LDS approach, referred to as "LDS iterative", in which a smaller sample, made up of $G \leq G'$ points, is selected (specifically, we used $G = 100$). In order to learn from the current sample (i.e., the one obtained at the previous iteration), the subsequent sample (made up of G new trials) is conducted over a smaller search domain. The new search domain is defined as the maximal hyperrectangular sub-space volume (multi-dimensional), such that its center point coincides with the best point recorded at the previous iteration.

In the numerical results reported in the following, we conducted a total of 10,000 trials for each method (i.e., where a "trial" consists of operating the system and computing F). For the suggested dipole-like approach, we performed 5000 iterations ($K = 5000$), where each one was assigned a sample of two random points. For the LDS iterative technique, 100 iterations were implemented, each one characterized by a sample of G random points. We set the discounting coefficient r equal to 0.001. All trials were performed in MATLAB on an Intel Xeon 2.5GHz CPU with 16 GB of RAM.

5.2 Results

In this section, we present the results obtained by the dipole-like approach and the LDS-iterative one in all the considered case studies, from Case A up to Case H.

Case A: Non-linear and non-concave function (additive)

In this case, we have to solve $\min \left\{ F(p_1, ..., p_N) = p_1^3 + \sum_{i=2}^{N} \left(p_i - i\right)^2 \right\}$ with $p_i \in [-100, 100]$. The optimal solution of this problem is given by $p_1^* = -100$, $p_i^* = i$, $i = 2,\ldots,N$, and $F^* = -1{,}000{,}000$.

Table 1 reports the results obtained for different values of N. We observe that, while the gap to optimality increases with problem size for both methods, the dipole-like approach maintains significantly better accuracies than the benchmark.

Case B: Non-linear and non-concave function (multiplicative)

In this case, we have to solve $\max\left\{F(p_1,...,p_N)=\prod_{i=1}^{N}p_i\right\}$ with $p_i\in[-1,1]$, $i=1,2,\ldots,N$. The optimal solution of this problem is $p_i^*=1$, $i=1,\ldots,N$ and $F^*=1$.

Table 2 reports the results for different values of N. Similarly to the former case, we observe a relatively small CPU time (in sec.) for both methods. Case B exemplifies a particular case where the iterative LDS method achieves superior performance when $N\geq 30$. We assign this outcome to the unique circumstances of the problem

Table 1 Summary of the results for Case A

	N=2	*N=5*	*N=10*	*N=20*	*N=30*	*N=50*	*N=100*
LDS iterative							
Objective F (in units of 1000)	− 986.38	− 971.91	− 947.84	− 893.65	− 843.69	− 741.60	− 600.72
CPU time (in sec.) for each iteration	0.0009	0.0018	0.0013	0.0022	0.0032	0.0052	0.0106
Dipole-like							
Objective F (in units of 1000)	− 998.27	− 998.06	− 998.30	− 998.11	− 997.99	− 997.43	− 991.25
CPU time (in sec.) for each iteration	0.0093	0.0232	0.0457	0.0910	0.1361	0.2273	0.4474

investigated in Case B where the likelihood of finding a high objective value in any sub-domain of the entire domain is nearly zero. This likelihood strongly decreases with the number of dimensions.

Case C: The Dixon function

In this case, we have to solve $\min\left\{F(p_1,...,p_N)=(p_1-1)^2+\sum_{i=2}^{N}i(2p_i^2-p_{i-1})^2\right\}$ with $p_i\in[-10,10]$, $i=1,2,\ldots,N$. The optimal solution of this problem is $p_i^*=2^{-\frac{2^i-2}{2^i}}$, $i=1,2,\ldots,N$ and$F^*=0$.. This well-known test function (see Fig. 2) is taken from the Virtual Library of Simulation Experiments [58]. Considering that the objective in this case is measured in units of 1,000,000 (see Table 3) for the LDS

Table 2 Summary of the results for Case B

	N=2	*N=5*	*N=10*	*N=20*	*N=30*	*N=50*	*N=100*
LDS iterative							
Objective F	0.9883	0.9756	0.9580	0.9321	0.9265	0.8978	0.8355
CPU time (in sec.) for each iteration	0.0003	0.0006	0.0012	0.0021	0.0031	0.0052	0.0101
Dipole-like							
Objective F	0.9987	0.9957	0.9891	0.9703	0.8388	0	0
CPU time (in sec.) for each iteration	0.0086	0.0233	0.0458	0.0914	0.1381	0.2326	0.4643

iterative method, the superiority of the dipole-like method as compared to the LDS-iterative one is notable for all problem sizes.

