An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation

Erik Cuevas · Alonso Echavarría · Marte A. Ramírez-Ortegón

Abstract The ability of an Evolutionary Algorithm (EA) to find a global optimal solution depends on its capacity to find a good rate between exploitation of found-so-far elements and exploration of the search space. Inspired by natural phenomena, researchers have developed many successful evolutionary algorithms which, at original versions, define operators that mimic the way nature solves complex problems, with no actual consideration of the exploration-exploitation balance. In this paper, a novel nature-inspired algorithm called the States of Matter Search (SMS) is introduced. The SMS algorithm is based on the simulation of the states of matter phenomenon. In SMS, individuals emulate molecules which interact to each other by using evolutionary operations which are based on the physical principles of the thermal-energy motion mechanism. The algorithm is devised by considering each state of matter at one different exploration-exploitation ratio. The evolutionary process is divided into three phases which emulate the three states of matter: gas, liquid and solid. In each state, molecules (individuals) exhibit different movement capacities. Beginning from the gas state (pure exploration), the algorithm modifies the intensities of exploration and exploitation until the solid state (pure exploitation) is reached. As a result, the approach can substantially improve the balance between exploration-exploitation, yet preserving the good search capabilities of an evolutionary approach. To illustrate the proficiency and robustness of the proposed algorithm, it is compared to other well-known evolutionary methods including novel variants that incorporate diversity preservation schemes. The comparison examines several standard benchmark functions which are commonly considered within the EA field. Experimental results show that the proposed method achieves a good performance in comparison to its counterparts as a consequence of its better exploration-exploitation balance.

Keywords Evolutionary algorithms · Global optimization · Nature-inspired algorithms

1 Introduction

Global optimization [1] has delivered applications for many areas of science, engineering, economics and others, where mathematical modeling is used [2]. In general, the goal is to find a global optimum for an objective function which is defined over a given search space. Global optimization algorithms are usually broadly divided into deterministic and stochastic methods [3]. Since deterministic methods only provide a theoretical guarantee of locating a local minimum of the objective function, they often face great difficulties in solving global optimization problems [4]. On the other hand, evolutionary algorithms are usually faster in locating a global optimum [5]. Moreover, stochastic methods adapt easily to black-box formulations and extremely ill-behaved functions, whereas deterministic methods usually rest on at least some theoretical assumptions about the problem formulation and its analytical properties (such as Lipschitz continuity) [6].

Evolutionary algorithms, which are considered as members of the stochastic group, have been developed by a combination of rules and randomness that mimics several natu-
An optimization algorithm inspired by the States of Matter that improves the balance

ral phenomena. Such phenomena include evolutionary processes such as the Evolutionary Algorithm (EA) proposed by Fogel et al. [7], De Jong [8], and Koza [9], the Genetic Algorithm (GA) proposed by Holland [10] and Goldberg [11], the Artificial Immune System proposed by De Castro et al. [12] and the Differential Evolution Algorithm (DE) proposed by Price and Storn [13]. Some other methods which are based on physical processes include the Simulated Annealing proposed by Kirkpatrick et al. [14], the Electromagnetism-like Algorithm proposed by Iker et al. [15] and the Gravitational Search Algorithm proposed by Rashedi et al. [16]. Also, there are other methods based on the animal-behavior phenomena such as the Particle Swarm Optimization (PSO) algorithm proposed by Kennedy and Eberhart [17] and the Ant Colony Optimization (ACO) algorithm proposed by Dorigo et al. [18].

Every EA needs to address the issue of exploration-exploitation of the search space. Exploration is the process of visiting entirely new points of a search space whilst exploitation is the process of refining those points within the neighborhood of previously visited locations, in order to improve their solution quality. Pure exploration degrades the precision of the evolutionary process but increases its capacity to find new potential solutions. On the other hand, pure exploitation allows refining existing solutions but adversely driving the process to local optimal solutions. Therefore, the ability of an EA to find a global optimal solution depends on its capacity to find a good balance between the exploitation of found-so-far elements and the exploration of the search space [19]. So far, the exploration-exploitation dilemma has been an unsolved issue within the framework of EA.

Although PSO, DE and GSA are considered the most popular algorithms for many optimization applications, they fail in finding a balance between exploration and exploitation [20]; in multimodal functions, they do not explore the whole region effectively and often suffers premature convergence or loss of diversity. In order to deal with this problem, several proposals have been suggested in the literature [21–46]. In most of the approaches, exploration and exploitation is modified by the proper settings of control parameters that have an influence on the algorithm’s search capabilities [47]. One common strategy is that EAs should start with exploration and then gradually change into exploitation [48]. Such a policy can be easily described with deterministic approaches where the operator that controls the individual diversity decreases along with the evolution. This is generally correct, but such a policy tends to face difficulties when solving certain problems with multimodal functions that hold many optima, since a premature takeover of exploitation over exploration occurs. Some approaches that use this strategy can be found in [21–29]. Other works [30–34] use the population size as reference to change the balance between exploration and exploitation. A larger population size implies a wider exploration while a smaller population demands a shorter search. Although this technique delivers an easier way to keep diversity, it often represents an unsatisfactory solution. An improper handling of large populations might converge to only one point, despite introducing more function evaluations. Recently, new operators have been added to several traditional evolutionary algorithms in order to improve their original exploration-exploitation capability. Such operators diversify particles whenever they concentrate on a local optimum. Some methods that employ this technique are discussed in [35–46].

Either of these approaches is necessary but not sufficient to tackle the problem of the exploration-exploitation balance. Modifying the control parameters during the evolution process without the incorporation of new operators to improve the population diversity makes the algorithm defenseless against premature convergence and may result in poor exploratory characteristics of the algorithm [48]. On the other hand, incorporating new operators without modifying the control parameters leads to an increase in computational cost and weakens the exploitation process of candidate regions [39]. Therefore, it does seem reasonable to incorporate both of these approaches into a single algorithm.

In this paper, a novel nature-inspired algorithm, known as the States of Matter Search (SMS) is proposed for solving global optimization problems. The SMS algorithm is based on the simulation of the states of matter phenomenon. In SMS, individuals emulate molecules which interact to each other by using evolutionary operations based on the physical principles of the thermal-energy motion mechanism. Such operations allow the increase of the population diversity and avoid the concentration of particles within a local minimum. The proposed approach combines the use of the defined operators with a control strategy that modifies the parameter setting of each operation during the evolution process. In contrast to other approaches that enhance traditional EA algorithms by incorporating some procedures for balancing the exploration-exploitation rate, the proposed algorithm naturally delivers such property as a result of mimicking the states of matter phenomenon. The algorithm is devised by considering each state of matter at one different exploration-exploitation ratio. Thus, the evolutionary process is divided into three stages which emulate the three states of matter: gas, liquid and solid. At each state, molecules (individuals) exhibit different behaviors. Beginning from the gas state (pure exploration), the algorithm modifies the intensities of exploration and exploitation until the solid state (pure exploitation) is reached. As a result, the approach can substantially improve the balance between exploration-exploitation, yet preserving the good search capabilities of an evolutionary approach. To illustrate the proficiency and robustness of the proposed algorithm, it has been compared to other well-known evolutionary methods including recent variants that incorporate diversity preservation.

© Springer
schemes. The comparison examines several standard benchmark functions which are usually employed within the EA field. Experimental results show that the proposed method achieves good performance over its counterparts as a consequence of its better exploration–exploitation capability.

This paper is organized as follows. Section 2 introduces basic characteristics of the three states of matter. In Sect. 3, the novel SMS algorithm and its characteristics are both described. Section 4 presents experimental results and a comparative study. Finally, in Sect. 5, some conclusions are discussed.

2 States of matter

The matter can take different phases which are commonly known as states. Traditionally, three states of matter are known: solid, liquid, and gas. The differences among such states are based on forces which are exerted among particles composing a material [49].

