Genetic Algorithms

Shichang Li and Dengfeng Li

Genetic algorithms (GAs), developed by John Holland and his colleagues in 1975 [1], has been applied to many fields in search and optimization that belongs to the larger class of evolutionary algorithms (EAs). This approach is a metaheuristic inspired by the mechanisms of evolution, such as selection, crossover, and mutation. Drawing on the theory of biological evolution, the goals of their research have been solved into a biological evolution process by genetic algorithm, which generates next-generation solutions through operations such as duplication, crossover, and mutation. In this way, after *n* generations of evolution, it is very likely to evolve function with high fitness values. In fact, there are several methods available for solving the global optimization problems in materials science and related fields. Some of the well-known GAs include artificial bee colony, particle swarm optimization, differential evolution, etc.

1 Artificial Bee Colony Algorithms

The artificial bee colony algorithm was proposed by Karaboga (2005) [2] to simulate the intelligent behavior of a honeybee swarm. It consists of three main parts: food sources, employed foragers, and unemployed foragers. In order to better explain the basic principles of artificial bee colony algorithm, the three basic elements are introduced in detail as follows:

S. Li · D. Li (✉)
School of Science, Chongqing University of Posts and Telecommunications, Chongqing, China
e-mail: lisc@cqupt.edu.cn; lidf@cqupt.edu.cn

Y. Cheng et al. (eds.), *Artificial Intelligence for Materials Science*, Springer Series in Materials Science 312, https://doi.org/10.1007/978-3-030-68310-8_5

1. Food sources

   In the algorithm, food sources are mainly to represent different solutions, which depends on many factors such as its proximity to the nest, its richness or concentration of its energy, and the ease of extracting this energy. The “profitability” was used to evaluate the excellence of a food source.
2. Employed foragers

   Also known as a leader, it corresponds to the food source collected. The leading bee stores information about a food source (distance relative to the hive, direction, food source abundance, etc.) and shares this information with other bees with a certain probability.
3. Unemployed foragers

   The main task of unemployed foragers is to find and exploit food sources. There are two types of unemployed foragers: scouts and onlookers. Scouts begin to search for new food sources near the hive; Onlookers is waiting in the hive and to find the nectar source by sharing relevant information with the leader (employed foragers). In general, the average number of scouts is about 5–10% of the colony.

In the formation of searching the food sources, the exchange of information between bees is the most important part. The dancing area is the most important information exchange place in the hive. The dancing of the bees is called a waggle dance. The information of food source is shared with other bees in the dancing area, leading to express the profitability of food source on the dance floor. Therefore, the onlookers can observe a large number of food sources and choose more profitable sources due to some internal motivation or possible external clue. The probability of bees being recruited is proportional to the profitability of the food source.

1.1 Main Steps of the Artificial Bee Colony Algorithm

Based on the above explanation of initializing the algorithm population, employed bee phase, probabilistic selection scheme, onlooker bee phase, and scout bee phase, the pseudo-code of the artificial bee colony algorithm is given below [3]:

Algorithm 1 Artificial Bee Colony Algorithm

```
01: Initialize the population of solutions $x_{i,j} = 1, 2 \cdots SN, j = 1, 2 \cdots n$,
$trial_i = 0$, $trial_i = 0$ is the non-improvement number of the solution $X_i$, used for
abandonment (food source $X_i$ depending on its probability value $p_i$)
02: Evaluate the population
03: cycle = 1
04: **repeat**
        {– – Produce a new food source population for employed bees – –}
06:   **for**$i = 1$ to $SN$**do**
07:       Produce a new food source $V_i$ for the employed bee of the food source $X_i$ using
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j})$ and evaluate its quality
08:       Apply a greedy selection process between $V_i$ and , select the better one.
09:       If solution $X_i$ does not improve $trial_i = trial_i + 1$, otherwise $trial_i = 0$
10: **end for**
11:       Calculate the probability values $p_i$ for the solutions using fitness values.
            {– – Produce a new food source population for onlooker bees – –}
12:       $t = 0, i = 1$
13: **repeat**
14:        **if**   $random < \mathrm{p}_i$   **then**
15:         Produce a new food source for onlooker bee
16:         Apply a greedy selection process between $V_i$ and $X_i$, select the better one
17:         If solution $X_i$ does not improve $trial_i = trial_i + 1$, otherwise $trial_i = 0$
18:         $t = t + 1$
19: **endif**
20: **until** $(t = SN)$
      {– – Determine Scout – –}
21:   **if** max(trial$_i$) > limit **then**
22:       Replace $X_i$ with a new randomly produced solution
23: **end if**
24:   Memorize the best solution achieved so far
25:   cycle = cycle + 1
26: **until** (cycle=Maximum Cycle Number)
```