Case D: The Rosenbrock function

In this case, we have to solve $\min\left\{F(p_1,\ldots,p_N)=\sum_{i=1}^{N-1}100\left(p_{i+1}-p_i^2\right)^2+\left(p_i-1\right)^2\right\}$ where $p_i\in[-5,10]$, $i=1,2,\ldots,N$. The optimal solution of this problem is given by $p_i^*=1$, $i=1,2,\ldots,N$ and $F^*=0$**.** This function is unimodal and the global minimum lies in a narrow, parabolic valley (see Fig. 3). However, even though this valley is easy to find, convergence to the minimum is difficult [48]. Table 4 reports the results for different values of N. Similar conclusions as for Case C can be drawn from the obtained results.

Case E: The Styblinski–Tang function

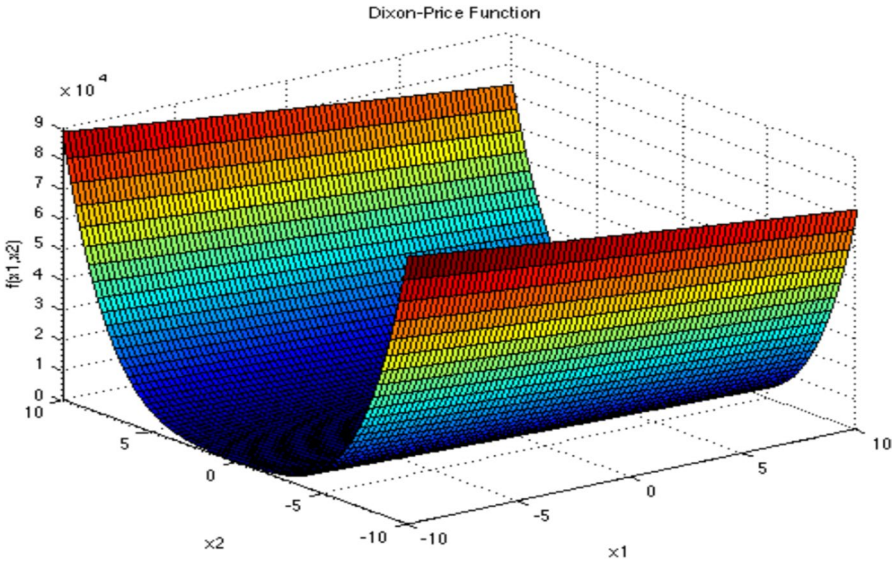

Fig. 2 The Dixon function

Table 3 Summary of the results for Case C

	$N=2$	$N=5$	$N=10$	$N=20$	$N=30$	$N=50$	$N=100$
LDS iterative							
Objective F (in units of 1,000,000)	0.0132	0.0859	0.3448	2.3051	6.8283	27.1778	133.0885
CPU time (in sec.) for each iteration	0.0007	0.0017	0.0014	0.0030	0.0033	0.0058	0.0104
Dipole-like							
Objective F	0.0248	0.4022	2.0890	7.9878	37.8144	202.4710	8799.3650
CPU time (in sec.) for each iteration	0.0094	0.0023	0.0457	0.0911	0.1358	0.2273	0.4460

In this case, we have to solve $\min\left\{F(p_1,...,p_N)=\frac{1}{2}\sum_{i=1}^{N}\left(p_i^4-16p_i^2+5p_i\right)\right\}$ where $p_i \in [-5,5]$, $i=1,2,\ldots,N$. The optimal solution of this problem is given by $p_i^*=-2.903534$, $i=1,2,\ldots,N$, and $F^*=-39.16599$N. This well-known test function (see Fig. 4) is taken from the Virtual Library of Simulation Experiments [58]. A significant gap between the two methods in reaching the optimal objective is showcased for Case E in Table 5. The problem size seems to negatively affect the LDS iterative methods much faster than it affects the dipole-like method.

