Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(7): 12211–12239.
DOI: 10.3934/mbe.2023544
Received: 28 February 2023
Revised: 10 April 2023
Accepted: 08 May 2023
Published: 17 May 2023

Research article

An improved immune algorithm with parallel mutation and its application

Lulu Liu[*] and Shuaiqun Wang

College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China

* **Correspondence:** Email: liululu70531@163.com; Tel: +8602138282803; Fax: +8602138282803.

Abstract: The objective of this paper is to design a fast and efficient immune algorithm for solving various optimization problems. The immune algorithm (IA), which simulates the principle of the biological immune system, is one of the nature-inspired algorithms and its many advantages have been revealed. Although IA has shown its superiority over the traditional algorithms in many fields, it still suffers from the drawbacks of slow convergence and local minima trapping problems due to its inherent stochastic search property. Many efforts have been done to improve the search performance of immune algorithms, such as adaptive parameter setting and population diversity maintenance. In this paper, an improved immune algorithm (IIA) which utilizes a parallel mutation mechanism (PM) is proposed to solve the Lennard-Jones potential problem (LJPP). In IIA, three distinct mutation operators involving cauchy mutation (CM), gaussian mutation (GM) and lateral mutation (LM) are conditionally selected to be implemented. It is expected that IIA can effectively balance the exploration and exploitation of the search and thus speed up the convergence. To illustrate its validity, IIA is tested on a two-dimension function and some benchmark functions. Then IIA is applied to solve the LJPP to exhibit its applicability to the real-world problems. Experimental results demonstrate the effectiveness of IIA in terms of the convergence speed and the solution quality.

Keywords: immune algorithm; mutation operators; Cauchy mutation; Gaussian mutation; lateral mutation; Lennard-Jones potential problem

1. Introduction

Due to their potential to solve complex optimal problems, nature-inspired algorithms (NIAs), such as evolutionary algorithms, artificial immune systems, swarm intelligence, and simulated annealing, have received a lot of attention [1, 2]. One of the recently developed population based optimization methods is the immune algorithm (IA), which is based on the biological immune system theory [3–5]. IA simulates the mechanisms of the biological immune response when a biological immune system is exposed to an antigen. Clonal proliferation [6], negative selection [7], immune network [8], danger theory [9], and dendritic cell model [10] are the most commonly used mechanisms of the biological immune response. The proposed technique illustrated in this paper is mainly based on the clonal selection and proliferation principle [11–13]. IA has been successfully applied to many aspects such as function optimization [14], control [15], learning [16], pattern recognition [17], engineering optimization [18, 19], protein structure prediction [20, 21], and multiple sequence alignment [22].

In general, IA is used to solve combinatorial optimization problems due to its learning capability and self-organizing. Certain important issues should be considered to design an excellent IA. One such issue is maintaining the population diversity as the search progress. IA begins with various candidate solutions and manipulates them traversing the search space to achieve better solutions by some operations, i.e., mutation, selection and clonal proliferation. If the diversity of population is too limited to facilitate the algorithm visiting more unexplored search regions, the premature convergence of algorithm occurs, which is likely to result in local optimal trapping problems. To alleviate this problem, Gao et al. [23] has proposed an IA with feedback mechanisms to effectively handle the population size and thus maintain the diversity well. Other attempts on resolving this problems may refer to [24, 25].

Another key issue while designing IA is to balance exploitation and exploration of the search in decision space. Exploration is the process of visiting the entirely new regions of a search space, whereas exploitation is the process of visiting those regions of a search space through the neighborhood of previously visited points [26]. In order to design a successful algorithm, it is necessary to establish a good ratio between exploration and exploitation. After realizing this, Khilwani et al. [27] proposed a fast immune algorithm (FCA) by combining cauchy mutation (CM) with gaussian mutation (GM). CM enables the search to carry out long distance jumps, thus specializing in exploring unvisited regions [28]. Besides, GM is more capable of exploitation in small regions of a smooth decision space. Nevertheless, due to the randomness of the search dynamics both in CM and GM, the performance of the algorithm may not achieve better convergence speed. One possible method to improve the convergence speed of the algorithm is the usage of more enriched search dynamics. In [29], Lee et al. has proposed a new mutation operator, which is based on Lévy distribution to generate an offspring that is far away from its parent, aimed to search the decision space much faster. Moreover, information exchanges (i.e., crossover) between two different solutions may lead a fast assimilation for the evolution. Most of the solutions undergone information exchanges will possess some common information (or sub-solution), therefore accelerating the convergence speed. In [30], Gong et al. has proposed a Baldwinian learning operator to enable the current solution learn from the co-adapted alleles in certain environments. Motivated by the mechanism of idiotypic network [31], a lateral interaction mutation (LM) has been put forward to effectively realize the knowledge exchange during the period of getting solutions [32].

Essentially, the evolution of solutions in IA is driven by the selection and mutation operators. Selection promotes the searching towards the regions of the best individuals. Heavy selection pressure urges the search towards more exploitation, and little selection pressure pushes the search towards more exploration [33,34]. Mutation enables the original solutions to move toward different regions. On the one hand, mutation randomly modifies individuals with a given probability, and thus increases the structural diversity of a population. From this point of view, a mutation operator is more of an exploration operator. Such an operator (CM, GM, etc.) facilitates the recovery of genetic diversity which would be lost during the selection phase, and explores new solutions. On the other hand, mutation can also refine an individual by conditionally incorporating the information derived from other individuals into it, aiming to generate the possibility of better offspring and faster convergence. From this perspective, such a mutation operator (LM, etc.) is more of an exploitation operator. As stated above, it is crucial to balance exploration and exploitation of the search. As a result, apart from the selection operator, two key questions should be considered to design a positive IA: 1) which kind of mutation operators should be used in the algorithm, and 2) what ratio of each mutation should be implemented by properly managing the control-parameter settings.

To address the above problems, a new parallel mutation (PM) framework is proposed in this paper. In PM, three distinct mutation operators (GM, CM and LM) are strictly selected to carry out the individual evolution. Due to its fine-grained search characteristic, GM is responsible for the exploitation within a small search neighborhood. Owing to its coarse-grained search property, CM is capable to perform hill climbing mutations, and thus it will be beneficial to make the algorithm jump out of the local optima. Assisted by the surrounding individuals, the individual undergone LM is manipulated to learn knowledge from the neighborhood, therefore a fast convergence speed would be realized. During the implementation of the algorithm, each single mutation operator in PM is assigned an operation probability and a randomly generated probability is used to control which operator will be carried out to perform the search. By making use of the introduced PM framework, an effective immune algorithm (IIA) is proposed. It is expected that IIA can well balance exploration and exploitation of the search, by means of not only the reduction of the number of useless or redundant iteration number, but also the guidance of the search to promising regions which are close to the global optima with a high probability.

Lennard-Jones potential problem (LJPP) is a potential energy minimization problem, where neutral atoms are subject to two distinct forces under the limit of large distance and short distance: a dispersion force at long ranges, and a repulsion force at short distance, resulting from the overlapping electron orbits. In LJPP, the objective function is non-convexity, and there are huge number of local minima which is growing exponentially with the number of the atoms. Due to the excessive number of local minima for LJPP, it make extremely hard to find a global minimum of the energy. In fact, LJPP belongs to the classical NP-hard problems [35, 36] for which no algorithm is guaranteed to find the global optimal. Hence, the traditional local optimization methods may not be efficient for such problems. At the earliest, Northby firstly introduced a effective and successful methods to LJPP [37] and further refined in [38]. This method is beginning local optimization with initial configurations which are produced by randomly placing atoms in predefined points in space. Then, a large number of efficient approaches are used to solve LJJP, such as basin hopping method [39], the big-bang method [40], Leary's descent method [41], genetic algorithms [42–44], differential evolution [45] and particle swarm optimization [35]. A geometric crossover technique has been introduced in [46, 47].

To verify the effectiveness and applicability of the algorithm, IIA is testing on a two-dimension function, some benchmark functions and a real-world LJPP. Benchmark functions are often used in optimization scenarios, while LJPP is more practical and is hard to be solved. Thus, IIA is tested from two different aspects. Via the function optimization, especially a multimodal function, the capability of jumping out of local minimum of IIA is well illustrated and demonstrated. While via resolving the real-world optimization problem, IIA has a certain potential to deal with the larger scale problems. Experiments are conducted based on the above two problems, and the illustrative and comparative results could clearly show the effects of PM on IIA.