Algorithm 2 A Novel Initialization Approach

```
01: Set the maximum number of chaotic iteration $K \geq 300$, the population size SN, and the
individual counter $i = 1, j = 1$
                {– – chaotic systems – –}
03: **for** $i = 1$ to $SN$ **do**
04:   **for** $j = 1$ to $n$ **do**
05:     Randomly initialize variables $ch_{0,j} \in (0, 1)$, set iteration counter k = 0
06:     **for** $k = 1$ to K **do**
07:       $ch_{k+1,j} = \sin(\pi ch_{k,j})$
08:     **end for**
09:       $x_{ij} = x_{\min,j} + ch_{k,j}(x_{\max,j} - x_{\min,j})$
10:     **end for**
11: **end for**
              {– – Opposition-based learning method – –}
13: Set the individual counter $i = 1, j = 1$
14: **for** $i = 1$ to $SN$ **do**
15:   **for** $j = 1$ to $n$ **do**
16:   $ox_{i,j} = x_{\min,j} + x_{\max,j} - x_{i,j}$
17:   **end for**
18: **end for**
19: Selecting $SN$ fittest individuals from set the $\{X(SN)\} \cup OX(SN)$ as initial population
25:   cycle = cycle + 1
26: **until** (cycle=Maximum Cycle Number)
```

1.2 Applications of the Artificial Bee Colony in Functional Materials

Interestingly, artificial bee colony has been tailored successfully, to solve a wide variety of discrete and continuous optimization problems. The distribution of published research articles on artificial bee colony with respect to applications, hybridizations, and modifications is shown in Fig. 1 [4].

The artificial bee colony also has been mostly implemented in design of many materials. For example, variables in design of multilayer radar-absorbing material with various numbers of layers are optimally determined using artificial bee colony which is one of the latest natural-inspired algorithms [5]. The multilayer radar-absorbing material in terms of electrical and geometric variables is conceptually synthesized with the aid of artificial bee colony optimization algorithm. Five truss examples with fixed-geometry and up to 200 elements were studied to verify that the ABC algorithm is an effective optimization algorithm in the creation of an optimal design for truss structures [6].

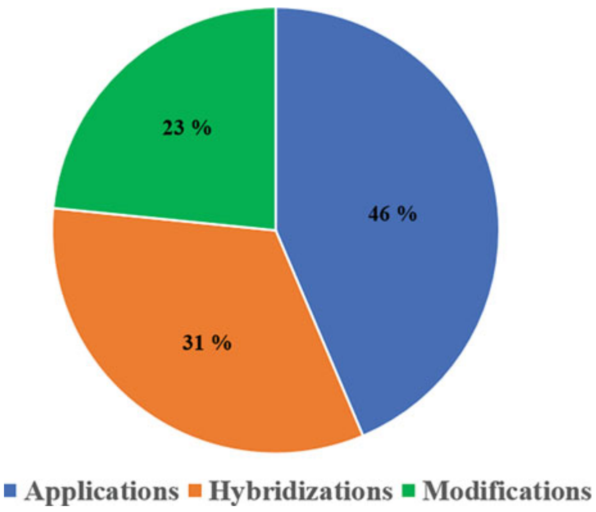


Fig. 1 The distribution of published research articles on artificial bee colony

1.3 Current Status of Research on Artificial Bee Colony Algorithms

In 2005, Karaboga proposed a more complete model of artificial bee colony algorithm and solved the function optimization problem with artificial bee colony algorithm, which achieved certain results. From the experimental results, the artificial bee colony algorithm is more excellent than the heuristic algorithm in terms of nonrestrictive numerical optimization.

More and more attempts have been made to improve the algorithm. Gao and Liu [7] introduced a new initialization approach and a novel search mechanism to improve the artificial bee colony algorithm. The results showed that the improved algorithm can outperform some conventional algorithms in accuracy, convergence speed, stability, and robustness. Zhu and Kwong [8] proposed an improved artificial bee colony algorithm, called Gbest-guided artificial bee colony algorithm. The results show that the Gbest-guided artificial bee colony algorithm possesses superior performance in most of the experiments, as compared to the conventional artificial bee colony algorithm. A modified artificial bee colony algorithm, developed by Gao and Liu [3], has been shown to be competitive to other population-based algorithms. The new optimization algorithm is based on that the bee searches only around the best solution of the previous iteration, leading to improving the exploitation.

2 Particle Swarm Optimization Algorithms

Particle swarm optimization (PSO) is a branch of evolutionary algorithms, which is inspired by the choreography of a bird flock that seems to be effective for optimizing a wide range of functions. In fact, it can be viewed as a distributed behavior algorithm that performs multidimensional search. PSO was first proposed by Kennedy and Eberhart in 1995 [9].

The principle of POS is similar to other evolutionary algorithms, and it also moves individuals in the group to a better place. Unlike other evolutionary algorithms, the particle swarm algorithm does not use any evolutionary operators, each individual member is regarded as a particle, which runs at a certain speed in space, and uses flying speed and position to adjust the entire algorithm space. Therefore, all the individuals can quickly find the global stable position and near-optimal geographical position. The PSO has been experimentally proven to be competitive and better than most of the algorithms on many optimization problems.