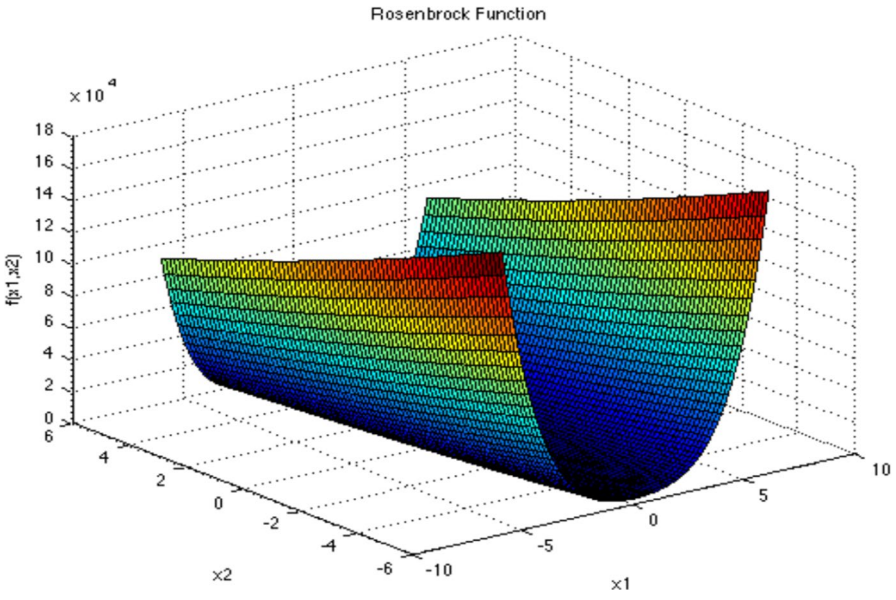

Fig. 3 The Rosenbrock function

Table 4 Summary of the results for Case D

	$N=2$	$N=5$	$N=10$	$N=20$	$N=30$	$N=50$	$N=100$
LDS iterative							
Objective F (in units of 1,000,000)	0.0196	0.1191	0.3129	1.1391	1.9168	3.7620	13.8132
CPU time (in sec.) for each iteration	0.0008	0.0013	0.0011	0.0017	0.0041	0.0053	0.0183
Dipole-like							
Objective F	0.4602	4.2127	124.7651	82.6268	342.2423	431.3262	3212.9200
CPU time (in sec.) for each iteration	0.0089	0.0197	0.0395	0.0774	0.1154	0.1986	0.3795

Although both methods use small CPU time (in sec.), the LDS iterative utilizes less CPU time (in sec.) than the dipole-like method.

Case F: The Zakharov function

In this case, we have to solve $\min\left\{F(p_1,\ldots,p_N)=\sum_{i=1}^{N}p_i^2+\left(\sum_{i=1}^{N}0.5ip_i\right)^2+\left(\sum_{i=1}^{N}0.5ip_i\right)^4\right\}$ where $p_i\in[-5,10]$, $i=1,2,\ldots,N$. The optimal solution of this problem is given by $p_i^*=0$, $i=1,2,\ldots,N$, and $F^*=0$. The Zakharov function (see Fig. 5) has no local minima except for the one that corresponds to the global minimum. Considering that the results of the objective are presented after taking Log (see Table 6), the dipole-like method shows significantly better performance than the benchmark.

Case G: Optimization under a random environment

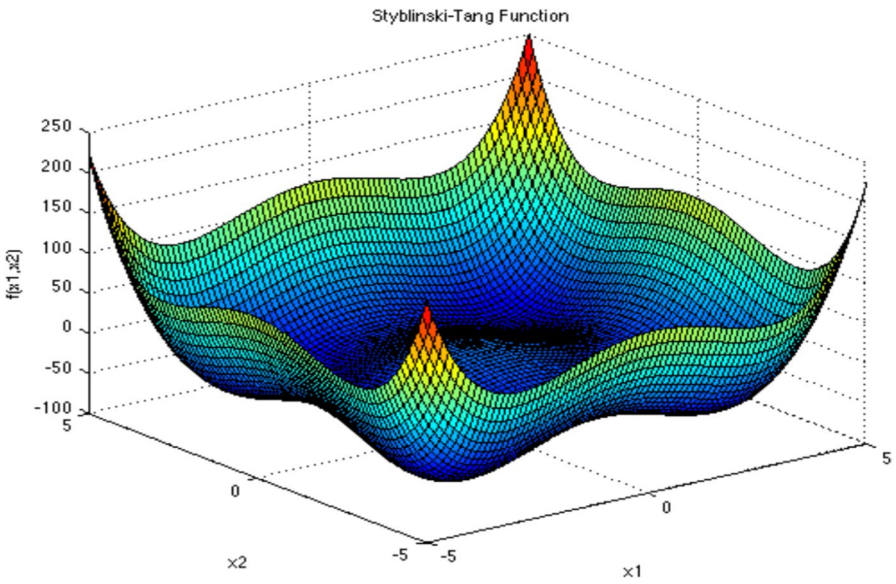

Fig. 4 The Styblinski–Tang function

Table 5 Summary of the results for Case E

	N=2	*N=5*	*N=10*	*N=20*	*N=30*	*N=50*	*N=100*
LDS iterative							
Objective *F/N*	− 11.637	− 10.532	− 6.041	3.394	26.769	48.047	69.468
CPU time (in sec.) for each iteration	0.0007	0.0008	0.0011	0.0042	0.0031	0.0051	0.0100
Dipole-like							
Objective *F/N*	− 39.163	− 39.162	− 34.920	− 34.916	− 36.790	− 34.607	− 32.408
CPU time (in sec.) for each iteration	0.0091	0.0229	0.0459	0.0915	0.1391	0.2280	0.4527

This example represents a scenario in which the environment changes on a regular basis (as is typical in real-time systems), and, as a result of these exogenous uncertainties, the objective produces inconsistent outcomes. To illustrate this situation, we assume the Dixon objective (see Case C), i.e., $z = F(p_1, ..., p_N) = (p_1 - 1)^2 + \sum_{i=2}^{N} i(2p_i^2 - p_{i-1})^2$ where $p_i \in [-10, 10]$, $i = 1,2,\ldots,N$, which is always positive, and we add a fraction ρ (positive or negative) to the objective z that represents the level of inconsistency, where $0 \le \rho \le 1$. Thus, the new objective is a random number, uniformly distributed over the