The rest of the paper is organized as follows. Section 2 describes the the model of LJPP. Section 3 introduces the PM framework and presents the details of IIA. The experimental results and discussions of IIA whenever solving the benchmark functions and LJPP are shown in Section 4. Finally, Section 5 presents some general remarks to conclude the paper.

The contributions of this study are as follows: 1) We propose a novel algorithm for improving the performance of search algorithms. 2) We demonstrate the effectiveness of the proposed algorithm through extensive experiments on benchmark problems. 3) We provide a comprehensive analysis of the proposed algorithm, including its advantages and limitations. 4) We compare the proposed algorithm with existing state-of-the-art algorithms in the literature and show that it outperforms them in terms of search performance. 5) We discuss the potential applications of the proposed algorithm in various domains.

2. Lennard-Jones potential problem

Lennard-Jones potential problem (LJPP) is a potential energy minimization problem and it involves the minimization of molecular potential energy associated with pure Lennard-Jones cluster [45, 48]. LJPP is a multi-modal optimization problem comprised of an exponential number of local minima [48]. The lattice structure of Lennard-Jones cluster has an icosahedral core and a combination of surface lattice points. Most of the global minima which have structures based upon the Mackay icosahedrons can be found in the Cambridge Cluster Database. An algorithm can be tested over this function because of its capability to conform molecular structure, where the atoms are organized in such a way that the molecule has minimum energy. Lennard-Jones pair potential for D atoms which is given by the Cartesian coordinates

$$\overrightarrow{p_i} = \{\overrightarrow{x_i}, \overrightarrow{y_i}, \overrightarrow{z_i}\}, i = 1, 2, ...D \tag{2.1}$$

is represented as follows:

$$V_D(p) = \sum_{i=1}^{D-1} \sum_{j=i+1}^{D} (r_{ij}^{-12} - 2.r_{ij}^{-6}) \tag{2.2}$$

where

$$r_{ij} = \left\|\overrightarrow{p_j} - \overrightarrow{p_i}\right\|_2 \tag{2.3}$$

with gradient

$$\nabla_j V_D(p) = -12 \sum_{i=1, i\neq j}^{D} (r_{ij}^{-14} - r_{ij}^{-8})(\overrightarrow{p_j} - \overrightarrow{p_i}), j = 1, 2, ..., D \tag{2.4}$$

Lennard-Jones potential has the minimum value at a particular distance between the two points [49]. The variation of Lennard-Jones pair potential $V(r) = r^{-12} - 2.r^{-6}$ with pair distance r have been plotted in Figure 1. It can be clearly found that it has a unique minimum value of -1 for a particular value of r = 1. Its value increases very fast when r decreases from the optimum value and tends to infinity near r = 0. This tendency of pair potential curve makes it turn to be a very hard optimization problem.

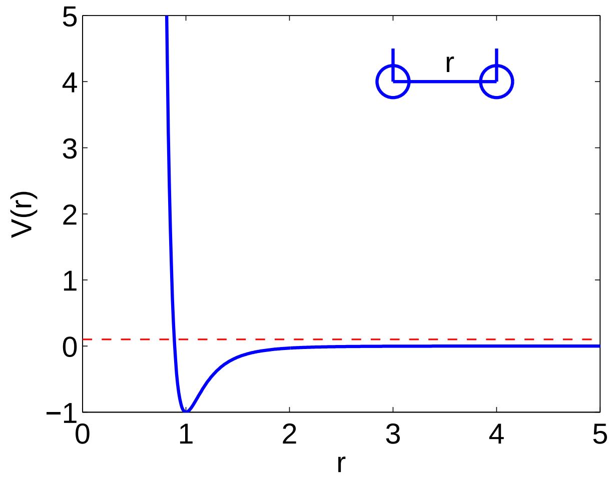

Figure 1. Variation of pair potential with *r*.

3. Immune algorithm with parallel mutation

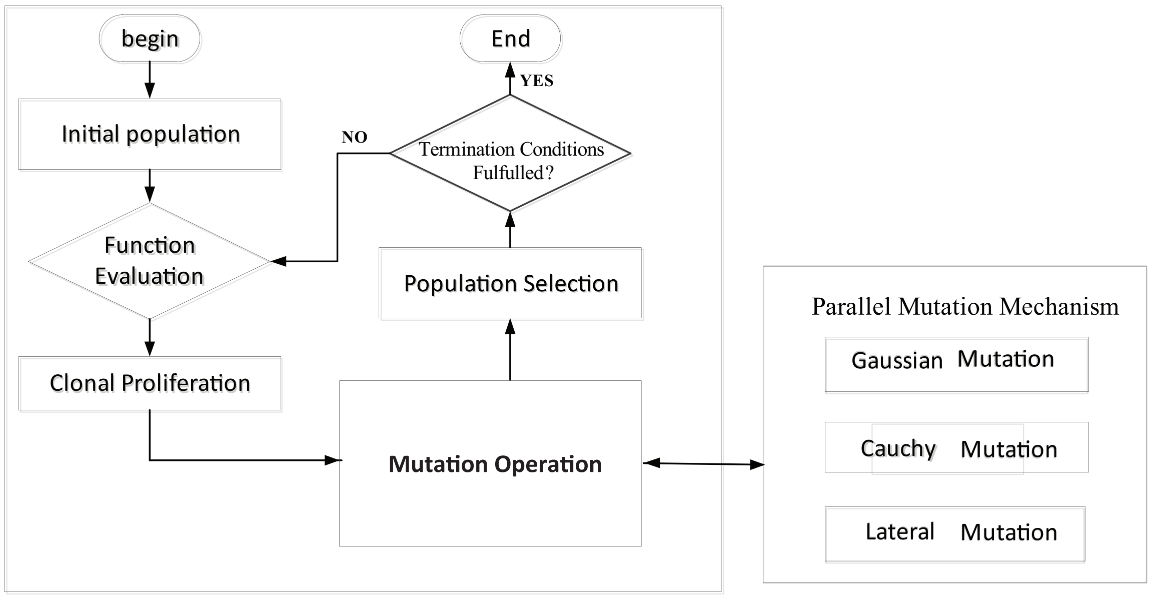

Figure 2. The procedure of a generic immune algorithm.

In the literature, several IA models inspired by the mechanism or function of immune cells in the human immune system have been developed [12, 50, 51]. Among them, most IA techniques emulate the clonal selection principle which depicts the response procedure when human being is exposed to the extrinsic substances such as the bacteria and viruses. In clonal selection principle, two key operations take place, i.e., the differentiation and proliferation of the immune cells. From the algorithmic perspective, differentiation can be viewed as mutation operator of individuals, while proliferation be regarded as reproduction of individuals. The generic computational framework of IA can be depicted on the left and the proposed PM mechanism on the right in the Figure 2. Intuitively, the quality of mutated populations is determined by the capacity of mutation operators [52, 53], which directly influences the performance of the algorithm. In other words, mutation drives the population to evolve more promising individuals. In order to make the mutation operator possess more search dynamics, a parallel mutation framework is introduced in this paper. Three distinct single mutation operators are strictly chosen to be implemented in the PM.

3.1. Mutation operators

To make the paper self-explanatory, three mutation operators including GM, CM and LM are introduced briefly and respectively. To make the description more concise, We combine the representation of the existing population of people to be $\{X_1, ..., X_i, ..., X_N\}$, where the i-th individual $X_i = (x_{i1}, ..., x_{ij}, ..., x_{iD})$, N stands for the population's size,whereas D stands for the current optimization problem's dimension. As to LJPP, D is the number of atoms. Being manipulated by mutation operators, the mutated population is represented by $\{X'_1, ..., X'_i, ..., X'_N\} = P\{X_1, ..., X_i, ..., X_N\}$, where $X'_i = (x'_{i1}, ..., x'_{ij}, ..., x'_{iD})$, P represents the mutation used in IIA, and $P \in$ {CM, GM, LM}.

Gaussian mutation (GM): the mean value μ and the standard deviation σ are two GM characteristics that are utilized to calculate the mutation's step size. The one-dimension gaussian density function with the mean value μ and the standard deviation σ is defined by:

$$f_{gau}(x) = \frac{1}{\sigma\sqrt{2\pi}} exp^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

Like many other researches [27,54,55], the origin-centered gaussian density function is the foundation of the simplified GM mutation strategy when $\mu = 0$, $\sigma = 1$. In a single iteration of the population, GM is implemented based on the following equation:

$$x'_{ij} = x_{ij} + s_g N_j(0, 1) \tag{3.2}$$

where $N_j(0, 1)$ is a gaussian random number that is normally distributed and has a mean of 0 and a standard deviation of 1. The population's serial number is represented by the symbol i, and the dimension is represented by the symbol j. s_g is the step size of mutation, and it is generated by [27,55]:

$$s_g = random(+, -)\sqrt{2ln(w_g\sqrt{2\pi})} \tag{3.3}$$

where w_g is a random number that is generated uniformaly across the range of $[0, f_{gau}(0)]$.