In the particle swarm algorithm, the position of each particle is updated using its velocity vector as depicted in Fig. 2 [10]. And the detail operation through Eq. (1) as following:

$$\begin{aligned} V_{i,j}(t+1) &= \omega V_{i,j}(t) + c_1 r_{1,j}\left[pbest_{i,j}(t) - X_{i,j}(t)\right] \\ &+ c_2 r_{2,j}\left[gbest_j(t) - X_{i,j}(t)\right] X_{i,j}(t+1) = X_{i,j}(t) + V_{i,j}(t+1) \end{aligned} \tag{1}$$

where $pbest_{i,j} = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$ and $gbest_j = (pbest_1, pbest_2, \ldots, pbest_D)$ are current location and population global location, respectively. X is a set of positions of i particles in a D-dimensional space. ω is inertia weight. c_1 and c_2 are self-confidence factor and swarm confidence factor. r_1 and r_2 are two separately generated random numbers. The algorithm flowchart of PSO is presented in Fig. 3.

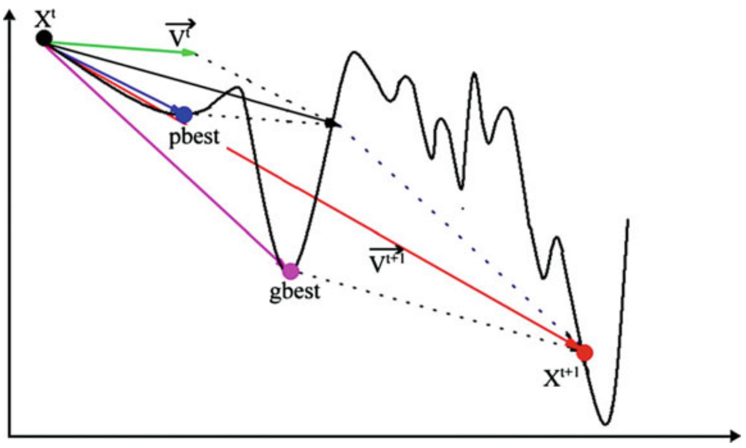
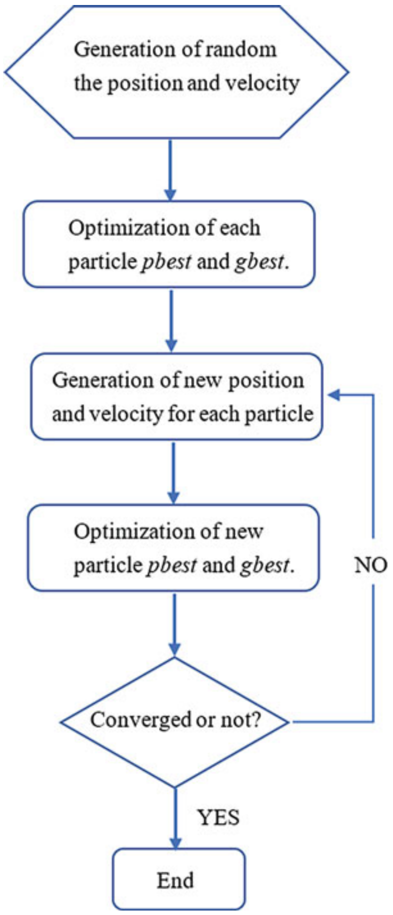


Fig. 2 Depiction of the velocity and position updates in PSO

Fig. 3 The flowchart of PSO


Generation of random the position and velocity
Optimization of each particle *pbest* and *gbest*.
Generation of new position and velocity for each particle
Optimization of new particle *pbest* and *gbest*.
NO
Converged or not?
YES
End

The steps of PSO evolution are as follows:

1. Generate random the position and velocity of all particles, and optimize each particle *pbest* and *gbest*.
2. Generate new position and velocity for each particle and compare them with the previous *pbest*. If the current position is better than previous *pbest*, then replace previous *pbest*. Otherwise, keep the current value.
3. For each particle, compare its position with the previous *gbest*. If the present position is better than *gbest*, then previous *gbest* is replaced. Otherwise, keep the current value.
4. Updating of the velocity and position of each particle, using the above equation.
5. If the convergence condition is not met, return to step (2). There are several criteria for convergence, such as free energy, cell sizes, and symmetry.
6. If the convergence condition is met, the present *gbest* value is taken as the optimal solution.

2.1 Applications of the Particle Swarm Optimization in Multifunctional Materials

Recently, PSO algorithm has been employed to various optimization problems for 0D nanoparticles or clusters, 2D layers and its atom adsorption, 2D surface reconstructions, and 3D crystals. And this model has been successfully used to inversely design multifunctional materials (e.g., superhard materials, electrides, optical materials) for many ternary and ternary compounds.

For example, PSO algorithm is used to predict the structures of many ternary Np-H compounds. The searching results of the energetically most favorable structures of NpHx ($x = 1$–10) found by us are plotted in Fig. 4 by PSO algorithm at ambient and high pressures. The complete pressure–composition phase diagram of Np–H compounds is shown in Fig. 5, with the stable structures identified using colors and space groups. The relative formation enthalpies of the energetically most favorable structures of NpHx found by us are plotted in Fig. 6 with cell sizes of 1–6 formula units, covering the pressure range up to 200 GPa. For ternary compounds, the PSO algorithm has been successfully used to predict the ambient pressure structures of $PuGaO_3$ and $CeGaO_3$, as shown in Fig. 7 [11].