In the gas phase, molecules present enough kinetic energy so that the effect of intermolecular forces is small (or zero for an ideal gas), while the typical distance between neighboring molecules is greater than the molecular size. A gas has no definite shape or volume, but occupies the entire container in which it is confined. Figure 1a shows the movements exerted by particles in a gas state. The movement experimented by the molecules represent the maximum permissible displacement $\rho_1$ among particles [50]. In a liquid state, intermolecular forces are more restrictive than those in the gas state. The molecules have enough energy to move relatively to each other still keeping a mobile structure. Therefore, the shape of a liquid is not definite but is determined by its container. Figure 1b presents a particle movement $\rho_2$ within a liquid state. Such movement is smaller than those considered by the gas state but larger than the solid state [51]. In the solid state, particles (or molecules) are packed together closely with forces among particles being strong enough so that the particles cannot move freely but only vibrate. As a result, a solid has a stable, definite shape and a definite volume. Solids can only change their shape by force, as when they are broken or cut. Figure 1c shows a molecule configuration in a solid state. Under such conditions, particles are able to vibrate (being perturbed) considering a minimal $\rho_3$ distance [50].

In this paper, a novel nature-inspired algorithm known as the States of Matter Search (SMS) is proposed for solving global optimization problems. The SMS algorithm is based on the simulation of the states of matter phenomenon that considers individuals as molecules which interact to each other by using evolutionary operations based on the physical principles of the thermal-energy motion mechanism. The algorithm is devised by considering each state of matter at one different exploration–exploitation ratio. Thus, the evolutionary process is divided into three stages which emulate the three states of matter: gas, liquid and solid. In each state, individuals exhibit different behaviors.

3 States of matter search (SMS)

3.1 Definition of operators

In the approach, individuals are considered as molecules whose positions on a multidimensional space are modified as the algorithm evolves. The movement of such molecules is motivated by the analogy to the motion of thermal-energy.

The velocity and direction of each molecule’s movement are determined by considering the collision, the attraction forces and the random phenomena experimented by the molecule set [52]. In our approach, such behaviors have been implemented by defining several operators such as the direction vector, the collision and the random positions operators, all of which emulate the behavior of actual physics laws.

The direction vector operator assigns a direction to each molecule in order to lead the particle movement as the evolution process takes place. On the other side, the collision operator mimics those collisions that are experimented by molecules as they interact to each other. A collision is considered when the distance between two molecules is shorter than a determined proximity distance. The collision operator is thus implemented by interchanging directions of the
involved molecules. In order to simulate the random behavior of molecules, the proposed algorithm generates random positions following a probabilistic criterion that considers random locations within a feasible search space.

The next section presents all operators that are used in the algorithm. Although such operators are the same for all the states of matter, they are employed over a different configuration set depending on the particular state under consideration.

### 3.1.1 Direction vector

The direction vector operator mimics the way in which molecules change their positions as the evolution process develops. For each n-dimensional molecule \( \mathbf{p}_i \) from the population \( \mathbf{P} \), it is assigned an n-dimensional direction vector \( \mathbf{d}_i \) which stores the vector that controls the particle movement. Initially, all the direction vectors \( \mathbf{D} = \{ \mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{N_p} \} \) are randomly chosen within the range of \([-1, 1]\).

As the system evolves, molecules experiment several attraction forces. In order to simulate such forces, the proposed algorithm implements the attraction phenomenon by moving each molecule towards the best so-far particle. Therefore, the new direction vector for each molecule is iteratively computed considering the following model:

\[
\mathbf{d}_i^{k+1} = \mathbf{d}_i^k + \left( 1 - \frac{k}{\text{gen}} \right) \cdot 0.5 + \mathbf{a}_i \tag{1}
\]

where \( \mathbf{a}_i \) represents the attraction unitary vector calculated as \( \mathbf{a}_i = (\mathbf{p}_{\text{best}}^i - \mathbf{p}_i)/\|\mathbf{p}_{\text{best}}^i - \mathbf{p}_i\| \), being \( \mathbf{p}_{\text{best}}^i \) the best individual seen so-far, while \( \mathbf{p}_i \) is the molecule \( i \) of population \( \mathbf{P} \). \( k \) represents the iteration number whereas \( \text{gen} \) involves the total iteration number that constitutes the complete evolution process.

Under this operation, each particle is moved towards a new direction which combines the past direction, which was initially computed, with the attraction vector over the best individual seen so-far. It is important to point out that the relative importance of the past direction decreases as the evolving process advances. This particular type of interaction avoids the quick concentration of information among particles and encourages each particle to search around a local candidate region in its neighborhood, rather than interacting to a particle lying at distant region of the domain. The use of this scheme has two advantages: first, it prevents the particles from moving toward the global best position in early stages of algorithm and thus makes the algorithm less susceptible to premature convergence; second, it encourages particles to explore their own neighborhood thoroughly, just before they converge towards a global best position. Therefore, it provides the algorithm with local search ability enhancing the exploitative behavior.

In order to calculate the new molecule position, it is necessary to compute the velocity \( \mathbf{v}_i \) of each molecule by using:

\[
\mathbf{v}_i = \mathbf{d}_i \cdot v_{\text{init}} \tag{2}
\]

being \( v_{\text{init}} \) the initial velocity magnitude which is calculated as follows:

\[
v_{\text{init}} = \frac{\sum_{j=1}^n (b_j^{\text{high}} - b_j^{\text{low}})}{n} \cdot \beta \tag{3}
\]

where \( b_j^{\text{low}} \) and \( b_j^{\text{high}} \) are the low \( j \) parameter bound and the upper \( j \) parameter bound respectively, whereas \( \beta \in [0, 1] \).

Then, the new position for each molecule is updated by:

\[
p_i^{k+1} = p_i^k + v_{i,j} \cdot \text{rand}(0, 1) \cdot \rho \cdot (b_j^{\text{high}} - b_j^{\text{low}}) \tag{4}
\]

where \( 0.5 \leq \rho \leq 1 \).

### 3.1.2 Collision

The collision operator mimics the collisions experimented by molecules while they interact to each other. Collisions are calculated if the distance between two molecules is shorter than a determined proximity value. Therefore, if \( \|\mathbf{p}_i - \mathbf{p}_j\| < r \), a collision between molecules \( i \) and \( q \) is assumed; otherwise, there is no collision, considering \( i, q \in \{1, \ldots, N_p\} \) such that \( i \neq q \). If a collision occurs, the direction vector for each particle is modified by interchanging their respective direction vectors as follows:

\[
\mathbf{d}_i = \mathbf{d}_q \quad \text{and} \quad \mathbf{d}_q = \mathbf{d}_i \tag{5}
\]

The collision radius is calculated by:

\[
r = \frac{\sum_{j=1}^n (b_j^{\text{high}} - b_j^{\text{low}})}{n} \cdot \alpha \tag{6}
\]

where \( \alpha \in [0, 1] \).

Under this operator, a spatial region enclosed within the radius \( r \) is assigned to each particle. In case the particle regions collide with each other, the collision operator acts upon particles by forcing them out of the region. The radius \( r \) and the collision operator provide the ability to control diversity throughout the search process. In other words, the rate of increase or decrease of diversity is predetermined for each stage. Unlike other diversity-guided algorithms, it is not necessary to inject diversity into the population when particles gather around a local optimum because the diversity will be preserved during the overall search process. The collision incorporation therefore enhances the exploratory behavior in the proposed approach.
3.1.3 Random positions

In order to simulate the random behavior of molecules, the proposed algorithm generates random positions following a probabilistic criterion within a feasible search space.

For this operation, a uniform random number \( r_m \) is generated within the range \([0, 1]\). If \( r_m \) is smaller than a threshold \( H \), a random molecule’s position is generated; otherwise, the element remains with no change. Therefore, such an operation can be modeled as follows:

\[
    p_{i,j}^{k+1} = \begin{cases} 
        b_{j}^{low} + \text{rand}(0, 1) \cdot (b_{j}^{high} - b_{j}^{low}) & \text{with probability } H \\
        p_{i,j}^{k+1} & \text{with probability } (1 - H)
    \end{cases}
\]  

(7)

where \( i \in \{1, \ldots, N_p\} \) and \( j \in \{1, \ldots, n\} \).