Cauchy mutation (CM): the cauchy distribution,as depicted in Eq (3.4), is the primary foundation of CM.

$$f_{cauchy}(x) = \frac{1}{\pi}\frac{1}{1 + x^2} \tag{3.4}$$

The absence of an expectation is the most significant feature of cauchy distribution [27,55]. The shape of $f_{cauchy}(x)$ is identical to that of the gaussian density function, but it moves very slowly toward the axis. The CM is implemented as follows [27]:

$$x'_{ij} = x_{ij} + s_c\delta_j \tag{3.5}$$

where δ_j is a cauchy random variable. s_c is the step size of cauchy mutation which is generated by:

$$s_c = random(+, -)\sqrt{\frac{1}{w_c\pi} - 1} \tag{3.6}$$

where s_c is a uniformaly generated random number in the range $[0, f_{cauchy}(0)]$.

Lateral mutation (LM): the lateral mutation according to the idiotypic network theory [8,32], lateral interaction typically occurs in immune systems. In other words, not only a foreign antigen but also the external idiotopes can identify each paratope on a person. Motivated by this mechanism, LM is implemented as:

$$x'_{ij} = (1 - \beta)x_{ij} + \beta x_{kj} \tag{3.7}$$

where $k \in \{1, 2, ..., N\}$ and $k \neq i$. The learning rate $\beta \in (0, 1)$ is a real number generated randomly.

The reason why we choose the forementioned three mutation operators to be involved in PM is their distinct search dynamics. In Figure 3, we summarize the characteristics of each mutation operator respectively. The optimization problem's solution space is depicted by the solid rectangle S. The dashed circles denote contour lines of affinity, and the inner circles indicate that they represent higher affinities than the outer ones. In this figure we can notice that individuals mutated by GM and CM only utilize random perturbation on the individual itself, while those mutated by LM can make use of the information from the environment. As remarked by Yao et al. [55], Due to its long, flat density function tails, CM is more likely than GM to produce offspring far from its parent. It is more likely to carry out large variations in individuals because it is possible to execute longer jumps with a great probability. As a result, it is frequently used as a learning strategy to address the local optima trapping issue. However, as stated in [30,56], learning from one's surroundings offers a positive alternative and, most likely, a simpler approach to improving one's search performance. In particular, the mechanism in LM directs the current search by utilizing the data for a randomly selected member of the population. Successful guidance necessitates a significant amount of redundant searching if the chosen guide is far from the global optimal solution, as the quality of the chosen guide is crucial. It is worth emphasizing that only one or two single mutation operators can not fully realize the exploration and exploitation of the search. It is suggested that a well-designed mutation operator should contain at least three properties: 1) a fine-grained search ability derived from GM, 2) a coarse-grained search aptitude rooted in CM, and 3) a fast convergence manipulated by LM.

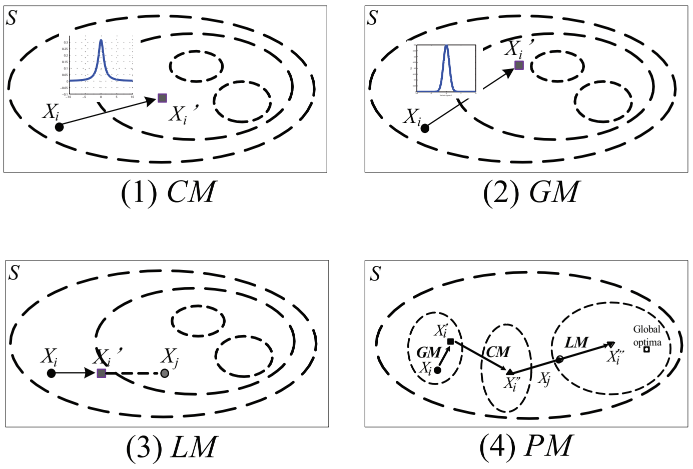

Figure 3. The characteristics of the mutation operators: CM, GM, LM and PM.

3.2. Framework of the PM mechanism

In order to incorporate the three above mentioned complementary search dynamics in the algorithm simultaneously, a parallel mutation (PM) framework is proposed in this paper. The pseudocode for implementing the mutation mechanism is shown in Table 1. Three parameters $Prob_{CM}$, $Prob_{GM}$ and $Prob_{LM}$ satisfying Eq (3.8) are used to control the probability of the implementation of each mutation operator.

$$Prob_{CM} + Prob_{GM} + Prob_{LM} = 1 \tag{3.8}$$

In the process of solution evolution, the current individual X_i may mutate to X_i' by means of GM, in the next generation, the individual X_i' may mutate to X_i'' by means of CM, then the individual X_i'' may mutate to X_i''' by means of LM. Only one mutation operator is used in each iteration of the individual maturation, and the mutation operator that is used is chosen at random. However, PM will have all of the search dynamics of the embedded single mutation operators because, after a large number of iterations, each single mutation operator will be implemented multiple times.

Table 1. The pseudo-code of PM mechanism in IIA.

```
Begin: Input an individual $X_i$
initialize $Prob_{CM}$, $Prob_{GM}$ and $Prob_{LM}$
set $q = rand()$
If $0 < q < Prob_{GM}$, execute the cauchy mutation GM
Else-If $q < Prob_{CM} + Prob_{GM}$, execute the gaussian mutation CM
Else-If $q < Prob_{CM} + Prob_{GM} + Prob_{LM}$, execute the lateral mutation LM
End-If
End: Output the matured individual $X_i'$
```

In the PM framework, each embedded single mutation operator will be carried out if its corresponding cumulative probability is larger than a randomly generated number, namely q, thus it is deduced that this operator's search dynamic contributes to the search of the entire algorithm. On the other hand, rather than being implemented deterministically, the single operator is implemented at random due to the stochastic nature of q. In addition, the fact that the implementation probabilities for each mutation operator are adaptable suggests that resetting these probabilities can effectively coordinate the search dynamics of the resulting PM. Several remarks can be concluded regarding PM:

1) PM inherits GM's ability to exploit individuals and can conduct fine-grained searches within a relatively small neighborhood.

2) Due to the occasional large mutations in CM, PM is able to conduct coarse-grained searches in a wider area around the individual. In some cases, these larger mutations are advantageous in that they cause the algorithm to jump out of the local optima.

3) The environment of LM can provide PM with information, such as from a guiding individual. Such a mutation would be very effective in accelerating convergence if the guiding individual is al-

ready close to the global optima, and it would also be very possible to improve the average quality of solutions.

3.3. The suggested probability of the mutation operators

It is important to set the values of the operation probabilities for each single mutation parameter. Such combination of setting values puts significant influence on the performance of the algorithm. Usually, the trial-and-error approach, which is a time-consuming and tedious method, can be performed in an ad-hoc manner to set these values. In this paper, two kinds of parameter setting strategies are given. One is a simpler approach, the suggested probability of each mutation operator is equal. The other is affinity-based rule to determine the values. These two approaches will be introduced and performed on benchmark functions and LJPP to compare the performance, respectively. After setting the initial probability of each mutation operator, the adaptive mechanism has been proposed in our algorithm. The learning probability values of each mutation operator are gradually changing according to the iterations. The general form is given as following:

$$Prob_{CM} = Prob_{CM} * (1 - \frac{t}{T_{max}}) \tag{3.9}$$

$$Prob_{GM} = Prob_{GM} * (1 - \frac{t}{T_{max}}) \tag{3.10}$$

$$Prob_{LM} = Prob_{LM} + \frac{t}{T_{max}} * (Prob_{CM} + Prob_{GM}) \tag{3.11}$$

in which, $Prob_{CM}, Prob_{CM}, Prob_{LM}$ represents the participation probability of each mutation operator respectively, T_{max} is the number of maximum iterations, t is the current iteration.

1) Simpler approach with a gradually decreasing learning values (PMGD).

This is a very simple method which is not related with the search ability and search dynamic mechanism of each mutation operator. In this article, we design the strategy only related to the number of iterations and the current iteration. At the beginning of the experiment, the probability of each mutation operator is set to equal (1/3).

2) Approach depending on the fitness values of each mutation operator (PMDF).