2.2 Current State of the Particle Swarm Optimization

Since the PSO was proposed in 1995, the mathematical model of PSO is relatively simple and its application is surprisingly considerable, which has prompted a lot of researchers to study it. There have been a lot of research achievements in theoretical research of algorithm, the modification of model, and the fusion arithmetic.

1. Theoretical research

   Ozcan and Mohan [12] have analyzed the theory of particle swarm algorithm in 1999, the formula of the algorithm has been updated for the first time. Then Clerc and Kennedy analyzed the operation of particle swarm algorithm in multidimensional space [13]. Trelea have investigated the convergence of particle swarm algorithm by using discrete dynamic system and proposed some methods of parameter selection that are helpful for the overall stability and convergence of the algorithm. It has been well confirmed in the experimental data [14].
2. The modification of model

   In 1998, Shi and Eberhart published a paper about modified PSO algorithms at the International Conference on Swarm Intelligence. Inertia weight was introduced for the first time by this algorithm, with such improvements widely used in the academic community, gradually converted to the standard particle swarm optimization algorithm which is widely used at present. In order to improve this algorithm, Eberhart and Shi proposed to adjust with fuzzy system

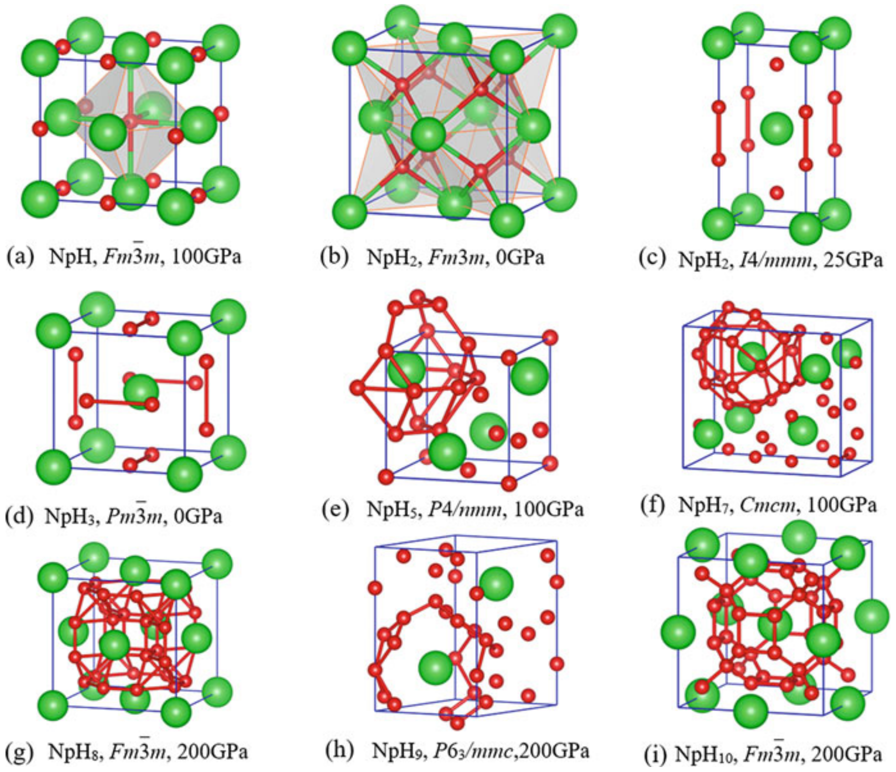

(a) NpH, $Fm\bar{3}m$, 100GPa
(b) NpH_2, $Fm3m$, 0GPa
(c) NpH_2, $I4/mmm$, 25GPa
(d) NpH_3, $Pm\bar{3}m$, 0GPa
(e) NpH_5, $P4/nmm$, 100GPa
(f) NpH_7, $Cmcm$, 100GPa
(g) NpH_8, $Fm\bar{3}m$, 200GPa
(h) NpH_9, $P6_3/mmc$,200GPa
(i) NpH_{10}, $Fm\bar{3}m$, 200GPa

Fig. 4 Crystal structures of the predicted stable neptunium hydrides compounds: (**a**) $Fm\bar{3}m$ structure of NpH at 100 GPa, (**b**) $Fm\bar{3}m$ structure of NpH_2 at ambient pressure, (**c**) $P6_3/mmc$ structure of NpH_2 at 25 GPa, (**d**) $P6/mmm$ structure of NpH_2 at 150 GPa, (**e**) $P6_3cm$ structure of NpH_3 at ambient pressure, (**f**) $Pnma$ structure of NpH_3 at 50 GPa, (**g**) $R\bar{3}m$ structure of NpH_3 at 100 GPa, (**h**) $Cmcm$ structure of NpH_3 at 160 GPa, (**i**) $I4/mmm$ structure of NpH_4 at 200 GPa. The large and small spheres denote neptunium and hydrogen atoms, respectively

[15]. Subsequently, Clerc confirmed that the effectiveness of the proposed algorithm was convergent [13].

3. The fusion arithmetic

   Angeline has introduced the selection operation in evolutionary computation and proposed a hybrid particle swarm optimization algorithm model [16, 17].