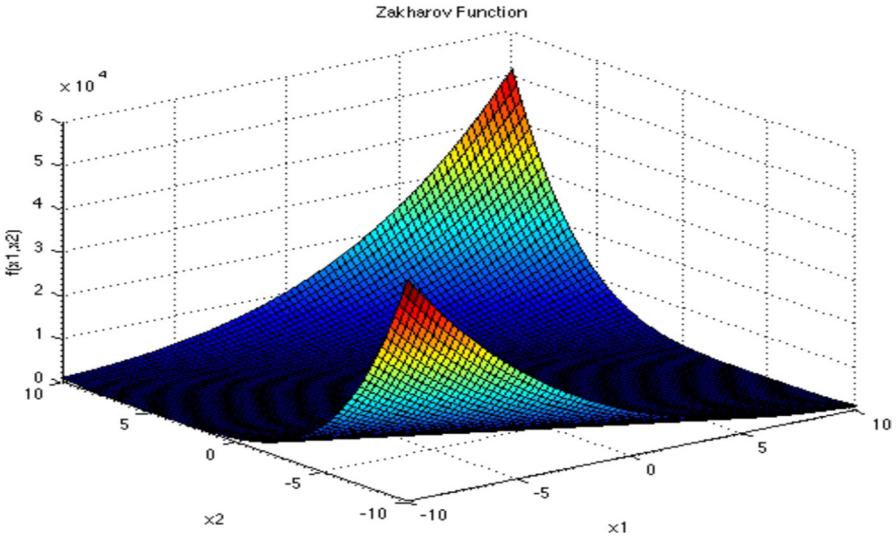

Fig. 5 The Zakharov function

Table 6 Summary of the results for Case F

	N=2	*N=5*	*N=10*	*N=20*	*N=30*	*N=50*	*N=100*
LDS iterative							
Objective *log(F)*	5.726	10.596	14.469	18.520	21.311	32.258	37.764
CPU time (in sec.) for each iteration	0.0004	0.0007	0.0012	0.0021	0.0030	0.0049	0.0088
Dipole-like							
Objective *log(F)*	− 6.458	− 1.579	3.249	8.345	11.567	13.310	18.686
CPU time (in sec.) for each iteration	0.0081	0.0211	0.0505	0.0779	0.1149	0.1927	0.3849

interval $[(1-\rho)z, (1+\rho)z]$. In this example, we assume a fixed, moderate problem size ($N=10$) and vary the level of inconsistency. Both the dipole-like and LDS-iterative methods show relatively stable performance (objective values) irrespective of the level of dynamic inconsistency (see Table 7). The example in Case G indicates that the performance of the dipole-like approach deteriorates slightly with the level of inconsistency, but for all values of ρ, it significantly outperforms the LDS-iterative approach.

Case H: The Styblinski–Tang function and the effect of the compression

In order to assess the potential of further compressing the searching domain as suggested in step 4 of the algorithm reported in Sect. 4.1, we consider this last case study, which repeats the example of the Styblinski–Tang function (see Case E) for several number of sub-intervals D_n. The results (see Table 8) indicate that the initial choice of $D_n = 10{,}000$ in our experiments can be slightly improved (approxomately 0.000003%) and in some cases even worsened, however with nearly 5-time greater CPU computional time. Yet, in cases wherease the initial number of subintervals is small (e.g., $D_n = 1000$), the potential in preformance improving is approx-

Table 7 Summary of the results for Case G

	$\rho = 0$	$\rho = 0.05$	$\rho = 0.1$	$\rho = 0.15$	$\rho = 0.2$	$\rho = 0.25$	$\rho = 0.3$
LDS iterative							
Objective F (in units of 1,000,000)	0.0355	0.0348	0.0395	0.0357	0.0354	0.0352	0.0349
CPU time (in sec.) for each iteration	0.0012	0.0012	0.0013	0.0012	0.0012	0.0013	0.0012
Dipole-like							
Objective F	1.538	1.619	1.841	1.801	1.858	1.860	1.898
CPU time (in sec.) for each iteration	0.0458	0.0458	0.0458	0.0458	0.0458	0.0458	0.0458

imately equal to 3%, obtained by shifting to a 5-time greater compression (i.e., $D_n = 10{,}000$), however again with nearly 5-time greater CPU computational time. We conclude that the possibility of compression may be appealing only for the cases where the initial number of subinterval is small, and in general it is case-based.