3.1.4 Best Element Updating

Despite the fact that this updating operator does not belong to the State of Matter metaphor, it is used to simply store the best so-far solution. In order to update the best molecule \( \mathbf{p}^{best} \) seen so-far, the best found individual from the current \( k \) population \( \mathbf{p}^{best,k} \) is compared to the best individual \( \mathbf{p}^{best,k-1} \) of the last generation. If \( \mathbf{p}^{best,k} \) is better than \( \mathbf{p}^{best,k-1} \) according to its fitness value, \( \mathbf{p}^{best} \) is updated with \( \mathbf{p}^{best,k} \), otherwise \( \mathbf{p}^{best} \) remains with no change. Therefore, \( \mathbf{p}^{best} \) stores the best historical individual found so-far.

3.2 SMS algorithm

The overall SMS algorithm is composed of three stages corresponding to the three States of Matter: the gas, the liquid and the solid state. Each stage has its own behavior. In the first stage (gas state), exploration is intensified whereas in the second one (liquid state) a mild transition between exploration and exploitation is executed. Finally, in the third phase (solid state), solutions are refined by emphasizing the exploitation process.

3.2.1 General procedure

At each stage, the same operations are implemented. However, depending on which state is referred, they are employed considering a different parameter configuration. The general procedure in each state is shown as pseudo-code in Algorithm 1. Such procedure is composed of five steps and maps the current population \( \mathbf{P}^k \) to a new population \( \mathbf{P}^{k+1} \). The algorithm receives as input the current population \( \mathbf{P}^k \) and the configuration parameters \( \rho, \beta, \alpha, \) and \( H \), whereby it yields the new population \( \mathbf{P}^{k+1} \).

### Algorithm 1 General procedure executed by all the states of matter

**Step 1:** Find the best element of the population \( \mathbf{P} \)

\[
    \mathbf{p}^{best} \in \{\mathbf{P}\} | f(\mathbf{p}^{best}) = \max \{ f(\mathbf{p}_1), f(\mathbf{p}_2), \ldots, f(\mathbf{p}_{N_p}) \}
\]

**Step 2:** Calculate \( \mathbf{v}_{init} \) and \( r \)

\[
    \mathbf{v}_{init} = \frac{\sum_{j=1}^{n}(b_{j}^{high} - b_{j}^{low}) \cdot \beta}{n} \quad r = \frac{\sum_{j=1}^{n}(b_{j}^{high} - b_{j}^{low}) \cdot \alpha}{n}
\]

**Step 3:** Compute the new molecules by using the Direction vector operator 3.1.1

\[
    a_i = (\mathbf{p}^{best} - \mathbf{p}_i) / \|\mathbf{p}^{best} - \mathbf{p}_i\|
\]

for \((j = 1; j < n + 1; j++)\)

\[
    d_{i,j}^{k+1} = d_{i,j}^{k} \cdot \left( 1 - \frac{k}{\text{gen}} \right) \cdot 0.5 + a_{i,j}
\]

\[
    v_{i,j} = d_{i,j}^{k+1} \cdot \mathbf{v}_{init}
\]

\[
    p_{i,j}^{k+1} = p_{i,j}^{k} + v_{i,j} \cdot \text{rand}(0, 1) \cdot \rho \cdot (b_{j}^{high} - b_{j}^{low})
\]

end for

end for

**Step 4:** Solve collisions by using the Collision operator 3.1.2

for \((i = 1; i < N_p + 1; i++)\)

for \((j = 1; j < N_p + 1; j++)\)

if \((\|\mathbf{p}_i - \mathbf{p}_j\| < r) \text{ and } (i \neq j)\)

\[
    t = d_i
\]

\[
    d_i = d_j
\]

\[
    d_j = t
\]

end if

end for

end for

**Step 5:** Generate new random positions by using the Random positions operator 3.1.3

for \((i = 1; i < N_p + 1; i++)\)

if \((r_m < H)\) then; where \( r_m \in \text{rand}(0, 1) \)

for \((j = 1; j < n + 1; j++)\)

\[
    p_{i,j}^{k+1} = b_{j}^{low} + \text{rand}(0, 1) \cdot (b_{j}^{high} - b_{j}^{low})
\]

end for

end if

end for

end for
An optimization algorithm inspired by the States of Matter that improves the balance

3.2.2 The complete algorithm

The complete algorithm is divided into four different parts. The first corresponds to the initialization stage, whereas the last three represent the States of Matter. All the optimization process, which consists of a gen number of iterations, is organized into three different asymmetric phases, employing 50% of all iterations for the gas state (exploration), 40% for the liquid state (exploration–exploitation) and 10% for the solid state (exploitation). The overall process is graphically described by Fig. 2. At each state, the same general procedure (see Algorithm 1) is iteratively used considering the particular configuration predefined for each State of Matter. Figure 3 shows the data flow for the complete SMS algorithm.

Initialization  The algorithm begins by initializing a set P of Np molecules (P = {p1, p2, ..., pNp}). Each molecule position pi is an n-dimensional vector containing the parameter values to be optimized. Such values are randomly and uniformly distributed between the pre-specified lower initial parameter bound b_lowj and the upper initial parameter bound b_highj, just as it is described by the following expressions:

\[ p_{i,j}^0 = b_{\text{low}}^j + \text{rand}(0, 1) \cdot (b_{\text{high}}^j - b_{\text{low}}^j), \]

\[ j = 1, 2, \ldots, n; \quad i = 1, 2, \ldots, N_p, \]

where j and i are the parameter and molecule index, respectively, whereas zero indicates the initial population. Hence, pij is the j-th parameter of the i-th molecule.

Gas state  In the gas state, molecules experiment severe displacements and collisions. Such state is characterized by random movements produced by non-modeled molecule phenomena [52]. Therefore, the ρ value from the direction
the solid state can be summarized as follows: The computational procedure for the gas state can be summarized as follows:

Step 1: Set the parameters $\rho \in [0.8, 1]$, $\beta = 0.8$, $\alpha = 0.8$ and $H = 0.9$ being consistent with the solid state.

Step 2: Apply the general procedure which is illustrated in Algorithm 1.

Step 3: If 50% of the total iteration number is completed ($1 \leq k \leq 0.5 \cdot \text{gen}$), then the process continues to the liquid state procedure; otherwise go back to step 2.

Step 4: Set the parameters $\rho \in [0.3, 0.6]$, $\beta = 0.4$, $\alpha = 0.2$ and $H = 0.2$ being consistent with the liquid state.

Step 5: Apply the general procedure that is defined in Algorithm 1.

Step 6: If 90% (50% from the gas state and 40% from the liquid state) of the total iteration number is completed ($0.5 \cdot \text{gen} < k \leq 0.9 \cdot \text{gen}$), then the process continues to the solid state procedure; otherwise go back to step 5.

Liquid state Although molecules currently at the liquid state exhibit restricted motion in comparison to the gas state, they still show a higher flexibility with respect to the solid state. Furthermore, the generation of random positions which are produced by non-modeled molecule phenomena is scarce [53]. For this reason, the $\rho$ value from the direction vector operator is bounded to a value between 0.3 and 0.6. Similarly, the random position operator $H$ is configured to a value close to zero in order to allow the random generation of fewer molecule positions. In the liquid state, collisions are also less common than in the gas state, so the collision radius, that is controlled by $\alpha$, is set to a smaller value in comparison to the gas state. The liquid state is the second phase and lasts for the 40% of all iterations which compose the complete optimization process. The computational procedure for the liquid state can be summarized as follows:

Step 7: Set the parameters $\rho \in [0.0, 0.1]$ and $\beta = 0.1$, $\alpha = 0$ and $H = 0$ being consistent with the solid state.

Step 8: Apply the general procedure that is defined in Algorithm 1.

Step 9: If 100% of the total iteration number is completed ($0.9 \cdot \text{gen} < k \leq \text{gen}$), the process is finished; otherwise go back to step 8.

It is important to clarify that the use of this particular configuration ($\alpha = 0$ and $H = 0$) disables the collision and generation of random positions operators, which have been illustrated in the general procedure.