Because the PSO algorithms have some of significant advantages, such as simple model, easy to implement, and non-gradient and less variables, it shows excellent results in the continuous discontinuous optimization, combinatorial optimization, and dynamic optimization. Recently, PSO algorithm has been employed to various optimization problems for 0D nanoparticles or clusters, 2D layers and its atom adsorption, 2D surface reconstructions, and 3D crystals. The application of PSO in structure prediction has been proved to be a popular technique. The POS algorithms

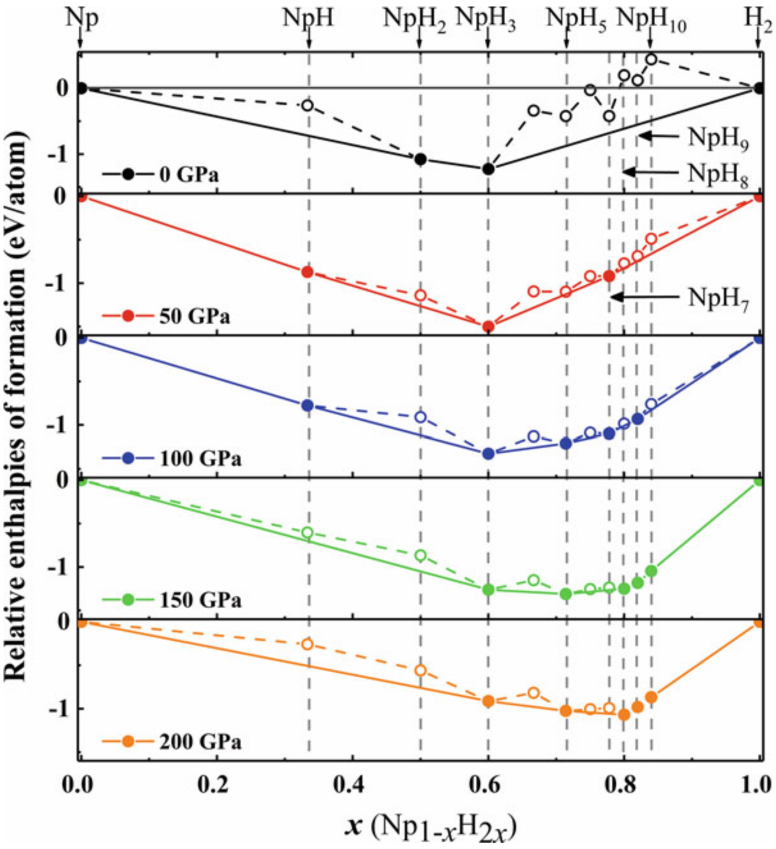


Fig. 5 Convex hulls of the Np–H system at selected pressures. Solid points connected by the solid line denote thermodynamic stable phases, while empty points connected by the dotted line represent unstable/metastable phases

have been interface to some mature structure prediction codes, such as CALYPSO [18] and USPEX [19], which have been successfully applied to investigate a great variety of materials at high pressures.

3 Differential Evolution Algorithms

3.1 Brief Introduction of the Differential Evolution Algorithm

Differential Evolution (DE) is a parallel direct search method which optimizes a problem by iteratively trying to improve a candidate solution for the multidimensional optimization problem. DE is a particular method to create new vector (also