5.3 Summary of the simulation results

The results presented in the previous section demonstrate that the proposed dipole-like approach is a valuable, effective tool for parameter calibration in the case of continuous domains. This conclusion applies to almost all of the problem instances presented above. The advantages of the iterative LDS method relative to

Table 8 Summary of the results for Case H

	N=2	*N=5*	*N=10*	*N=20*	*N=30*	*N=50*	*N=100*
$D_n = 100$							
Objective F	− 39.117	− 36.300	− 36.296	− 34.883	− 36.747	− 33.972	− 33.282
CPU time (in sec.) for each iteration	0.0002628	0.00025	0.00035	0.00076	0.00102	0.00143	0.003183
$D_n = 500$							
Objective F	− 39.155	− 39.570	− 36.270	− 34.906	− 34.423	− 34.585	− 32.157
CPU time (in sec.) for each iteration	0.0003708	0.00087	0.00144	0.00249	0.00380	0.00659	0.013882
$D_n = 1000$							
Objective F	− 39.159	− 39.159	− 34.916	− 34.910	− 36.313	− 33.467	− 32.778
CPU time (in sec.) for each iteration	0.0008588	0.00166	0.00316	0.00597	0.00859	0.01473	0.029555
$D_n = 5000$							
Objective F	− 39.163	− 39.162	− 34.918	− 34.915	− 35.894	− 34.602	− 32.657
CPU time (in sec.) for each iteration	0.005145	0.00634	0.01238	0.02730	0.03573	0.06015	0.1283
$D_n = 10{,}000$							
Objective F	− 39.163	− 39.162	− 34.919	− 34.916	− 36.789	− 34.606	− 32.408
CPU time (in sec.) for each iteration	0.005120	0.01155	0.02344	0.04941	0.07109	0.11897	0.243189
$D_n = 50{,}000$							
Objective F	− 39.164	− 39.163	− 34.921	− 34.210	− 36.324	− 34.048	− 32.374
CPU time (in sec.) for each iteration	0.022748	0.05775	0.13700	0.28616	0.36145	0.56521	1.183

the dipole-like approach are mainly associated with faster run times for all problems and dimensions. Yet, it is worth noting that the longest CPU time recorded for the dipole-like approach in all trials is less than 0.5 s (in the absence of compression). This means that, for almost all known real-time black-box systems, the CPU time of the proposed approach would not be a barrier to applicability. We point out that the comparison between the dipole-like approach and random sampling with low-discrepancy sequences was performed with the same number of evaluations of the function F. The reduced computational requirements of the LDS iterative approach can be ascribed to the fact that it only requires the function F to be evaluated at the sample points, whereas in the dipole framework, this evaluation (see Step 2 in the algorithm in Sect. 4.1) occupies only a fraction of the time required for the overall application of the method, which also includes updating the various one-dimensional distribution functions. A major flaw of the LDS iterative method is the non-negligible likelihood that the best point determined up to the current time (i.e., in an earlier iteration) includes a position that is very close to one of the edges. When this occurs, it significantly reduces the size of the search space in future iterations, which could lead to poor

performance. The performance of the LDS iterative method could probably be improved (relative to the results reported in this paper) by increasing the number of sample points, at the price of higher computational effort. However, note that in complex, real-world systems, the evaluation of the function F may be very expensive, since it requires the considered system to be operated with a certain combination of parameters.

6 Conclusions and discussion

6.1 Main features of the proposed approach

An automatic mechanism for online black-box adaptive parameter estimation of complex processes has been developed. The proposed approach is well suited continuous parameter domains. The method involves building a sequence of random pairs of trials, called dipoles, in order to adapt online the probability density function of the unknown parameters. More specifically, the method updates the probability distributions of the parameters only in a small neighborhood of the two source points, while avoiding any significant changes in more distant regions of the search domain. The proposed algorithm seeks online the transient optimum, adapts itself according to the dynamic environmental changes, and improves the average performance over time by accumulating information from both successful and unsuccessful trials.

A theoretical analysis has been developed, and several numerical examples have been presented to demonstrate the applicability and effectiveness of the technique. The numerical simulations have shown that the proposed approach overcomes some of the difficulties that often arise in the estimation of the parameters of real-time systems, such as initial trial dependency, dependency on intrinsic coefficients, insufficient exploitation, and slow convergence for high-dimensional problems. Relying on just several numerical examples might not be sufficient to conclude validity, yet the self-improvement of the dipole-like method with subsequent iterations is generally proved (see Proposition 1). This implies that optimality or near-optimality is not reached arbitrarily. Moreover, the selected examples are commonly used to test optimization algorithms.

6.2 Managerial insights

Several practical aspects make the dipole-like algorithm attractive to both practitioners and theoreticians:

1. Relative to existing methods of online parameter estimation, the proposed approach places lower demands on computational resources: the overall operational history does not need to be stored since only the updated probability density function (and the best point to date) is used at each iteration. This property arises due to the operation at the core of the algorithm, which simply employs

one-dimensional probability density functions. This increases the applicability of the proposed approach, making it suitable for online parameter calibration of systems with a large number of parameters. On the contrary, alternative models of adaptive identification, such as online simulation, cluster analysis, and neural networks, typically require a large amount of historical data to be handled online (i.e., updated, sorted, and searched). Furthermore, the proposed dipole-like method guarantees high accuracy: the accuracy depends strongly on the chosen number D_n of sub-intervals for each dimension. Clearly, the higher the value of D_n, the higher the accuracy, but the greater the required computational effort. Hence, the system manager should attempt to achieve a tradeoff between accuracy and computational burden.