4 Experimental results

A comprehensive set of 24 functions, collected from Refs. [54–61], has been used to test the performance of the proposed approach. Tables 8–11 in Appendix A present the benchmark functions used in our experimental study. Such functions are classified into four different categories: Unimodal test functions (Table 8), multimodal test functions (Table 9), multimodal test functions with fixed dimensions (Table 10) and functions proposed for the GECCO contest (Table 11). In such tables, $n$ indicates the dimension of the function, $f_{\text{opt}}$ is the optimum value of the function and $S$ is the subset of $R^n$. The function optimum position ($x_{\text{opt}}$) for $f_1$, $f_2$, $f_4$, $f_6$, $f_7$, $f_{10}$, $f_{11}$ and $f_{14}$ is at $x_{\text{opt}} = [0]^n$, for $f_3$, $f_8$ and $f_9$ is at $x_{\text{opt}} = [1]^n$, for $f_5$ is at $x_{\text{opt}} = [420.96]^n$, for $f_{18}$ is at $x_{\text{opt}} = [0]^n$, for $f_{12}$ is at $x_{\text{opt}} = [0.0003075]^n$ and for $f_{13}$ is at $x_{\text{opt}} = [-3.32]^n$. In case of functions contained in Table 11, the $x^{\text{opt}}$ and $f_{\text{opt}}$ values have been set to default values which have been obtained from the Matlab© implementation for GECCO competitions, as it is provided in [59]. A detailed description of optimum locations is given in Appendix A.

4.1 Performance comparison to other meta-heuristic algorithms

We have applied the SMS algorithm to 24 functions whose results have been compared to those produced by the Gravitational Search Algorithm (GSA) [16], the Particle Swarm Optimization (PSO) method [17] and the Differential Evolution (DE) algorithm [13]. These are considered the most popular algorithms in many optimization applications. In order to enhance the performance analysis, the PSO algorithm with a territorial diversity-preserving scheme (TPSO) [39] has also been added into the comparisons. TPSO is considered a recent PSO variant that incorporates a diversity preservation scheme in order to improve the balance between exploration and exploitation. In all comparisons, the population has been set to 50. The maximum iteration number for functions in Tables 8, 9 and 11 has been set to 1000.
and for functions in Table 10 the iterations have been set to 500. Such stop criterion has been selected to maintain compatibility to similar works reported in the literature [4, 16].

The parameter setting for each algorithm in the comparison is described as follows:

1. **GSA [16]:** The parameters are set to $G_0 = 100$ and $\alpha = 20$; the total number of iterations is set to 1000 for functions $f_1$ to $f_{11}$ and 500 for functions $f_{12}$ to $f_{14}$. The total number of individuals is set to 50. Such values are the best parameter set for this algorithm according to [16].

2. **PSO [17]:** The parameters are set to $c_1 = 2$ and $c_2 = 2$; besides, the weight factor decreases linearly from 0.9 to 0.2.

3. **DE [13]:** The DE/Rand/1 scheme is employed. The crossover probability is set to $CR = 0.9$ and the weight factor is set to $F = 0.8$.

4. **TPSO [39]:** The parameter $\alpha$ has been set to 0.5. This value is found to be the best configuration according to [39]. The algorithm has been tuned according to the set of values which have been originally proposed by its own reference.

The experimental comparison between metaheuristic algorithms, with respect to SMS, has been developed according to the function-type classification as follows:

1. **Unimodal test functions** (Table 8).
2. **Multimodal test functions** (Table 9).
3. **Multimodal test functions with fixed dimension** (Table 10).
4. **Test functions from the GECCO contest** (Table 11).

### 4.1.1 Unimodal test functions

This experiment is performed over the functions presented in Table 8. The test compares the SMS to other algorithms such as GSA, PSO, DE and TPSO. The results for 30 runs are reported in Table 1 considering the following performance indexes: the Average Best-so-far (AB) solution, the Median Best-so-far (MB) and the Standard Deviation (SD) of best-so-far solution. The best outcome for each function is boldfaced. According to this table, SMS delivers better results than GSA, PSO, DE and TPSO for all functions. In particular, the test remarks the largest difference in performance, which is directly related to a better trade-off between exploration and exploitation. Just as it is illustrated by Fig. 4, SMS, DE and GSA have similar convergence rates at finding the optimal minimal, yet are faster than PSO and TPSO.

A non-parametric statistical significance proof known as the Wilcoxon’s rank sum test for independent samples [62, 63] has been conducted over the “average best-so-far” (AB) values from Table 3.

### Table 2 $p$-values produced by Wilcoxon’s test comparing SMS vs. PSO, SMS vs. GSA, SMS vs. DE and SMS vs. TPSO over the “average best-so-far” (AB) values from Table 3

|       | SMS vs PSO | SMS vs GSA | SMS vs DE | SMS vs TPSO |
|-------|------------|------------|-----------|-------------|
| $f_1$ | $3.94 \times 10^{-5}$ | $7.39 \times 10^{-4}$ | $1.04 \times 10^{-6}$ | $4.12 \times 10^{-4}$ |
| $f_2$ | $5.62 \times 10^{-5}$ | $4.92 \times 10^{-4}$ | $2.21 \times 10^{-6}$ | $3.78 \times 10^{-4}$ |
| $f_3$ | $6.42 \times 10^{-8}$ | $7.11 \times 10^{-7}$ | $1.02 \times 10^{-4}$ | $1.57 \times 10^{-4}$ |
| $f_4$ | $1.91 \times 10^{-8}$ | $7.39 \times 10^{-4}$ | $1.27 \times 10^{-6}$ | $4.22 \times 10^{-4}$ |


Fig. 4 Convergence rate comparison of GSA, PSO, DE, SMS and TPSO for minimization of (a) \( f_1 \) and (b) \( f_3 \) considering \( n = 30 \)

Table 3 Minimization result of benchmark functions in Table 9 with \( n = 30 \). Maximum number of iterations = 1000

| Function | SMS | GSA | PSO | DE | TPSO |
|----------|-----|-----|-----|----|------|
| \( f_3 \) | AB 1756.862345 | 9750.440145 | 4329.650468 | 4963.600685 | 1893.673916 |
| MB 0.070624076 | 9838.388135 | 4233.282929 | 5000.245932 | 50.23617893 |
| SD 1949.048601 | 405.1365297 | 699.7276454 | 202.2888921 | 341.2367823 |
| \( f_6 \) | AB 10.95067665 | 15.18970458 | 129.4928091 | 196.1369499 | 1.234589423 |
| MB 0.07142491 | 3.9294268 | 129.4942809 | 196.1369499 | 1.234589423 |
| SD 14.38387472 | 4.508037915 | 27.8011038 | 9.65993059 | 7.764931264 |
| \( f_7 \) | AB 0.000299553 | 0.000575111 | 0.19630233 | 0.98547042 | 0.00234862 |
| MB 8.67349E-05 | 0 | 0.011090373 | 0.991214493 | 0.000482084 |
| SD 0.000623992 | 0.0021752 | 0.702516846 | 0.031985616 | 0.000196428 |
| \( f_8 \) | AB 1.35139E-05 | 2.792846799 | 1450.666769 | 304.6986718 | 1.753493426 |
| MB 7.14593E-06 | 2.723230534 | 0.675050254 | 51.8661185 | 1.002364819 |
| SD 2.0728E-05 | 4.508037915 | 170.798785 | 554.2231579 | 0.856294537 |
| \( f_9 \) | AB 0.0002080591 | 14.49783478 | 136.6888694 | 67251.29956 | 5.284029512 |
| MB 0.000623992 | 0.0021752 | 0.702516846 | 0.031985616 | 0.000196428 |
| SD 0.000623992 | 0.0021752 | 0.702516846 | 0.031985616 | 0.000196428 |
| \( f_{10} \) | AB 0.003412411 | 40.59204902 | 365.7806149 | 822.087914 | 9.63693364 |
| MB 0.003150999 | 18.02351657 | 7360.920758 | 6318.57249 | 1.023483601 |
| SD 0.003150999 | 18.02351657 | 7360.920758 | 6318.57249 | 1.023483601 |
| \( f_{11} \) | AB 0.199873346 | 1.121397135 | 0.857971914 | 3.70346788 | 0.452733836 |
| MB 0.199873346 | 1.121397135 | 0.857971914 | 3.70346788 | 0.452733836 |
| SD 0.073029674 | 0.271747312 | 1.736399225 | 0.278860779 | 0.247510642 |