After preliminary experiments, the optimal performance of each single mutation operator is verified. The principle of the deterministic setting rule is that the more powerful the search capacity of the mutation is, the larger operation probability it possesses, and vice versa. It is expected that the exploration and exploitation of the search can be well balanced by setting operation probabilities in such a manner. Further experimental evidences are presented in Section 4. The process of probability decision is as follows:

In the preliminary experiment, the search capacities of each single mutation operator are observed based on 30 independent runs. The mean objective function values obtained by CM, GM, LM separately are represented as f_{CM}, f_{GM}, f_{LM} respectively. The maximum value among them are computed to normalize the operation probability.

$$f_{max} = max\{f_{CM}, f_{GM}, f_{LM}\} \tag{3.12}$$

$$f'_{CM} = abs\{f_{CM} - \lceil f_{max} \rceil\} \tag{3.13}$$

$$f'_{GM} = abs\{f_{GM} - \lceil f_{max} \rceil\} \tag{3.14}$$

$$f'_{LM} = abs\{f_{LM} - \lceil f_{max} \rceil\} \tag{3.15}$$

where $\lceil f_{max} \rceil$ rounds up to the closest integer that approaches the value of f_{max}. As for minimization problems, the adjusted function values f'_{CM}, f'_{GM}, f'_{LM} are obtained by reducing $\lceil f_{max} \rceil$, where $abs\{.\}$ means the absolute value of its variable. The deterministic setting rule of these operation probabilities are based on the following equations.

$$Prob_{CM} = \frac{f'_{CM}}{f'_{CM} + f'_{GM} + f'_{LM}} \tag{3.16}$$

$$Prob_{GM} = \frac{f'_{GM}}{f'_{CM} + f'_{GM} + f'_{LM}} \tag{3.17}$$

$$Prob_{LM} = \frac{f'_{LM}}{f'_{CM} + f'_{GM} + f'_{LM}} \tag{3.18}$$

It is clear that the above three operation probabilities belong to the interval of $[0, 1]$ while satisfying Eq (3.8). The more fit individuals a mutation operator generates, the larger operation probability it possesses. By doing so, the three distinct mutation operators can be well merged together in PM to achieve a better search performance. It should be noted that, although it is possible to make these operation probabilities self-adapted or evolved by algorithmic parameter's coordination [57], such advanced setting methods are too sophisticated to design, which are even more difficult than the design of the optimization problem itself. As a result, general guidelines on how to set these operation probabilities are provided based on Eqs (3.12)–(3.18) after the search features of each mutation operator are identified.

3.4. Procedures of IIA

The general procedure of IIA consists of five components as shown in Figure 2. The procedure involving initialization, affinity evaluation, proliferation, mutation, and selection are regarded as one generation of the evolution of population. The complete algorithm with PM incorporated is stated here.

1) Initialization: on the part of the function optimization problems, randomly initializes N individuals in the D-dimensional decision space P_i, $i = 1, 2, ..., N$, where $P_i = (p_{i1}, ..., p_{ij}, ..., p_{iD})$ is the ith individual in the D-dimensional decision space. It should be noted that the representation of solutions for LJPP is slightly different from that of the function optimization problems. For LJPP, the search dimension size is $3 \times D$, where D denotes the number of the atoms, and 3 means the three Cartesian dimension that atoms locate. The upper and lower bounds of the ith dimension in the decision space is $[low_j, up_j]$. To make the randomly generated individual be a feasible solution for the problem, the following formula is used.

$$p_{ij} = low_j + \mu * (up_j - low_j), \quad i = 1, ..., N; j = 1, ..., D \tag{3.19}$$

where $\mu \in [0, 1]$ is a random number with the uniform distribution.

2) Affinity Evaluation: compute the affinity values $f(P_i)$, $i = 1, ..., N$, for each individual in the population. In general, an individual with larger affinity value indicates a better solution for the problem. For minimization problems, the affinity values is reversely proportional to the function values (or

energy values in Eq (2.2)):

$$f(P_i) = V_D(P_i), \quad i = 1, ..., N \tag{3.20}$$

3) Clonal Proliferation: for each individual P_i, asexually produces m_i clones where

$$m_i = \left\lfloor M * \frac{N - i}{N} \right\rfloor \tag{3.21}$$

where M is the clone size, $\lfloor . \rfloor$ is the operator that rounds its arguments towards the closest integer. m_i denotes the clone number of the ith individual after all the individuals in an descending order of the affinity are sorted. It is obvious that the amount of generated clones of an individual is inversely proportional to its affinity. In other words, the better the individual is, the more clones it produces. Afterwards, the population of individuals is represented by $(P_{1,1}, P_{1,2}, ..., P_{1,m_1}; ...; P_{N,1}, P_{N,2}, ..., P_{N,m_N})$;

4) Mutation: for each clonal individual P_{ij}, $i = 1, ..., N$, $j = 1, ..., m_i$, mutates it according to Algorithm 1. After mutation, the population becomes $(P'_{1,1}, P'_{1,2}, ..., P'_{1,m_1}; ...; P'_{N,1}, P'_{N,2}, ..., P'_{N,m_N})$. At this point, it is worth indicating that the mutated individual goes out of the search bounds is re-initialized by using Eq (3.19) to make it to be a feasible solution.

5) Selection: evaluates the affinity of each individual in the mutated population to obtain the updated population P_i, $i = 1, 2, ..., N$, where

$$P_i = P'_{i,t} \;\; s.t. \; f(P'_{i,t}) = max\{f(P_{i,j})\} \tag{3.22}$$

6) Termination condition: The above steps 2)–5) would be repeated until the termination condition (i.e., the iteration number reaches T_{max}) is satisfied. Finally the optimal individual is outputted.

4. Experimental studies

4.1. Parameter setting

Table 2. The parameters used in IIA.

Parameter	Meaning	Value
N	The population size	50
M	The constant clonal size	10
D	The dimension of problem	–
$Prob_{CM}$	The operation probability for *CM*	–
$Prob_{GM}$	The operation probability for *GM*	–
$Prob_{LM}$	The operation probability for *LM*	–
T_{max}	maximum iteration number	–

Note: the symbol "–" means it is problem-oriented.

The algorithm's performance is strongly influenced by parameter values. Before it can be used to optimize problems, this paper's user-defined parameters need to be well coordinated. Table 2 summarizes the parameters used in IIA. N denotes the population size of the algorithm. Large value of N results in more individuals involving in performing search in the decision space, thus might achieve better search performance. However, large value of N also implies more implementation times of mutation and selection of the operators, thereafter it requires more computational time. As suggested in [11, 58], a trade-off between the quality of solutions and computational time is to set N as 50. The clonal size can be controlled by parameter M. Large value of M indicates more clones produced in the algorithm, and the more fit the individual is, the more plentiful clones it generates. As all m_i clones of each individual P_i will undergo the progress of the parallel mutation, P_i will have m_i times of opportunities to mature (i.e., become a more fit individual for the problem). Nevertheless, too many times of mutation for an individual might guide the search implemented in the same region, thus would easily cause diversity loss and pre-maturation of the population. Preliminary simulation results suggest that a compromising value for M is 10. Moreover, the parameter D means the search dimension which is problem-oriented.

The three operation probabilities in PM ($Prob_{CM}$, $Prob_{GM}$ and $Prob_{LM}$) are difficult to determine. In the previous section, we have proposed two types of determined mechanism for operation probabilities. The first category is easy to perform and only depends on the current iteration, however, the second type is different from the first one and based on the search abilities of every operator. We will illustrate these two mechanisms in the following.

1) The initial participation probability of CM, GM, LM with PMGD mechanism

The initially participation probabilities of the three mutation operators in the PMGD mechanism are equal. Therefore, the probability is set 1/3.

2) The initial participation probability of CM, GM, LM with PMDF mechanism

The participation probabilities of the three mutation operators depend on the fitness values of search capability in the PMDF mechanism. Therefore, the fitness value of a single mutation operator must be obtained and the participation probabilities are calculated according to the fitness values proportion.