2. The compression step is well-suited to applications that involve continuous search spaces. However, it could also be adapted to processes with discrete parameter spaces. For instance, the values obtained after the compression stage could be rounded to the nearest element in the original discrete search space. However, further work would be required to adapt the method for use with discrete spaces.
3. Another alternative to running the computational experiments is by initially choosing a smaller number D_n of sub-intervals. However, to overcome the expected loss of accuracy, the compression step in the suggested dipole-like method (see step 4 of the algorithm reported in Sect. 4.1) is needed, and, accordingly, this will slowly increase D_n. This alterative may be appealing when the CPU computational time is expected to be significant for higher values of D_n. We leave the issue of finding the best initial number of sub-intervals under specified conditions for future research.
4. When implementing the proposed method in a real-life environment, the time interval between the two trials of a dipole should be as small as possible. This is because a shorter interval increases the similarity in the environmental conditions of the two trials, which in turn increases the efficiency and reliability of the method.

6.3 Limitations and future research directions

The proposed approach suffers from a number of limitations, which suggest directions for future research. First, our methodology assumes a given feasible domain of parameters. This assumption excludes the possibility that the transient optimum is located outside the search space. When the system environment becomes highly dynamic and can diverge substantially from that initially estimated, the probability of retaining a high level of consistency decreases. Further research is required to develop a mechanism that updates online the boundaries of the search domain in order to improve the efficiency of the optimization algorithm. Second, the dipole-like procedure assumes a high level of consistency between the conditions of the inputs and those of the outputs. Such consistency is vital for obtaining reliable information to update the random generators. For highly dynamic systems, especially when there is a significant delay between inserting the input parameters and measuring the output performance, this assumption may not hold, and thus the reliability of

the procedure would be expected to decrease. Further research is required to link the operational state of the system to a separate calibrating mechanism.

We have validated the suggested dipole-like method through applying eight different and well-known examples. This could diminish the generality of the conclusion under which the suggested method is relatively efficient. We suggest that future research works would test the dipole-like method in numerous other examples which consist of also higher dimensionality as well as different complexity.

Acknowledgements M. Gaggero and M. Sanguineti were partially supported by the Italian Ministry of Research (PRIN Project 2022S8XSMY) and by the FISA-2022-00827 Project "UAV-FIRE". M. Sanguineti was partially supported by the Next Generation EU program (PNRR 2022 Project "MOTUS") and by the project of the PDGP DIT.AD021.104 "Optimization and Control Techniques" of the Institute of Marine Engineering, National Research Council of Italy, where he is Research Associate. M. Gaggero and M. Sanguineti are members of GNAMPA (Gruppo Nazionale per l'Analisi Matematica, la Probabilità e le loro Applicazioni) at IndAM (Istituto Nazionale di Alta Matematica "Francesco Severi").

Funding Open access funding provided by Bar-Ilan University.

Data availability Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aarts, E., Korst, J.: Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing. Wiley, Hoboken (1997)
2. Abramson, M.A.: Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm. Optim. Eng. **5**(2), 157–177 (2004)
3. Alarie, S., Amaioua, N., Cyr, C.: Global optimization with NOMAD for the simultaneous tuning of several power system stabilizers. In: CORS/INFORMS International Conference, Montreal (2015)
4. Alarie, S., Audet, C., Gheribi, A. E., Kokkolaras, M., Le Digabel, S.: Two decades of blackbox optimization applications. Working paper (2020)
5. Arzi, Y., Herbon, A.: Machine learning based adaptive production control for a multi-cell flexible manufacturing system operating in a random environment. Int. J. Prod. Res. **38**(1), 161–185 (2000)
6. Audet, C., Dennis, J.E., Jr.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**, 188–217 (2006)
7. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Berlin (2017)