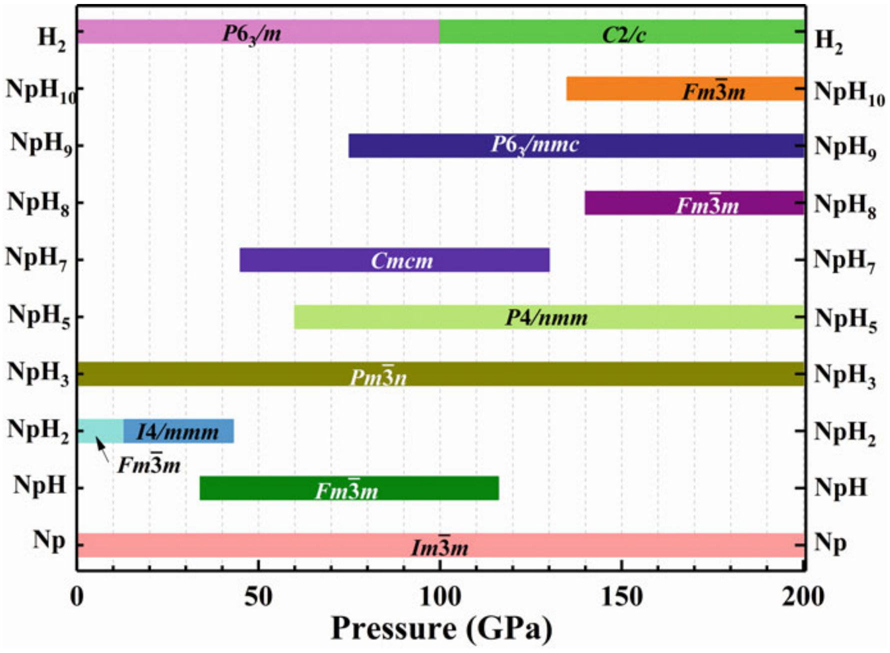


Fig. 6 Predicted pressure–composition phase diagram of the Np–H crystal phases

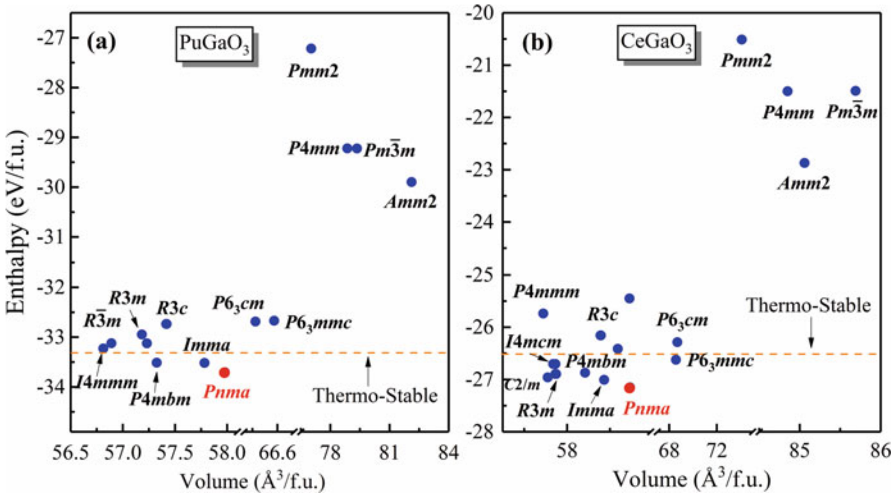
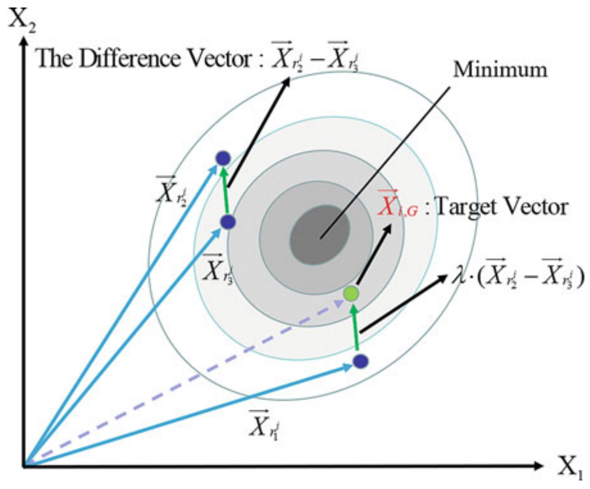


Fig. 7 Calculated enthalpy (for per formula unit) of predicted phases versus volume for (**a**) $PuGaO_3$ and (**b**) $CeGaO_3$ compounds with the PBEsol + U approach. Thermo-Stable level is the sum of the enthalpy of the decomposition products (PuO_2/CeO_2, Ga_2O_3, and α-Ga)

Fig. 8 DE mutation scheme for three random vectors



X_2
The Difference Vector : $\overrightarrow{X}_{r_2^i} - \overrightarrow{X}_{r_3^i}$
Minimum
$\overrightarrow{X}_{r_2^i}$
$\overrightarrow{X}_{i,G}$: Target Vector
$\overrightarrow{X}_{r_3^i}$
$\lambda \cdot (\overrightarrow{X}_{r_2^i} - \overrightarrow{X}_{r_3^i})$
$\overrightarrow{X}_{r_1^i}$
X_1

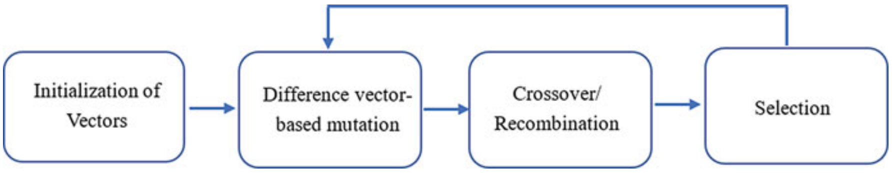


Fig. 9 The flowchart of the DE algorithm

known as genome/chromosome) for the population. While iterating over generations to evolve to an optimal state, existing chromosomes is used to create new offspring as potential candidates to make it to the subsequent generation. Diagram of the DE is depicted in Fig. 8. The main steps in DE are as follows [20]:

1. For each genome in the current population, we select three random vectors.
2. If a uniformly distributed random number ($rand_{i,j}$[0,1]) is less than the defined crossover rate, create a new offspring vector. Otherwise use the same genome as the parent.
3. Subtract two of these genome vectors.
4. Scale the difference of any two of these three vectors by a user-defined scale parameter λ.
5. Add the scaled difference vector to the third genome.

The *D*-dimension vector can be used to represent a set of *D*-dimensional parameters, called a single parameter [21]. A population consists of NP *D*-dimensional parameter vectors $x_{i,\,G}$, $i = 1, 2, \ldots,$ NP for each generation *G* [22]. The flowchart of the DE algorithm is shown in Fig. 9. Mutation, crossover, and selection are described below. For a more detailed description, please refer to [22].