Table 3. The fitness, amendatory fitness and operation probability obtained by each single mutation operator.

atoms		*CM*	*GM*	*LM*
10	f	-28.054846	-28.185371	-28.417731
	f'	0.054846	0.185371	0.417731
	Prob	0.083	0.282	0.635
38	f	-154.084139	-154.334810	-159.646073
	f'	0.084139	0.334810	5.646073
	Prob	0.014	0.055	0.931

The deterministic values are given below. Because the search performance of the algorithm strongly depends on their values. We present some guidelines on how to set them after the search capacity of each single mutation operator is observed. The preliminary experiments are conducted based on the LJPP with two different groups of atoms. Table 3 records the final affinity values f (Eq (3.20)) obtained by every single mutation operator, the resultant amendatory affinity values f' (Eqs (3.13)–(3.15)), and the corresponding operation probabilities *Prob* (Eqs (3.16)–(3.18)). From this table, it is obvious that the operator *LM* possesses the best search performance, *GM* is the second, and *CM* is the worst. The results are consistent with the search properties of each single mutation operator. *LM* enables individuals to exchange information with each other, thus accelerate the convergence speed to acquire better solutions. *GM* modifies the individual to carry out a local exploitation within the neighborhood slightly, and improves the solution quality gradually. *CM* is able to visit the search space by jumping to a faraway location suddenly, but it can not focus on exploiting a small region, therefore its search performance is the worst. Actually, the search dynamics of the three mutation operators in PM are complementary. After observing such a phenomenon, the general guidelines of setting the operation probabilities can be summarized as:

1) None of the single mutation operator can be omitted in PM which means the probability can not be set to zero.

2) According to the search capacity, the ranked order during operation probabilities is $Prob_{CM} < Prob_{GM} < Prob_{LM}$.

Any combination obeying the two guidelines can be regarded as a suitable setting, although the actual setting is problem-oriented and it might follow Eqs (3.16)–(3.18) in every iteration. In the experiment, these values are settled as $Prob_{CM} = 0.1$, $Prob_{GM} = 0.3$ and $Prob_{LM} = 0.6$ in PMDF mechanism.

3) Other possible participation probability among three mutation operators

Tables 4 and 5 record the statistical values of solutions obtained by IIA with different learning probabilities in PM for the unimodal function f_1 and the multimodal function f_3, respectively. These learning probabilities are taken from eight representative design points, where a, b and c in "a/b/c" denote the assigned implementation probabilities of CM, GM and LM respectively. Tables show that the setting of the implementation probabilities in PM using "0.1/0.3/0.6" exhibits the best learning performance in terms of solution qualities. It is worth emphasizing that, although "0.1/0.3/0.6" might not be the optimal setting of the probabilities, it outperforms most of the other settings due to the usage of experimental design with mixtures. However, the initial participation probability is 1/3 for each mutation operator which is significantly better than single mutation operator or two mutation operators. Therefore, we can conclude that the IIA algorithm is the best when three mutation operators participate simultaneously, and outperforms single or two mutation operators.

Table 4. The statistical value of solutions obtained by IIA with different learning probabilities in PM for f_1 under 30 runs.

f_1:Probability	Mean	Std	Median	Min	Max
CM (1/0/0)	2.26E-04	6.20E-05	2.19E-04	1.09E-04	3.80E-04
GM (0/1/0)	6.79E-04	2.22E-04	6.46E-04	2.62E-04	1.20E-03
LM (0/0/1)	4.85E-04	1.70E-03	3.74E-07	1.18E-22	8.80E-03
CGM (0.5/0.5/0)	4.27E-05	5.42E-05	5.61E-06	8.15E-06	4.78E-05
CLM (0.5/0/0.5)	3.28E-06	3.37E-05	2.63E-08	5.50E-10	6.65E-06
GLM (0/0.5/0.5)	1.45E-06	1.57E-05	1.39E-08	1.29E-10	1.98E-06
IIA(1/3/1/3/1/3)	3.09E-09	9.37E-09	4.75E-011	8.67E-19	3.75E-08
IIA (0.1/0.3/0.6)	**7.05E-11**	**2.67E-10**	**5.34E-15**	**1.09E-24**	**1.42E-09**

Table 5. The statistical value of solutions obtained by IIA with different learning probabilities in PM for f_3 under 30 runs.

f_3:Probability	Mean	Std	Median	Min	Max
CM (1/0/0)	2.49E+02	2.37E+01	2.52E+02	2.02E+02	2.81E+02
GM (0/1/0)	2.25E+02	2.64E+01	2.25E+02	1.70E+02	2.72E+02
LM (0/0/1)	3.35E-02	1.82E-01	8.65E-11	0	9.95E-01
CGM (0.5/0.5/0)	4.21E+00	3.42E+00	3.51E+00	5.08E-02	3.17E+01
CLM (0.5/0/0.5)	4.23E-02	3.83E-01	3.33E-02	5.50E-03	1.24E-01
GLM (0/0.5/0.5)	7.70E-03	1.81E-01	1.89E-03	3.91E-03	1.30E-02
IIA (1/3/1/3/1/3)	6.94E-04	4.44E-03	6.49E-06	0	1.55E-06
IIA (0.1/0.3/0.6)	**5.15E-09**	**1.06E-08**	**1.82E-10**	**0**	**3.91E-08**

4.2. Identification of search characteristics on function optimization

4.2.1. Benchmark functions

Table 6. Benchmark problems used in the experiments.

Function Definition	Dim.	Domain $[low, up]^D$	f_{min}
$f_1(X) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^D$	0
$f_2(X) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	30	$[-10, 10]^D$	0
$f_3(X) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]^D$	0
$f_4(X) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]$ $+(y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]^D$	0
$f_5(X) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]^D$	-10.3909
$f_6(X) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]^D$	-10.53

In this paper, some benchmark functions with different categories are selected to evaluate the performance of the proposed algorithm. f_1 and f_2 are unimodal functions, f_3 and f_4 are multimodal functions with many local minima, f_5 and f_6 are multimodal functions with a few local minima. The function formula of them is given in Table 6 and the more detailed explanation of each function is listed in Appendix in the reference [55]. The symbol "Dim." denotes the dimension of the function. The domain $S = [low, up]^D$ represents the search space of feasible solutions for the D dimension problem. f_{min} is the global optimal solution. It should be noted that the results obtained are under the settings for parameters by, $N = 30$, $M = 5$, $T_{max} = 2000$ for high-dimensional functions and $T_{max} = 100$ for low-dimensional functions in all compared algorithms.

The first set of experiments aims to compare the search dynamics of the parallel mutation mechanism with the single mutation operators and the performance between the PMDF approach and PMGD strategy. To realize this, we constructed three variants of IIA using only one single mutation operator by turns. IIA-1, IIA-2 and IIA-3 represent the utilization of *CM*, *GM* and *LM* in the algorithm respectively. Due to the different deterministic mechanism of parameter setting, IIA and IIA-4 record the PMDF approach and PMGD strategy to determine the operation probability, respectively. Table 7 summarizes the average results of 30 independent runs obtained by IIA, IIA-1, IIA-2, IIA-3 and IIA-4. It is obvious that IIA with parallel mutation mechanism and the PMDF strategy has the best mean value and standard deviation. Figures 4–6 show the comparative results by means of convergence graphs and box-and-whisker diagrams of solutions over 30 runs for tested functions. From Figures 4–6, it is apparent that IIA performs better than the other three mutation operators in terms of convergence rate and solution quality, suggesting that PM has better dynamic searching characters than any single mutation operator. Hence we can conclude that the hybridization of different single algorithm is a feasible and promising method to enrich and promote the searching performance.

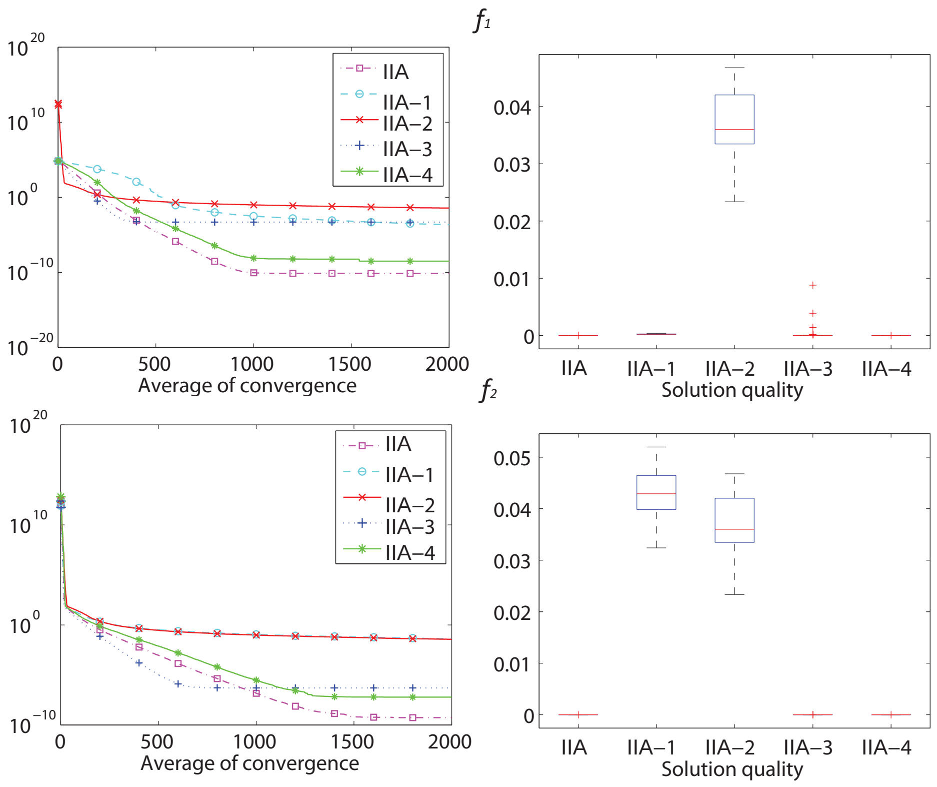

Figure 4. The comparative results of the convergence graphs and box-and-whisker diagrams of solutions during IIA (PMDF), IIA-1, IIA-2, IIA-3 and IIA-4 (PMGD) for unimodal functions f_1 and f_2.