8. Audet, C., Le Digabel, S., Tribes, C.: The mesh adaptive direct search algorithm for granular and discrete variables. SIAM J. Optim. **29**(2), 1164–1189 (2019)
9. Baz, M., Hunsaker, B., Prokopyev, O.: How much do we "pay" for using default parameters? Comput. Optim. Appl. **48**(1), 91–108 (2011)
10. Bengoetxea, E.: Inexact graph matching using estimation of distribution algorithms. Ph.D. thesis, École Nationale Superieure des Telecommunications, Paris, France (2002)
11. Cardil, A., Monedero, S., Ramírez, J., Silva, C.A.: Assessing and reinitializing wildland fire simulations through satellite active fire data. J. Environ. Manage **231**, 996–1003 (2019)
12. Cervellera, C., Gaggero, M., Macciò, D.: Efficient kernel models for learning and approximate minimization problems. Neurocomputing **97**, 74–85 (2012)
13. Cervellera, C., Gaggero, M., Maccio, D.: Low-discrepancy sampling for approximate dynamic programming with local approximators. Comput. Oper. Res. **43**, 108–115 (2014)
14. Charbuillet, C., Gas, B., Chetouani, M., Zarader, J.: Optimizing feature complementarity by evolution strategy: application to automatic speaker verification. Speech Commun. **51**(9), 724–731 (2009)
15. Chen, X., Kelley, C.T., Xu, F., Zhang, Z.: A smoothing direct search method for Monte Carlo-based bound constrained composite nonsmooth optimization. SIAM J. Sci. Comput. **40**(4), A2174–A2199 (2018)
16. Choquette, Y., Lavigne, P., Ducharme, P., Houdayer, A., Martin, J.P.: Apparatus and method for monitoring snow water equivalent and soil moisture content using natural gamma radiation. US Patent No. 7800051 B2 (2010)
17. Clifton, J., Laber, E.: Q-Learning: theory and applications. Annu. Rev. Stat. Appl. **7**, 279–301 (2020)
18. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: Proceedings of the 1st European Conference on Artificial Life, pp. 134–142, Elsevier Publishing (1992)
19. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-free Optimization. MPS/SIAM Series on Optimization 8. SIAM, Philadelphia (2009)
20. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theor. Comput. Sci. **344**, 243–278 (2005)
21. Gaggero, M., Gnecco, G., Sanguineti, M.: Dynamic programming and value-function approximation in sequential decision problems: error analysis and numerical results. J. Optim. Theory Appl. **156**, 380–416 (2013)
22. Gaggero, M., Gnecco, G., Sanguineti, M.: Approximate dynamic programming for stochastic N-stage optimization with application to optimal consumption under uncertainty. Comput. Optim. Appl. **58**, 31–85 (2014)
23. Gheribi, A.E., Le Digabel, S., Audet, C., Chartrand, P.: Identifying optimal conditions for magnesium based alloy design using the Mesh Adaptive Direct Search algorithm. Thermochim. Acta **559**, 107–110 (2013)
24. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**, 533–549 (1986)
25. Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J.E, Sculley, D.: Google vizier: a service for black-box optimization. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2017)
26. Gramacy, R.B., Le Digabel, S.: The mesh adaptive direct search algorithm with treed Gaussian process surrogates. Les cahiers du Gerad, Technical Report No. G-2011-37 (2011). http://www.optimization-online.org/DB_HTML/2011/07/3090.html
27. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm Evol. Comput. **1**(3), 111–128 (2011)
28. Henderson, D., Jacobson, S.H., Johnson, A.W.: The theory and practice of simulated annealing. Handbook of metaheuristics, pp. 287–319 (2003)
29. Holland, J.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press/Bradford Books edition, Cambridge (1992)
30. Holland, J.: Genetic algorithms and the optimal allocation of trials. SIAM J. Comput. **2**, 88–105 (1973)
31. Huang, C., Li, Y., Yao, X.: A survey of automatic parameter tuning methods for metaheuristics. IEEE Trans. Evol. Comput. **24**(2), 201–216 (2020)