3.1.1 Mutation

Mutation vectors are generated according to $v_{i,\,G+1} = x_{r1,\,G} + F \cdot (x_{r2,\,G} - x_{r3,\,G})$ with randomly selected indexes, corresponding to each target vector $x_{i,\,G}$. It is important to note that the indexes must be different from each other and different from the running index i. Consequently, the number of parameter vectors in a population must be at least four. F is an actual and constant factor $\in[0, 2]$ that controls the amplification of the difference vector $(x_{r2,\,G} - x_{r3,\,G})$.

It is necessary to note that the more subtle the differences between parameters of parent r_2 and r_3, the smaller the difference vector and therefore the perturbation. This means that if the population is close to the optimal value, the step size will decrease accordingly. This is similar to automatic step control in standard evolutionary strategies.

3.1.2 Crossover

The trial vector generated by the target vector is mixed with the mutated vector using the following scheme

$$u_{ji,G+1} = \begin{cases} vji, G+1 & \text{if}\,(r(j) \le \text{CR})\ \text{or}\ j = rn(i) \\ xji, G & \text{if}\,(r(j) > \text{CR})\ \text{and}\ j \ne rn(i) \end{cases}$$

where $j = 1, 2, \ldots, D$, $r(j) \in [0, 1]$ is the jth evaluation of a uniform random number generator. CR is the crossover constant $\in[0, 1]$. CR $= 0$ means that no crossover operator was used in the DE algorithm. rn(i) $\in (1, 2, \ldots, D)$ is a randomly chosen index which ensures that $u_{i,\,G+1}$ gets at least one element from . Otherwise, no new parent vector would be produced and the population would not alter.

3.1.3 Selection

If and only if the test vector produces a better cost function value than the parameter vector , it is accepted as the new parent vector of the next generation $G + 1$. This is a “greedy” option. If not, the target vector is retained again as the parent vector of generation $G + 1$. There are three strategy parameters altogether: NP: Number of members in a population, F: Amplification factor of the difference vector, CR: Crossover constant.

3.1.4 Other Variants of DE

There are numerous variants of DE which can be classified as DE/x/y/z, where x specifies the vector to be mutated, y is the number of difference vectors used, and z denotes the crossover scheme.

Table 1 Applications of differential evolution in medicine and pharmacology

Year	Researchers	Application
2004	Magoulas, Plagianakos, and Vrahatis	Colonoscopic diagnosis
2006	Koutsojannis and Hatzilygeroudis	Intelligent diagnosis and treatment of acid-base disturbances based on blood gas analysis data
2004–2005	Saastamoinen, Ketola, and Turunen	Sport medicine
2015	Konstantin Kozlov et al.	Geospatial immune
2017	T. Vivekanandan, N. ChSriman Narayana Iyengar	Heart disease

Table 2 Applications of differential evolution in optics community

Year	Researchers	Application
2004	Al-Kuzee, Matsuura, and Goodyear	Optimize plasma etch processes
2004	Zhang and Zhong	Calibrate camera
2005	Chan, Toader, and John	PBG design
2006	Akdagli and Yuksel	Laser diode nonlinearity
2006	Bluszcz and Adamiec	Optical stimulated luminescence decay
2006	Ling, Wu, Yang, and Wan	Design holographic grating
2007	Pan and Xie	Deformation measurement
2016	Fernando Lezama	Optical networks
2016	Soham Sarkar	Reflective optics system
2017	Md. Ghulam Saber	Optical material

x is note “rand” (randomly chosen population vector) or “best” (the best vector from the current population). Since we use only one difference vector, y is one in the described scheme. The current variant for z is “bin” (independent binomial experiments) which means crossover. With this notation, the conventional DE model can be written as . Another possibility is the method DE/best/2/bin, the mutant vector $v_{i,G+1} = x_{\text{best},G} + F \cdot (x_{r1,G} - x_{r2,G} + X_{r3,G} - x_{r4,G})$ can be obtained.

3.2 Applications of the Differential Evolution Algorithm

Recently, differential evolution algorithm has been intensively implemented in medical applications (focus on the diagnosis, classification, and treatment of cancer, Table 1), optics community (Table 2), the early accurate prediction of earthquakes (Table 3), and thermal engineering (Table 4) [23]. And most of the applications of differential evolution in physics focus on stellarator design and plasma. Other reported applications include chaos control [24], and optimization of the structure of atomic and molecular clusters [25].