Multimodal test functions Multimodal functions represent a good optimization challenge as they possess many local minima (Table 9). In the case of multimodal functions, final results are very important since they reflect the algorithm’s ability to escape from poor local optima and are able to locate a near-global optimum. Experiments using \( f_5 \) to \( f_{11} \) are quite relevant as the number of local minima for such functions increases exponentially as their dimensions increase. The dimension of such functions is set to 30. The results are averaged over 30 runs, reporting the performance index for each function in Table 3 as follows: the Average Best-so-far (AB) solution, the Median Best-so-far (MB) and the significance level) which is strong evidence against the null hypothesis. Therefore, such evidence indicates that SMS results are statistically significant and that it has not occurred by coincidence (i.e. due to common noise contained in the process).
An optimization algorithm inspired by the States of Matter that improves the balance

Fig. 5 Convergence rate comparison of PSO, GSA, DE, SMS and TPSO for minimization of (a) \( f_5 \) and (b) \( f_{11} \) considering \( n = 30 \)

Table 4 \( p \)-values produced by Wilcoxon's test comparing SMS vs. GSA, SMS vs. PSO, SMS vs. DE and SMS vs. TPSO over the “average best-so-far” (AB) values from Table 3

|        | SMS vs GSA | SMS vs PSO | SMS vs DE | SMS vs TPSO |
|--------|------------|------------|-----------|-------------|
| \( f_5 \) | 0.087      | 8.38 \times 10^{-4} | 4.61 \times 10^{-4} | 0.058       |
| \( f_6 \) | 0.062      | 1.92 \times 10^{-9} | 9.97 \times 10^{-8} | 0.012       |
| \( f_7 \) | 0.055      | 4.21 \times 10^{-5} | 3.34 \times 10^{-4} | 0.061       |
| \( f_8 \) | 7.74 \times 10^{-9} | 3.68 \times 10^{-7} | 8.12 \times 10^{-5} | 1.07 \times 10^{-5} |
| \( f_9 \) | 1.12 \times 10^{-8} | 8.80 \times 10^{-9} | 4.02 \times 10^{-8} | 9.21 \times 10^{-5} |
| \( f_{10} \) | 4.72 \times 10^{-9} | 3.92 \times 10^{-5} | 2.20 \times 10^{-4} | 7.41 \times 10^{-5} |
| \( f_{11} \) | 4.72 \times 10^{-9} | 3.92 \times 10^{-5} | 2.20 \times 10^{-4} | 4.05 \times 10^{-5} |

In the case of functions \( f_8, f_9, f_{10} \) and \( f_{11} \), SMS yields much better solutions than other methods. However, for functions \( f_5, f_6 \) and \( f_7 \), SMS produces similar results to GSA and TPSO. The Wilcoxon rank test results, which are presented in Table 4, demonstrate that SMS performed better than GSA, PSO, DE and TPSO for \( f_5, f_6 \) and \( f_7 \). The progress of the “average best-so-far” solution over 30 runs for functions \( f_5 \) and \( f_{11} \) is shown by Fig. 5.

Multimodal test functions with fixed dimensions In the following experiments, the SMS algorithm is compared to GSA, PSO, DE and TPSO over a set of multidimensional functions with fixed dimensions, which are widely used in the meta-heuristic literature. The functions used for the experiments are \( f_{12}, f_{13} \) and \( f_{14} \) which are presented in Ta-
Table 5 Minimization results of benchmark functions in Table 10 with \( n = 30 \). Maximum number of iterations = 500.

|        | SMS     | GSA      | PSO      | DE       | TPSO     |
|--------|---------|----------|----------|----------|----------|
| \( f_{12} \) | 0.004361206 | 0.051274735 | 0.020521847 | 0.006247895 | 0.008147895 |
| MB     | 0.004419241 | 0.051059414 | 0.020803912 | 0.004361206 | 0.003454528 |
| SD     | 2.40793E-15 | 0.016617355 | 0.021677285 | 8.7338E-15  | 6.37516E-15  |
| \( f_{13} \) | \(-3.862782148\) | \(-3.207627571\) | \(-3.122812884\) | \(-3.200286885\) | \(-3.311538343\) |
| MB     | \(-3.862782148\) | \(-3.222983851\) | \(-3.19877457\) | \(-3.200286885\) | \(-3.615938695\) |
| SD     | 2.40793E-15 | 0.016617355 | 0.021677285 | 8.7338E-15  | 6.37516E-15  |

Table 6 Minimization results of benchmark functions in Table 11 with \( n = 30 \). Maximum number of iterations = 1000.

|        | SMS     | GSA      | PSO      | DE       | TPSO     |
|--------|---------|----------|----------|----------|----------|
| \( f_{15} \) | \(-25.91760733\) | 57.15411412 | 134.3191481 | 183.6659439 | \(-18.63859195\) |
| MB     | \(-29.92042882\) | 57.38647154 | 133.1673936 | 186.723035 | \(-21.73646913\) |
| SD     | 2.38590437 | 14.20175945 | 68.4419497 | 38.0678428 | 12.5459285 |
| \( f_{16} \) | \(-57.89738314\) | \(-57.89605386\) | \(-40.5714691\) | \(-52.3502685\) | \(-52.564574931\) |
| MB     | \(-57.89738314\) | \(-57.89605386\) | \(-40.5714691\) | \(-52.3502685\) | \(-52.564574931\) |
| SD     | 0.00077726 | 0.000841082 | 4.812411459 | 1.769678878 | 1.346493542 |
| \( f_{17} \) | \(-449.9936552\) | 2015.050384 | 18201.78451 | 186.6285041 | 188.4369463 |
| MB     | \(-449.9936552\) | 2015.050384 | 18201.78451 | 186.6285041 | 188.4369463 |
| SD     | 0.000573064 | 1389.619208 | 6325.37951 | 2.880379023 | 1.453893855 |
| \( f_{18} \) | \(-1213.421542\) | 22038.7467 | 30055.82961 | 4355.34835 | 1452.346384 |
| MB     | \(-181.002827\) | 21908.86945 | 26882.92621 | 4228.56562 | 1401.749367 |
| SD     | 4050.267293 | 1770.050492 | 18048.5578 | 705.414378 | 532.364431 |
| \( f_{19} \) | \(-26975.80614\) | 23440.2683 | 30055.82961 | 4355.34835 | 1452.346384 |
| MB     | \(-2406.19301\) | 6517.39992 | 44733.97226 | 6048.33588 | 2863.49302 |
| SD     | 10128.06919 | 1389.619208 | 6325.37951 | 2.880379023 | 1.453893855 |
| \( f_{20} \) | \(-5716.886785\) | 6671.65533 | 44221.12187 | 58821.82993 | 29453.32822 |
| MB     | \(-5716.886785\) | 6671.65533 | 44221.12187 | 58821.82993 | 29453.32822 |
| SD     | 2406.19301 | 6517.39992 | 44733.97226 | 6048.33588 | 2863.49302 |
| \( f_{21} \) | \(-965.8899213\) | 181742714.4 | 7385919478 | 284396.8728 | 1051.438859 |
| MB     | \(-965.8899213\) | 181742714.4 | 7385919478 | 284396.8728 | 1051.438859 |
| SD     | 751.3821374 | 79542617.71 | 579950322 | 66484.87261 | 894.4384589 |
| \( f_{22} \) | \(-18617.61336\) | 30808.74384 | 444370.5566 | 429178.9416 | 2065.323956 |
| MB     | \(-18617.61336\) | 30808.74384 | 444370.5566 | 429178.9416 | 2065.323956 |
| SD     | 10932.4606 | 28009.57647 | 425696.8169 | 418480.2092 | 1943.343851 |
| \( f_{23} \) | \(-910.002925\) | 997.4123375 | 1026.55016 | 917.4176502 | 1017.3484548 |
| MB     | \(-910.002925\) | 997.4123375 | 1026.55016 | 917.4176502 | 1017.3484548 |
| SD     | 0.004747964 | 19.08754967 | 57.01221298 | 0.456448016 | 45.34396836 |