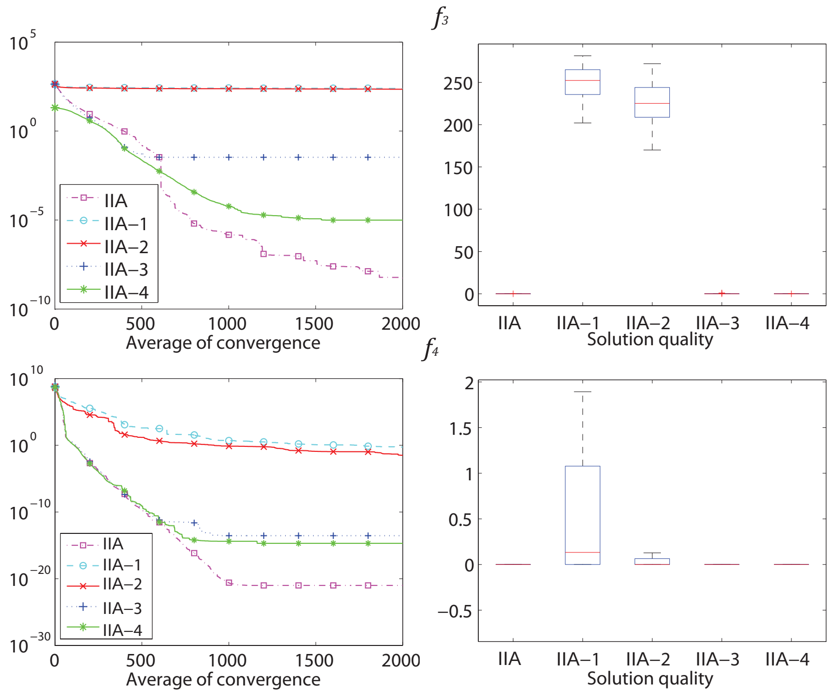

Figure 5. The comparative results of the convergence graphs and box-and-whisker diagrams of solutions during IIA (PMDF), IIA-1, IIA-2, IIA-3 and IIA-4 (PMGD) for multimodal functions with high dimension f_3 and f_4.

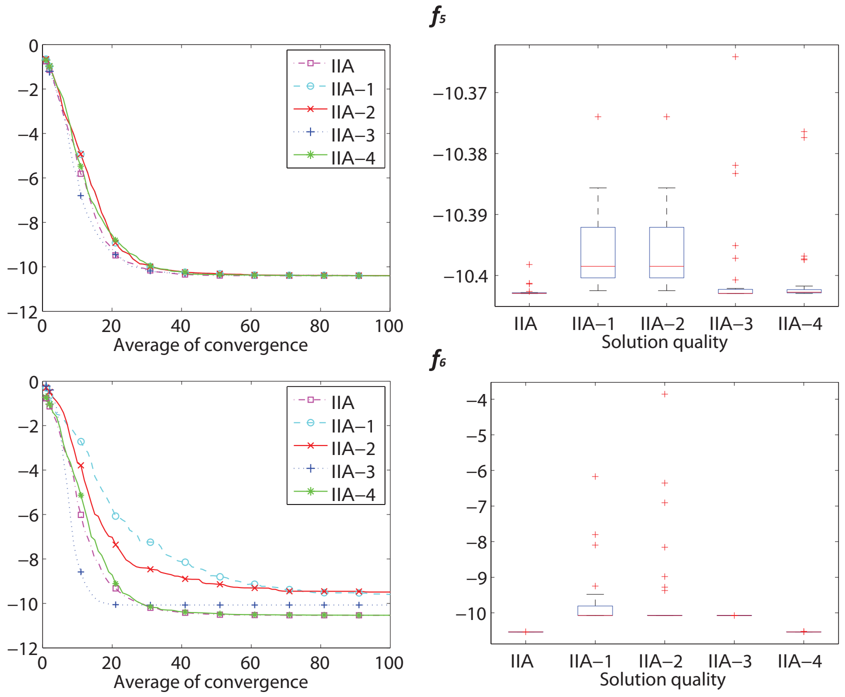

Figure 6. The comparative results of the convergence graphs and box-and-whisker diagrams of solutions during IIA (PMDF), IIA-1, IIA-2, IIA-3 and IIA-4 (PMGD) for multimodal functions with low dimension f_5 and f_6.

Table 7. The statistical results obtained by IIA (PMDF), IIA-1, IIA-2, IIA-3 and IIA-4 (PMGD) algorithms for benchmark functions under 30 runs.

	Algorithm	Mean	Std.	Median	Min	Max
f_1	**IIA**	7.05E-11	2.67E-10	5.34E-15	1.09E-24	1.42E-09
	IIA-1	2.26E-04	6.20E-05	2.19E-04	1.09E-04	3.80E-04
	IIA-2	6.79E-04	2.22E-04	6.46E-04	2.62E-04	1.20E-03
	IIA-3	4.85E-04	1.70E-03	3.74E-07	1.18E-22	8.80E-03
	IIA-4	3.09E-09	9.37E-09	4.75E-011	8.67E-19	3.75E-08
f_2	**IIA**	5.45E-10	2.97E-09	7.76E-14	3.50E-15	1.62E-08
	IIA-1	4.25E-02	4.80E-03	4.29E-02	3.24E-02	5.20E-02
	IIA-2	3.69E-02	5.80E-03	3.60E-02	2.34E-02	4.68E-02
	IIA-3	5.09E-07	2.76E-06	3.58E-06	2.22E-10	2.23E-04
	IIA-4	6.12E-08	1.82E-07	3.13E-09	4.52E-14	9.35E-07
f_3	**IIA**	5.15E-09	1.06E-08	1.82E-10	0	3.91E-08
	IIA-1	2.49E+02	2.37E+01	2.52E+02	2.02E+02	2.81E+02
	IIA-2	2.25E+02	2.64E+01	2.25E+02	1.70E+02	2.72E+02
	IIA-3	3.35E-02	1.82E-01	8.65E-11	0	9.95E-01
	IIA-4	9.90E-06	2.32E-05	1.43E-06	6.49E-11	1.04E-04
f_4	**IIA**	9.87E-22	1.95E-21	1.49E-23	3.64E-32	3.92E-21
	IIA-1	5.39E-01	9.11E-01	1.32E-01	1.00E-05	1.89E+00
	IIA-2	3.21E-02	6.39E-02	2.07E-04	8.70E-05	1.28E-01
	IIA-3	2.84E-014	2.52E-14	2.65E-14	4.81E-15	5.61E-14
	IIA-4	1.98E-15	3.83E-15	1.07E-16	1.22E-18	7.74E-15
f_5	**IIA**	-10.4029	7.63E-08	-10.4029	-10.4029	-10.4029
	IIA-1	-10.3935	8.00E-03	-10.3963	-10.4026	-10.3662
	IIA-2	-10.3953	6.40E-03	-10.3978	-10.4029	-10.3793
	IIA-3	-10.3997	8.66E-03	-10.4029	-10.4029	-10.3641
	IIA-4	-10.4004	6.66E-03	-10.4027	-10.4029	-10.3764
f_6	**IIA**	-10.5359	9.44E-04	-10.5363	-10.5364	-10.5319
	IIA-1	-9.5980	1.08E+00	-10.0750	-10.0750	-6.1722
	IIA-2	-9.4876	1.42E+00	-10.0750	-10.0750	-3.8573
	IIA-3	-10.0750	1.76E-07	-10.0750	-10.0750	-10.0750
	IIA-4	-10.5348	3.00E-03	-10.5360	-10.5364	-10.5216