32. Ibáñez, O., Ballerini, L., Cordón, O., Damas, S., Santamaría, J.: An experimental study on the applicability of evolutionary algorithms to craniofacial superimposition in forensic identification. Inf. Sci. **179**(23), 3998–4028 (2009)
33. Jarraya, B., Bouri, A.: Metaheuristic optimization backgrounds: a literature review. Int. J. Contemp. Bus. Stud. **3**, 31–44 (2012)
34. Jebalia, M., Auger, A., Hansen, N.: Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. Algorithmica **59**, 425–460 (2011)
35. Jin, Y.: Surrogate-assisted evolutionary computation: recent advances and future challenges. Swarm Evol. Comput. **1**(2), 61–70 (2011)
36. Jones, D., Grisso, R.D.: Golden section search as an optimization tool for spreadsheets. Comput. Electron. Agric. **7**(4), 323–335 (1992)
37. Khazukov, K., Shepelev, V., Karpeta, T., Shabiev, S., Slobodin, I., Charbadze, I., Alferova, I.: Real-time monitoring of traffic parameters. J. Big Data **7**, 1–20 (2020)
38. Kolykhalova, K., Gnecco, G. Sanguineti, M., Volpe, G., Camurri, A.: Automated analysis of the origin of movement: An approach based on cooperative games on graphs. IEEE Trans. Hum. Mach. Syst. **50**, 550–560 (2020)
39. Koziel, S., Leifsson, L.: Surrogate-Based Modeling and Optimization. Springer, Berlin (2013)
40. Krejca, M.S., Witt, C.: Theory of estimation-of-distribution algorithms B. In: Doerr, Neumann, F. (eds.) Theory of Evolutionary Computation, Chapter 9. Natural Computing Series, pp. 405–442. Springer, Berlin (2018)
41. Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Genetic Algorithms and Evolutionary Computation, vol. 2. Springer, Berlin (2002)
42. Larson, J., Menickelly, M., Wild, S.M.: Derivative-free optimization methods. Acta Numer. **28**, 287–404 (2019)
43. Lewis, R.M., Torczon, V., Trosset, M.W.: Direct search methods: then and now. J. Comput. Appl. Math. **124**, 191–207 (2000)
44. Ouelhadj, D., Hanachi, C., Bouzouia, B.: Multi-agent architecture for distributed monitoring in flexible manufacturing systems (FMS). In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065),vol. 3, pp. 2416–2421. IEEE (2000)
45. Pelikan, M., Sastry, K., CantúPaz, E.: Scalable Optimization Via Probabilistic Modeling: From Algorithms to Applications. Studies in Computational Intelligence, vol. 33. Springer, Berlin (2006)
46. Pelikan, M., Hauschild, M., Lobo, F.G.: Estimation of distribution algorithms. In: Janusz Kacprzyk, W.P. (ed.) Handbook of Computational Intelligence, pp. 899–928 (2015)
47. Peprah, A.K., Simon, K.A., Amponsah, S.K.: An optimal cooling schedule using a simulated annealing based approach. Appl. Math. **8**, 1195–1210 (2017)
48. Picheny, V., Wagner, T., Ginsbourger, D.: A benchmark of kriging-based infill criteria for noisy optimization. Struct. Multidiscip. Optim. **48**, 607–626 (2013)
49. Powell, M.J.D.: The NEWUOA software for unconstrained optimization without derivatives. In: Di Pillo, G., Roma, M. (eds.) Large-Scale Nonlinear Optimization. Springer, New York (2006)
50. Powell, M.J.D.: The BOBYQA algorithm for bound constrained optimization without derivatives. Report No. DAMTP 2009/NA06, CMS, University of Cambridge (2009)
51. Razavi, S., Tolson, B.A., Burn, D.H.: Review of surrogate modeling in water resources. Water Resour. Res. **48**(7) (2012). https://doi.org/10.1029/2011WR011527
52. Reeves, C.R.: Genetic Algorithms: Principles and Perspective. Kluwer, Boston (2003)
53. Roseline, R.A., Devapriya, M., Sumathi, P.: Pollution monitoring using sensors and wireless sensor networks: a survey. Int. J. Appl. Innov. Eng. Manag. **2**(7), 119–124 (2013)
54. Roy, N., Thrun, S.: Online self-calibration for mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (1999)
55. Rubinstein, R.Y.: The simulated entropy method for combinatorial and continuous optimization. Methodol. Comput. Appl. Probab. **2**, 127–190 (1999)
56. Schaul, T.: Studies in Continuous Black-Box Optimization. Ph.D. Thesis, Munich University (2011)
57. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423 (1948)
58. Surjanovic, S. & Bingham, D.: Virtual library of simulation experiments: test functions and datasets. Retrieved January 26, 2024, from http://www.sfu.ca/~ssurjano (2013)
59. Wales, D., Doye, J.: Global optimization by Basin–Hopping and the lowest energy structures of Lennard–Jones clusters containing up to 110 atoms. J. Phys. Chem. A **101**(28), 5111–5116 (1997)
60. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. Thesis, Cambridge University (1989)

61. Winter, S., Brendel, B., Igel, C.: Registration of bone structures in 3D ultrasound and CT data: comparison of different optimization strategies. Int. Congr. Ser. **1281**, 242–247 (2005)
62. Xi, M., Sun, W., Chen, J.: Survey of derivative-free optimization. Numer. Algebra Control Optim. **10**(4), 537–555 (2020)
63. Zweig, G., Siohan, O., Saon, G., Ramabhadran, B., Povey, D., Mangu, L., Kingsbury, B.: Automated quality monitoring for call centers using speech and NLP technologies. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology—Companion volume: Demonstrations, pp. 292–295 (2006)

Authors and Affiliations

Avi Herbon[1] · Mauro Gaggero[2] · Eugene Khmelnitsky[3] · Marcello Sanguineti[4]

✉ Avi Herbon
avher@bezeqint.net

[1] Department of Management, Bar-Ilan University, 5290002 Ramat-Gan, Israel

[2] Institute of Marine Engineering, National Research Council of Italy, Via De Marini 6, 16149 Genoa, Italy

[3] Department of Industrial Engineering, Tel Aviv University, 6997801 Tel Aviv, Israel

[4] Department of Computer Science, Bioengineering, Robotics, and Systems Engineering, University of Genoa, Via Opera Pia 13, 16145 Genoa, Italy