© Springer
An optimization algorithm inspired by the States of Matter that improves the balance

Fig. 7 Convergence rate comparison of PSO, GSA, DE, SMS and TPSO for minimization of (a) $f_{17}$ and (b) $f_{24}$

Table 7 $p$-values produced by Wilcoxon’s test that compare SMS vs. GSA, SMS vs. PSO, SMS vs. DE and SMS vs. TPSO, for the “average best-so-far” (AB) values from Table 6

| Function | SMS vs GSA | SMS vs PSO | SMS vs DE | SMS vs TPSO |
|----------|------------|------------|-----------|-------------|
| $f_{15}$ | 1.7344E-06 | 1.7344E-06 | 1.7344E-06 | 5.2334E-05 |
| $f_{16}$ | 9.7110E-05 | 1.7344E-06 | 1.7344E-06 | 3.1181E-05 |
| $f_{17}$ | 1.12654E-05 | 1.7344E-06 | 1.7344E-06 | 6.2292E-05 |
| $f_{18}$ | 1.7344E-06 | 1.7344E-06 | 1.7344E-06 | 1.8938E-05 |
| $f_{19}$ | 1.92092E-06 | 1.7344E-06 | 1.7344E-06 | 9.2757E-05 |
| $f_{20}$ | 1.7344E-06 | 9.7110E-05 | 2.1264E-06 | 8.3559E-05 |
| $f_{21}$ | 1.7344E-06 | 1.7344E-06 | 1.7344E-06 | 7.6302E-05 |
| $f_{22}$ | 1.7344E-06 | 1.7344E-06 | 1.7344E-06 | 6.4821E-05 |
| $f_{23}$ | 0.014795424 | 1.7344E-06 | 1.7344E-06 | 8.8351E-05 |
| $f_{24}$ | 1.7344E-06 | 1.7344E-06 | 1.7344E-06 | 9.9453E-05 |

5 Conclusions

In this paper, a novel nature-inspired algorithm called as the States of Matter Search (SMS) has been introduced. The SMS algorithm is based on the simulation of the State of Matter phenomenon. In SMS, individuals emulate molecules which interact to each other by using evolutionary operations that are based on physical principles of the thermal-energy motion mechanism. The algorithm is devised by considering each state of matter at one different exploration–exploitation ratio. The evolutionary process is divided into three phases which emulate the three states of matter: gas, liquid and solid. At each state, molecules (individuals) exhibit different movement capacities. Beginning from the gas state (pure exploration), the algorithm modifies the intensities of exploration and exploitation until the solid state (pure exploitation) is reached. As a result, the approach can substantially improve the balance between exploration–exploitation, yet preserving the good search capabilities of an EA approach.

SMS has been experimentally tested considering a suite of 24 benchmark functions. The performance of SMS has also been compared to the following evolutionary algorithms: the Particle Swarm Optimization method (PSO) [17], the Gravitational Search Algorithm (GSA) [16], the Differential Evolution (DE) algorithm [13] and the PSO algorithm with a territorial diversity-preserving scheme (TPSO) [39]. Results have confirmed a high performance of the proposed
method in terms of the solution quality for solving most of the benchmark functions.

The SMS’s remarkable performance is associated with two different reasons: (i) the defined operators allow a better particle distribution in the search space, increasing the algorithm’s ability to find the global optima; and (ii) the division of the evolution process at different stages, provides different rates between exploration and exploitation during the evolution process. At the beginning, pure exploitation is favored at the gas state, then a mild transition between exploration and exploitation features during the liquid state. Finally, pure exploitation is performed during the solid state.

Acknowledgements The proposed algorithm is part of the optimization system used by a biped robot supported under the grant CONACYT CB 181053.

Appendix A: List of benchmark functions

Table 8 Unimodal test functions

| Test function | $S$          | $f_{\text{opt}}$ | $n$ |
|---------------|--------------|------------------|-----|
| $f_1(x) = \sum_{i=1}^{n} x_i^2$ | $[-100, 100]^n$ | 0 | 30 |
| $f_2(x) = \text{max}(|x_i|, 1 \leq i \leq n)$ | $[-100, 100]^n$ | 0 | 30 |
| $f_3(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$ | $[-30, 30]^n$ | 0 | 30 |
| $f_4(x) = \sum_{i=1}^{n} x_i^4 + \text{rand}(0, 1)$ | $[-1.28, 1.28]^n$ | 0 | 30 |

Table 9 Multimodal test functions

| Test function | $S$          | $f_{\text{opt}}$ | $n$ |
|---------------|--------------|------------------|-----|
| $f_5(x) = 418.9829n + \sum_{i=1}^{n} (-x_i \sin(\sqrt{|x_i|}))$ | $[-500, 500]^n$ | 0 | 30 |
| $f_6(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10)$ | $[-5.12, 5.12]^n$ | 0 | 30 |
| $f_7(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left(\frac{\pi x_i}{\sqrt{i}}\right) + 1$ | $[-600, 600]^n$ | 0 | 30 |
| $f_8(x) = \frac{\pi}{n} \{10 \sin(\pi y_1) + \sum_{i=1}^{n-1} \left(y_i - 1\right)^2 + 10 \sin^2(\pi y_i + 1) + (y_n - 1)^2 + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\}$ | $[-50, 50]^n$ | 0 | 30 |
| $y_i = 1 + \frac{(x_i + 1)}{4}, u(x_i, a, k, m) = \left\{\begin{array}{ll} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < a \end{array}\right.$ | $[-50, 50]^n$ | 0 | 30 |
| $f_9(x) = 0.1 \sin(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]$ | $[-50, 50]^n$ | 0 | 30 |
| $f_{10}(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5 x_i)^2 + (\sum_{i=1}^{n} 0.5 x_i)^4$ | $[-10, 10]^n$ | 0 | 30 |
| $f_{11}(x) = 1 - \cos(2\pi \|x\|) + 0.1 \|x\|$ where $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$ | $[-100, 100]^n$ | 0 | 30 |

Table 10 Multimodal test functions with fixed dimensions

| Test function | $S$ | $f_{\text{opt}}$ | $n$ |
|---------------|-----|------------------|-----|
| $f_{12}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_i(b_i^2 + b_{i+1}x_i)}{b_i + b_{i+1}}\right]^2$ | $[-5, 5]^n$ | 0.00030 | 4 |
| $a = [0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.456, 0.0342, 0.0323, 0.0235, 0.0246]$ | | | |
| $b = [0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16]$ | | | |
| $f_{13}(x) = \sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} A_{ij} (x_j - P_{ij})^2)$, $A = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$ | $[0, 1]^n$ | -3.32 | 6 |
| $c = [1, 1.2, 3, 3.2], P = \begin{bmatrix} 0.131 & 0.169 & 0.556 & 0.012 & 0.828 & 0.588 \\ 0.232 & 0.413 & 0.830 & 0.373 & 0.100 & 0.999 \\ 0.23 & 0.141 & 0.352 & 0.288 & 0.304 & 0.665 \\ 0.404 & 0.882 & 0.873 & 0.574 & 0.109 & 0.038 \end{bmatrix}$ | | | |
| $f_{14}(x) = (1.5 - x_1(x_1 - 2))^2 + (2.5 - x_1(x_1 - 1))^2 + (2.5 - x_1(x_1 - 2))^2$ | $[-4.5, 4.5]^n$ | 0 | 2 |
An optimization algorithm inspired by the States of Matter that improves the balance