Since the DE algorithm [26] was proposed by Storn and Price in 1995, the algorithm has been favored by more and more researchers. The research on the algorithm

Table 3 Applications of differential evolution in seismology

Year	Researchers	Application
1998, 2000	Bartal et al.	Optimize the seismic networks in Israel
2007	Ruzek et al.	Find seismic velocity models yielding travel times consistent with observed experimental data
2018	Thomas Meehan	The roots of this algorithm and main developments are examined to offer a better understanding of its essential features

Table 4 Applications of differential evolution in thermal engineering

Year	Researchers	Application
2006	Coelho	Modeling of a thermal system
2007	Babu and Munawar	Design of heat exchangers
2016	G. Balaji	Thermal generator maintenance scheduling
2017	Emerson Hochsteiner deVasconcelos Segundo	Economic optimization design for shell-and-tube heat exchangers
2019	Feng Tan	Thermal analysis of spindle
2020	Mohammad H. Nadimi-Shahraki	Introducing a multi trial vector approach to combine various search strategies

and its application has shown a rapid growth during past few years. The DE was used to solve Chebyshev polynomials at the early stages. By comparing with a variety of metaheuristic algorithms, it was found that the algorithm showed increasing effectiveness. The algorithm is currently used in many fields, including neural networks, industrial engineering, mechanical engineering, electronic engineering, electrical engineering, control engineering, civil engineering, software engineering, image processing, and other fields.

4 Conclusions

In summary, evolutionary algorithms model is a very simple but very powerful stochastic global optimizer, which is widely used in many scientific and engineering fields for function optimization. Evolutionary algorithms make it possible to use computers to extract useful knowledge from massive amounts of the acquired knowledge. It should be pointed out that there is no absolute good or bad between different evolutionary algorithms. We should select the appropriate evolutionary algorithms with different data structures in different situations.

References

1. Holland, J. (1975). *Adaptation in natural and artificial systems* (p. 100). Ann Arbor: University of Michigan Press.
2. Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, (Vol. 200, pp. 1–10).
3. Gao, W.-F., & Liu, S.-Y. (2012). A modified artificial bee colony algorithm. *Computers & Operations Research, 39*, 687–697.
4. Khader, A. T., Al-betar, M. A., & Mohammed, A. A. (2013). Artificial bee colony algorithm, its variants and applications: A survey. *Journal of Theoretical & Applied Information Technology, 47*(2).
5. Toktas, A., Ustun, D., Yigit, E., Sabanci, K., & Tekbas, M. (2018). Optimally synthesizing multilayer radar absorbing material (Ram) using artificial bee colony algorithm. In *2018 XXIIIrd International Seminar/Workshop on Direct and Inverse Problems of Electromagnetic and Acoustic Wave Theory (DIPED), 24–27 Sept 2018* (pp. 237–241).
6. Sonmez, M. (2011). Artificial bee colony algorithm for optimization of truss structures. *Applied Soft Computing, 11*, 2406–2418.
7. Gao, W., & Liu, S. (2011). Improved artificial bee colony algorithm for global optimization. *Information Processing Letters, 111*, 871–882.
8. Zhu, G., & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation, 217*, 3166–3173.
9. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95—International Conference on Neural Networks, 27 Nov–1 Dec 1995* (Vol. 4, pp. 1942–1948).
10. Wang, Y., Lv, J., Zhu, L., & Ma, Y. (2010). Crystal structure prediction via particle-swarm optimization. *Physical Review B, 82*, 094116.
11. Li, S., Ye, X., Liu, T., Gao, T., Ma, S., & Ao, B. (2018). New insight into the structure of Pugao3 from ab initio particle-swarm optimization methodology. *Journal of Materials Chemistry A, 6*, 22798–22808.
12. Ozcan, E., & Mohan, C. K. (1999). Particle swarm optimization: Surfing the waves. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 6–9 July 1999* (Vol. 3, pp. 1939–1944).
13. Clerc, M., & Kennedy, J. (2002). The particle swarm—Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*, 58–73.
14. Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters, 85*, 317–325.
15. Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), 16–19 July 2000* (Vol. 1, pp. 84–88).
16. Angeline, P. J. (1998). Using selection to improve particle swarm optimization. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)* (pp. 84–89). New York: IEEE.
17. Løvbjerg, M., Rasmussen, T. K., & Krink, T. (2001). Hybrid particle swarm optimiser with breeding and subpopulations. In Proceedings of the genetic and evolutionary computation conference (Vol. 2001, pp. 469–476). San Francisco, USA.
18. Wang, Y., Lv, J., Zhu, L., & Ma, Y. (2012). Calypso: A method for crystal structure prediction. *Computer Physics Communications, 183*, 2063–2070.
19. Glass, C. W., Oganov, A. R., & Hansen, N. (2006). Uspex—Evolutionary crystal structure prediction. *Computer Physics Communications, 175*, 713–720.
20. Das, S., & Suganthan, P. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*, 4–31.

21. Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study for differential evolution. *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, 10*, 293–298.
22. Storn, R., & Price, K. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.
23. Qing, A. (2009). Chapter 2. Fundamentals of differential evolution. In *Differential evolution: Fundamentals and applications in electrical engineering* (pp. 41–60). Hoboken, NJ: Wiley. https://doi.org/10.1002/9780470823941.
24. Zelinka, I. (2005). Investigation on evolutionary deterministic chaos control–extended study. *Heuristica, 1000*, 30.
25. Ali, M. M., Smith, R., & Hobday, S. (2006). The structure of atomic and molecular clusters, optimised using classical potentials. *Computer Physics Communications, 175*, 451–464.
26. Storn, R., & Price, K. (1995). *Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces*. Technical Report Tr-95-012. International Computer Science, Berkeley, CA.