As for unimodal functions, the convergence speed is more interesting than the final optimization result. In Figure 4, IIA-3 has the fastest convergence speed at the beginning among all algorithms, indicating that LM is able to use information from other antibody to speed up the convergence, but frequent information exchange causes the algorithm fall into the local optimum easily. IIA-1 and IIA-2 have slow convergence rates because CM and GM only use a random perturbation in the process of solution maturing. However, it has the ability to jump out of local optimum. IIA and IIA-4 contains the advantages of the three mutation factors, therefore it has the second and the third convergence speed at the beginning. At the same time, IIA has the fastest convergence rate and obtains the best optimization value at the end of the algorithm, in other words, IIA with PMDF approach is significantly better than IIA-4 with PMGD strategy for unimodal functions. For multimodal functions, the finial results are much more important since they reflect an algorithm's ability to escape from local optima. From Figures 5 and 6, it is evident that IIA has the best optimization results than the other three single mutation operators for all the functions. To sum up, parallel mutation operator is more effective than single mutation operator in searching better solutions and it has complementary search effects on one another no matter how inferior their own search capacities are. It is also obvious that the experiment result obtained by IIA with PMDF approach is more excellent than IIA-4 with PMGD strategy for all the multimodal functions. That is to say, the deterministic probability mechanism of PMDF is suitable in this algorithm.

Table 8. Performance comparison between IIA (PMDF) and other IAs including FCA and CLONALG in terms of solution qualities on function optimization problems.

Func.	IIA (PMDF)	FCA	CLONALG	SIA
	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
f_1	7.05E-11 ± 2.67E-10	**0 ± 0**	7.36E-08 ± 5.15E-08	2.31E-08 ± 8.97E-09
f_2	5.45E-10 ± 2.97E-09	**0 ± 0**	**0 ± 0**	6.64E-08 ± 2.42E-08
f_3	**5.15E-09 ± 1.06E-08**	1.56E-07 ± 3.12E-07	7.15E-04±1.27E-04	1.56E-03± 3.12E-04
f_4	**9.87E-22 ± 1.95E-21**	7.94E-11 ± 4.25E-12	3.27E-07 ± 8.54E-08	6.57E-03 ± 5.44E-03
f_5	**-10.4029 ± 7.63E-08**	-9.9438 ± 0.0384	-9.2691 ± 0.284	-8.9342 ± 0.8972
f_6	**-10.5359 ± 9.44E-04**	-9.9622 ± 0.0503	-9.1528 ± 0.2987	-8.6654 ± 1.1503

Table 9. Performance comparison between IIA and other bio-inspired algorithms including IIA (PMGD) and PSO in terms of solution qualities on function optimization problems.

Func.	IIA (PMDF)	IIA (PMGD)	PSO
	Mean ± Std	Mean ± Std	Mean ± Std
f_1	**7.05E-11 ± 2.67E-10**	3.09E-09 ± 9.37E-09	7.98E02 ± 6.22E02
f_2	**5.45E-10 ± 2.97E-09**	6.12E-08 ± 1.82E-07	2.21E14 ± 1.21E15
f_3	**5.15E-09 ± 1.06E-08**	9.90E-06 ± 2.32E-05	1.88E02± 4.16E01
f_4	**9.87E-22 ± 1.95E-21**	1.98E-15 ± 3.83E-15	2.21E08 ± 1.95E08
f_5	**-10.4029 ± 7.63E-08**	-10.4004 ± 6.66E-03	2.11E01 ± 0.0763
f_6	**-10.5359 ± 9.44E-04**	-10.5348 ± 3.00E-03	1.162 ± 0.1410

The results compared with the existing IAs, i.e., SIA, FCA [27], PSO and CLONALG [56] are summarized in Tables 8 and 9. Each result of IIA is obtained from 30 independent runs. It is obvious that IIA outperforms FCA, SIA, PSO and CLONALG on multimodal functions f_3–f_6. For unimodal functions f_1 and f_2, IIA is superior to SIA, but it is inferiorer than FCA. So we can conclude that IIA has advantage in solving multimodal problems. In the following sub-section, IIA is applied on LJPP.

4.2.2. Two-dimension function

Problem G_1 [59]:

$$f = x.sin(4x) + 1.1y.sin(2y) \tag{4.1}$$

$$0 < x, y > 10, \text{ global minimum: } f(0.9039, 0.8668) = -18.5547$$

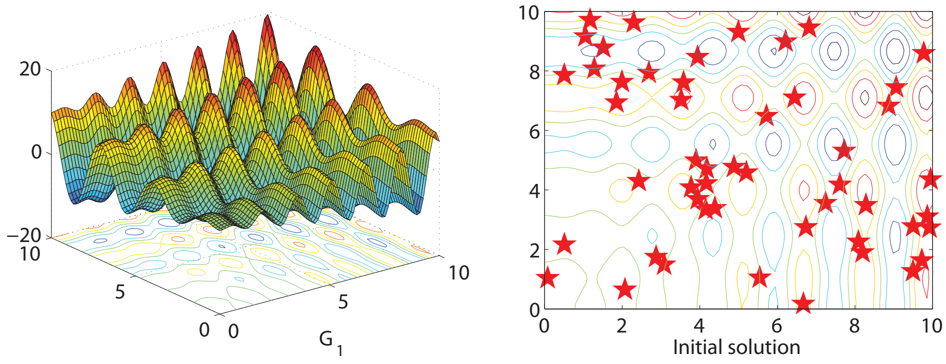

Figure 7. 3D plot of function and initial solutions for problem G_1.

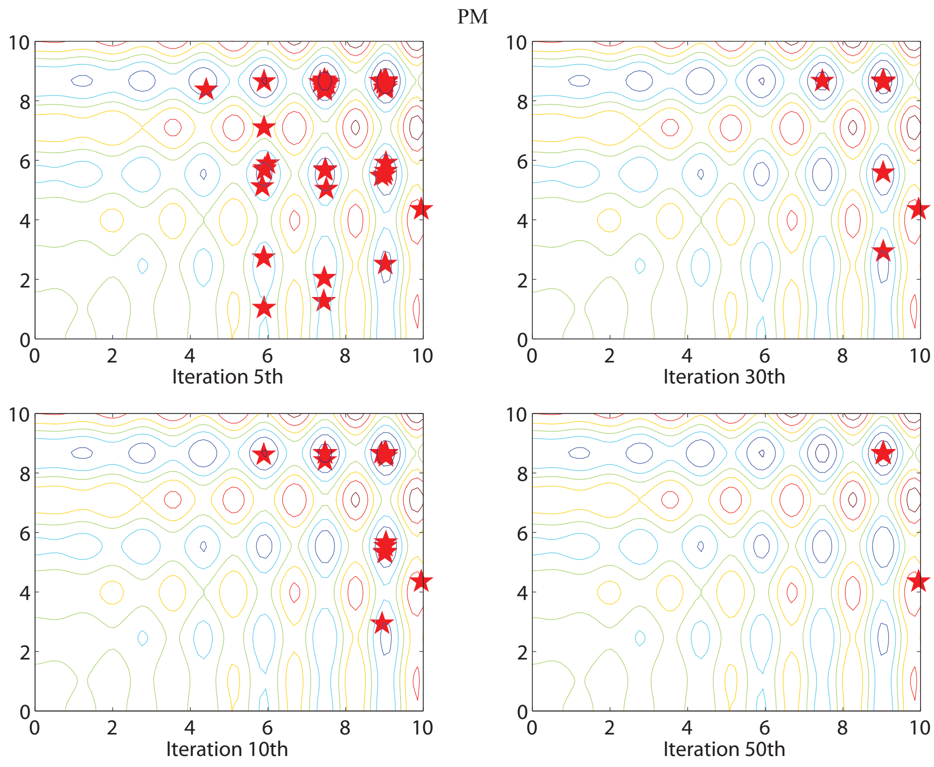

Figure 8. The distribution of solutions obtained at the iteration of the 5^{th}, 10^{th}, 30^{th} and 50^{th} by PM.

In this section, the search characteristics of the proposed IIA are verified on a two-dimension function shown in Eq (4.1). This problem is a minimization problem. Figure 7 depicts the 3D plot of the function and initial solutions for problem G_1 generated by IIA. The global minimum of this function in the interval $0 < x, y < 10$ is located at (0.9039, 0.8668) and the minimum value is -18.5547. It is visible that $G1$ is a multimodal function which has many local optima.