Table 11  Set of representative GECCO functions

| Test function | $S$ | $n$ | GECCO classification |
|---------------|-----|-----|----------------------|
| $f_{15}(x) = 10^6 \cdot z_i^2 + \sum_{i=2}^{n} z_i + f_{opt}$ | $[-5, 5]^n$ | 30 | GECCO2010 |
| $z = T_{opt}(x - x^{opt})$ | | | |
| $T_{opt} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, for any positive integer $n$, it maps element-wise | | | |
| $a = T_{opt}(h), a = [a_1, a_2, \ldots, a_n], h = (h_1, h_2, \ldots, h_n)$ | | | |
| $a_i = \text{sign}(h_i) \exp(K + 0.049(\sin(c_1 K) + \sin(c_2 K)))$, | | | |
| where | | | |
| $K = \begin{cases} \log(h_i) & \text{if } h_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$ | | | |
| $c_1 = \begin{cases} 10 & \text{if } h_1 > 0 \\ 5.5 & \text{otherwise} \end{cases}$ and $c_2 = \begin{cases} 7.9 & \text{if } h_1 > 0 \\ 3.1 & \text{otherwise} \end{cases}$ | | | |
| $f_{16}(x) = \sqrt{\sum_{i=1}^{n} (z_i)^{2f_{pen}(x)} + f_{opt}}$ | $[-5, 5]^n$ | 30 | GECCO2010 |
| $z = x - x^{opt}$ | | | |
| $f_{17}(x) = -\frac{1}{K} \sum_{i=1}^{n} z_i \sin(\sqrt{|z_i|}) + 4.189828874339 + 100 f_{pen}(x) + f_{opt}$ | $[-5, 5]^n$ | 30 | GECCO2010 |
| $x = 2 \times 1^+ \otimes x$ | | | |
| $\hat{z}_1 = \hat{z}_i, \hat{z}_{i+1} = \hat{z}_i + 0.25(\hat{z}_i - x_i^{opt})$ for $i = 1, \ldots, n - 1$ | | | |
| $z = 100(A^{10}(z - x^{opt}) + x^{opt})$ | | | |
| $f_{pen} : \mathbb{R}^n \rightarrow \mathbb{R}$ | | | |
| $a = f_{pen}(h), h = [h_1, h_2, \ldots, h_n]$ | | | |
| $a = 100 \sum_{i=1}^{n} \max(0, |h_i| - 5)^2$ | | | |
| $1^+$ is an $n$-dimensional vector with elements of $-1$ or $1$ computed considering equal probability | | | |
| $f_{18}(x) = \sum_{i=1}^{n} z_i^2 - 450$ | $[-100, 100]^n$ | 30 | GECCO2005 |
| $z = x - x^{opt}$ | | | |
| $f_{19}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} z_j)^2 - 450$ | $[-100, 100]^n$ | 30 | GECCO2005 |
| $z = x - x^{opt}$ | | | |
| $f_{20}(x) = (\sum_{i=1}^{n} (\sum_{j=1}^{i} z_j)^2 + 10.049(\sin(0(0, 1))) - 450$ | $[-100, 100]^n$ | 30 | GECCO2005 |
| $z = x - x^{opt}$ | | | |
| $f_{21}(x) = \max(|Ax - b|| - 310$ | $[-100, 100]^n$ | 30 | GECCO2005 |
| $A$ is an $n \times n$ matrix, $a_{i,j}$ are integer random numbers in the range $[-500, 500], \det(A) \neq 0$ | | | |
| $b_i = A_i \cdot o$ | | | |
| $A_i$ is the $i$-th row of $A$ whereas $o$ is an $n \times 1$ vector whose elements are random numbers in the range $[-100, 100]$ | | | |
| $f_{22}(x) = \sum_{i=1}^{n} (100(z_i^2 - z_i - 1)^2 + (z_i - 1)^2) + 390$ | $[-100, 100]^n$ | 30 | GECCO2005 |
| $z = x - x^{opt}$ | | | |
| $f_{23}(x) = \sum_{i=1}^{n} (A_i - B_i(x))^2 - 460$ | $[-\pi, \pi]^n$ | 30 | GECCO2005 |
| $A_i = \sum_{j=1}^{n} a_{i,j} \sin x_j + h_{ij} \cos x_j, B_i(x) = \sum_{j=1}^{n} a_{i,j} \sin x_j + h_{ij} \cos x_j$ | | | |
| For $i = 1, \ldots, n, a_{i,j}$ and $h_{ij}$ are integer random numbers in the range $[-100, 100]$ | | | |
| $a = [a_1, a_2, \ldots, a_n], a_j$ are random numbers in the range $[-\pi, \pi]$ | | | |
Table 11 (Continued)

| Test function | $S$ | $n$ | GECCO classification |
|---------------|-----|-----|----------------------|
| $f_{24}(x) = \sum_{i=1}^{10} \delta_i (x - x_i^0)^2 / \lambda_i$ | $[-5, 5]^n$ | 30 | GECCO2005 |
| $F_{1-2}(x) = \text{Ackley's function}$ | | | |
| $F_i(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20$ | | | |
| $F_{3-4}(x) = \text{Rastrigin's function}$ | | | |
| $F_i(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10)$ | | | |
| $F_{3-5}(x) = \text{Sphere function}$ | | | |
| $F_i(x) = \sum_{i=1}^{n} x_i^2$ | | | |
| $F_{3-6}(x) = \text{Weierstrass function}$ | | | |
| $F_i(x) = \sum_{i=1}^{n} [a_i \cos(2\pi b_i (x_i + 0.5))] - n \sum_{i=1}^{n} [a_i \cos(2\pi b_i (x_i \cdot 0.5))]$ | | | |
| $F_{9-10}(x) = \text{Griewank's function}$ | | | |
| $F_i(x) = \sum_{i=1}^{n} \frac{x_i^2}{\sqrt{n}} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$ | | | |
| $\delta = \left\{ \frac{10}{32}, 10, \frac{5}{12}, 2, 1, \frac{10}{100}, 5, \frac{10}{100} \right\}$ | | | |

The $x_i^{0}$ and $f_{opt}$ values have been set to default values which have been obtained from the Matlab©implementation for GECCO competitions, as it is provided in [51].

References

1. Han M-F, Liao S-H, Chang J-Y, Lin C-T (2012) Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems. Appl Intell. doi:10.1007/s10489-012-0393-5
2. Pardalos Panos M, Romeijn Edwin H, Tuy H (2000) Recent developments and trends in global optimization. J Comput Appl Math 124:209–228
3. Floudas C, Akrotirianakis I, Caratzoulas S, Meyer C, Karlrrath J (2005) Global optimization in the 21st century: advances and challenges. Comput Chem Eng 29(6):1185–1202
4. Ying J, Ke-Cun Z, Shao-Jian Q (2007) A deterministic global optimization algorithm. Appl Math Comput 185(1):382–387
5. Georgieva A, Jordanov I (2009) Global optimization based on novel heuristics, low-discrepancy sequences and genetic algorithms. Eur J Oper Res 196:413–422
6. Lera D, Sergeyev Ya (2010) Lipschitz and Hölder global optimization using space-filling curves. Appl Numer Math 60(1–2):115–129
7. Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley, Chichester
8. De Jong K (1975) Analysis of the behavior of a class of genetic adaptive systems. Ph.D. Thesis, University of Michigan, Ann Arbor
9. Koza JR (1990) Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Rep. No. STAN-CS-90-1314, Stanford University, CA
10. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
11. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison Wesley, Boston
12. de Castro LN, Von Zuben FJ (1999) Artificial immune systems: Part I—basic theory and applications. Technical report TR-DCA 01/99
13. Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimisation over continuous spaces. Tech. Rep. TR-95–012, ICSI, Berkeley, CA
14. Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680
15. Ilker B, Birbil S, Shu-Cherng F (2003) An electromagnetism-like mechanism for global optimization. J Glob Optim 25:263–282
16. Rashedia E, Nezamabadi-pour H, Saryazdi S (2011) Filter modeling using gravitational search algorithm. Eng Appl Artif Intell 24(1):117–122
17. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the 1995 IEEE international conference on neural networks, December 1995, vol 4, pp 1942–1948
18. Dorigo M, Maniezzo V, Colomi A (1991) Positive feedback as a search strategy. Technical Report No. 91-016, Politecnico di Milano
19. Tan KC, Chiam SC, Mamun AA, Goh CK (2009) Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. Eur J Oper Res 197:701–713
20. Chen G, Low CP, Yang Z (2009) Preserving and exploiting genetic diversity in evolutionary programming algorithms. IEEE Trans Evol Comput 13(3):661–673
21. Liu S-H, Mernik M, Bryant B (2009) To explore or to exploit: an entropy-driven approach for evolutionary algorithms. Int J Knowl-Based Intell Eng Syst 13(3):185–206
22. Alba E, Dorronsoro B (2005) The exploration/exploitation trade-off in dynamic cellular genetic algorithms. IEEE Trans Evol Comput 9(3):126–142
23. Fister I, Mernik M, Filipiˇc B (2010) A hybrid self-adaptive evolutionary algorithm for marker optimization in the clothing industry. Appl Soft Comput 10(2):409–422
24. Gong W, Cai Z, Jiang L (2008) Enhancing the performance of differential evolution using orthogonal design method. Appl Math Comput 206(1):56–69
25. Joan-Arinyo R, Luzon MV, Yeguas E (1991) Parameter tuning of pbil and chc evolutionary algorithms applied to solve the root identification problem. Appl Soft Comput 11(1):754–767
26. Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696
27. Sadegh M, Reza M, Palhang M (2012) LADPSO: using fuzzy logic to conduct PSO algorithm. Appl Intell 37(2):290–304
An optimization algorithm inspired by the States of Matter that improves the balance

28. Yadav P, Kumar R, Panda SK, Chang CS (2012) An intelligent tuned harmony search algorithm for optimization. Inf Sci 196(1):47–72

29. Khajehzadeh M, Rainha Taha M, El-Shafie A, Eslemi M (2012) A modified gravitational search algorithm for slope stability analysis. Eng Appl Artif Intell 25(8):1589–1597

30. Kounousis V, Katsaras CP (2006) A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 10(1):19–28

31. Han M-F, Liao S-H, Chang J-Y, Lin C-T (2012) Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems. Appl Intell. doi:10.1007/s10489-012-0393-5

32. Brest J, Maučec, MS (2008) Population size reduction for the differential evolution algorithm. Appl Intell 29(3):228–247

33. Li Y, Zeng X (2010) Multi-population co-genetic algorithm with double chain-like agents structure for parallel global numerical optimization. Appl Intell 32(3):292–310

34. Paenke I, Jin Y, Branke J (2009) Balancing population- and individual-level adaptation in changing environments. Adapt Behav 17(2):153–174

35. Cafaro L, Merelo JJ (2011) Diversity through multiculturality: assessing migrant choice policies in an island model. IEEE Trans Evol Comput 15(4):456–468

36. Gao H, Xu W (2011) Particle swarm algorithm with hybrid mutation strategy. Appl Soft Comput 11(8):5129–5142

37. Jia D, Zheng G, Khan MK (2011) An effective memetic differential evolution algorithm based on chaotic local search. Inf Sci 181(15):3175–3187

38. Lozano M, Herrera F, Cano JR (2008) Replacement strategies to preserve useful diversity in steady-state genetic algorithms. Inf Sci 178(3):4421–4433

39. Ostadmohammadi B, Mirzabeygi P, Panahi M (2013) An improved PSO algorithm with a territorial diversity-preserving scheme and enhanced exploration–exploitation balance. Swarm Evol Comput 11:1–15

40. Yang G-P, Liu S-Y, Zhang J-K, Feng Q-X (2012) Convergence strategy. Appl Soft Comput 11(8):5129–5142

41. Hasanzadeh M, Meybodi MR, Ebadzadeh MM (2012) Adaptive cooperative particle swarm optimizer. Appl Intell. doi:10.1007/s10489-012-0420-6

42. Aribarg T, Supratid S, Lursinsap C (2012) Optimizing the modified fuzzy anti-miner for efficient medical diagnosis. Appl Intell 37(3):357–376

43. Fernandes CM, Laredo JLJ, Rosa AC, Merelo JJ (2012) The sandpile mutation Genetic Algorithm: an investigation on the working mechanisms of a diversity-oriented and self-organized mutation operator for non-stationary functions. Appl Intell. doi:10.1007/s10489-012-0413-5

44. Gwak J, Sim KM (2013) A novel method for coevolving PS-optimizing negotiation strategies using improved diversity controlling EDAs. Appl Intell 38(3):384–417

45. Cheshmehgaz HR, Ishak Desa M, Wibowo A (2013) Effective local evolutionary searches distributed on an island model solving bi-objective optimization problems. Appl Intell 38(3):331–356

46. Cuevas E, Gonzalez M (2012) Multi-circle detection on images inspired by collective animal behaviour. Appl Intell. doi:10.1007/s10489-012-0396-2

47. Adra SF, Fleming PJ (2011) Diversity management in evolutionary many-objective optimization. IEEE Trans Evol Comput 15(2):183–195

48. Črepinšek M, Liu SH, Mernik M (2011) Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput Surv 1(1):1–33

49. Cerutti MG, Rubin, SH (2007) Infodynamics: Analogical analysis of states of matter and information. Inf Sci 177:969–987

50. Chowdhury D, Stauffer F (2000) Principles of equilibrium statistical mechanics. Wiley-VCH, New York

51. Betts DS, Turner RE (1992) Introductory statistical mechanics, 1st edn. Addison Wesley, Reading

52. Cengel YA, Boles MA (2005) Thermodynamics: an engineering approach, 5th edn. McGraw-Hill, New York

53. Bueche F, Hecht E (2011) Schaum’s outline of college physics, 11th edn. McGraw-Hill, New York

54. Piotrowski AP, Napiorkowski JJ, Kiczko A (2012) Differential evolution algorithm with separated groups for multi-dimensional optimization problems. Eur J Oper Res 216(1):33–46

55. Cocco Mariani V, Justi Luvizotto LG, Alessandro Guerra F, dos Santos Coelho L (2011) A hybrid shuffled complex evolution approach based on differential evolution for unconstrained optimization. Appl Math Comput 217(12):5822–5829

56. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

57. More J, Garbow BS, Hillstrom KE (1981) Testing unconstrained optimization software. ACM Trans Math Softw 7(1):17–41

58. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203(2):598–607

59. Black-Box Optimization Benchmarking (BBOB) 2010, 2nd GECCO Workshop for Real-Parameter Optimization. http://coco.gforge.inria.fr/doku.php?id=bbob-2010

60. Abdel-Rahman Hedar, Ali AF (2012) Tabu search with multi-level neighborhood structures for high dimensional problems. Appl Intell 37(2):189–206

61. Vafashoar R, Meybodi MR, Momeni Azandaryani AH (2012) CLA-DE: a hybrid model based on cellular learning automata for numerical optimization. Appl Intell 36(3):735–748

62. Garcia S, Molina D, Lozano M, Herrera F (2008) A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 special session on real parameter optimization. J Heuristics. doi:10.1007/s10732-008-9080-4

63. Shilane D, Martikainen J, Dudoit S, Ovaska S (2008) A general framework for statistical performance comparison of evolutionary computation algorithms. Inf Sci 178:2870–2879

Erik Cuevas received the B.S. degree with distinction in Electronics and Communications Engineering from the University of Guadalajara, Mexico in 1995, the M.Sc. degree in Industrial Electronics from ITESO, Mexico in 2000, and the Ph.D. degree from Freie Universität Berlin, Germany in 2005. From 2001 he was awarded a scholarship from the German Service for Academic Interchange (DAAD) as full-time researcher. Since 2007 he has been with University of Guadalajara, where he is currently a fulltime Professor in the Department of Electronics. From 2008, he is a member of the Mexican National Research System (SNI). His current research interests include computer vision and artificial intelligence.
Alonso Echavarría received the B.S. degree in Mechanical Engineering from the Instituto Tecnológico de Culiacan, Mexico in 2009. He is currently finishing his master degree in Electronics and Computer Science from Universidad de Guadalajara, Mexico. His current research interests include artificial intelligence, computer vision and evolutionary algorithms.