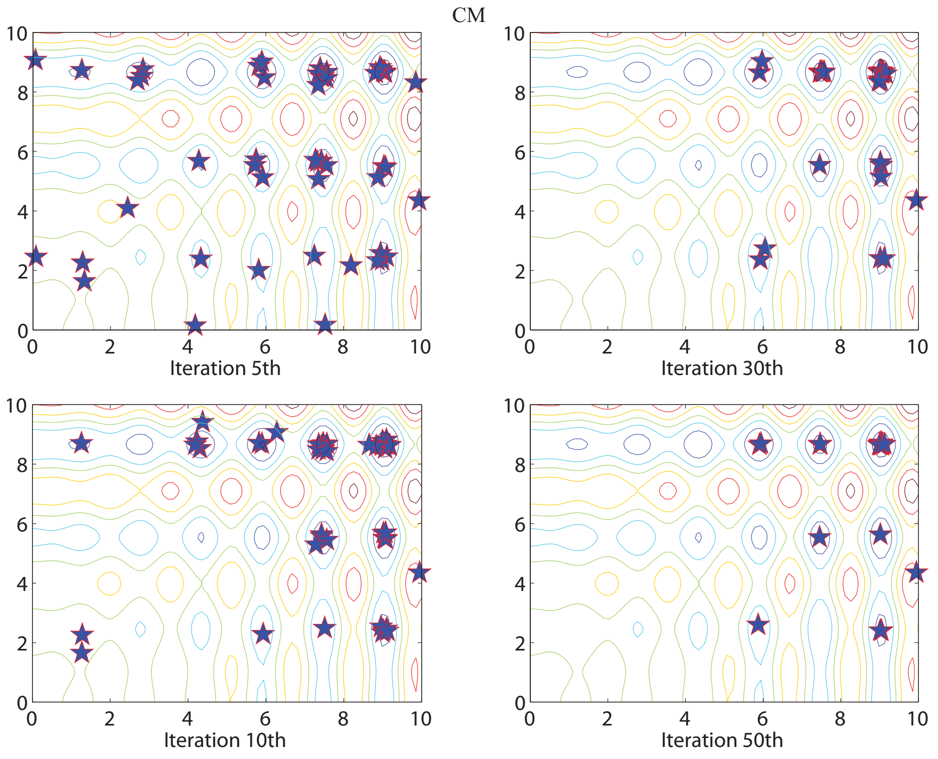

Figure 9. The distribution of solutions obtained at the iteration of the 5^{th}, 10^{th}, 30^{th} and 50^{th} by CM.

In order to identify the search characteristics of every single mutation operator and the proposed PM, a comparative experiment is conducted. The above four mutation operators (i.e., PM, CM, GM, and LM) are adopted one by one within the same algorithmic framework of IA. The same initial fifty solutions (as shown in Eq (4.1), where the star marks represent the solutions) which are randomly generated in the search space are utilized as the initial population of individuals. Figures 8–11 illustrate the distribution of solutions obtained at the iteration of the 5th, 10th, 30th and 50th by using the four mutation operators respectively. At first glance, the solutions in all figures concentrate on the nearby locations in the search space along with the iteration number's increment, indicating that all the constructed algorithms converge to the promising search areas gradually. It is apparent that PM enables the algorithm to possess the best search performance. Observing from the contour graph in Figure 8, it is clear that all solutions move to the local minima fast even at the first 5th iteration. At the 10th and 30th iteration, the convergence has been maintaining, and finally at the 50th iteration all solutions achieve the global minimum point and a local minimum point. Contrarily, the final solutions obtained by CM, GM and LM are trapped by more than one local minima. Thus, it can be concluded that PM is capable to manipulate individuals so as to find the global optima with a very greater probability and to be trapped by local minima with a smaller probability than any other mutations which used single mutation operators.

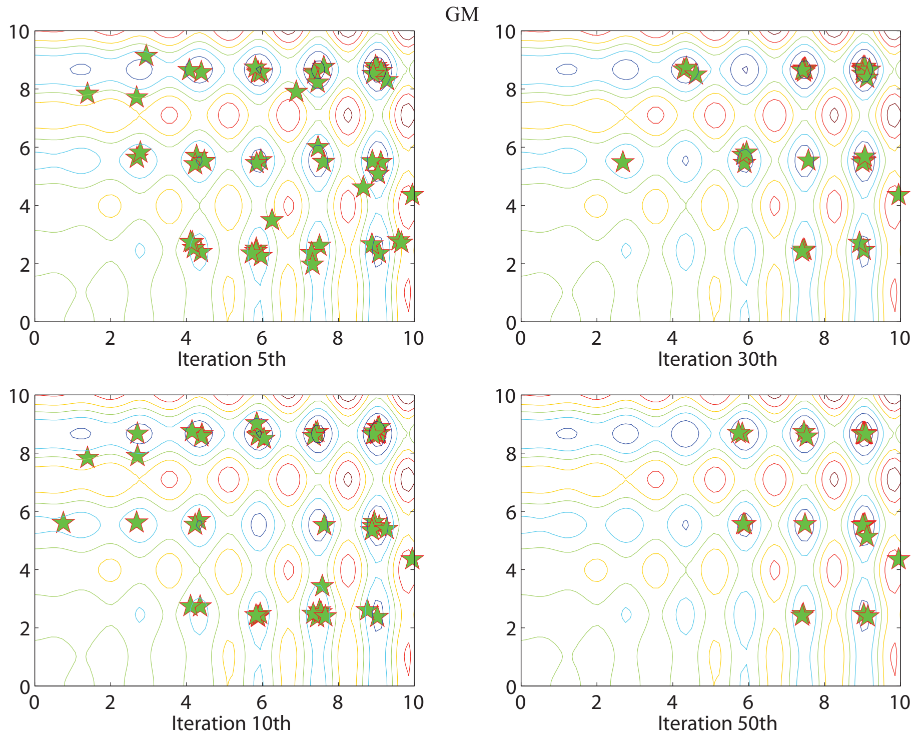

Figure 10. The distribution of solutions obtained at the iteration of of the 5th, 10th, 30th and 50th by GM.

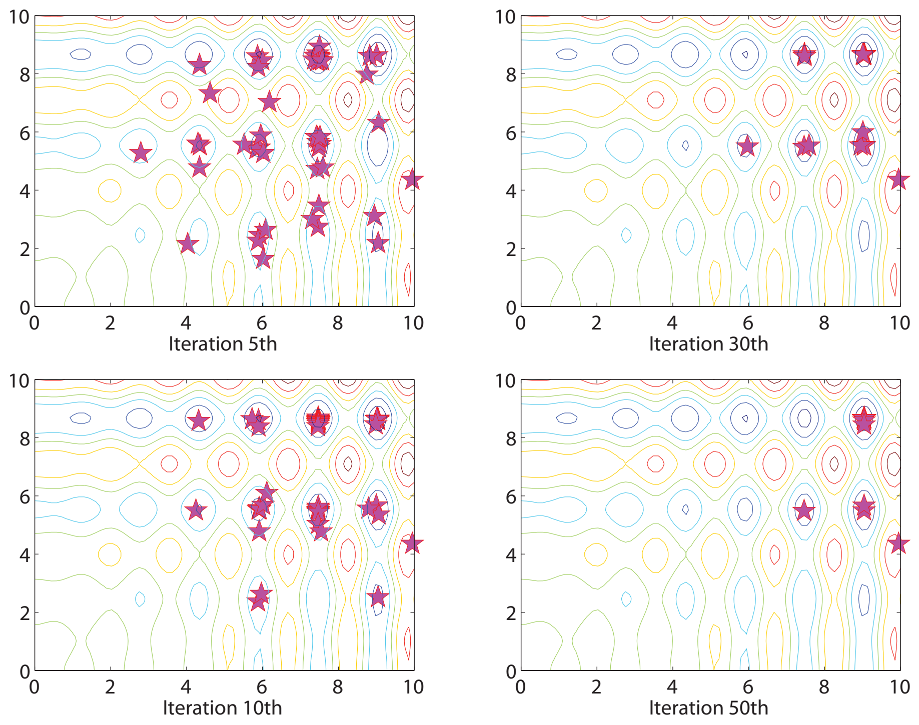

Figure 11. The distribution of solutions obtained at the iteration of of the 5th, 10th, 30th and 50th by LM.

Furthermore, the empirical results shown in Figures 9–11 support the above intuition in Section 3.1. Compared with CM and GM, LM exhibits better search performance in terms of solution quality and convergence speed, which sustains that information exchange among individuals can lead to a fast convergence. On the other hand, too much information exchange might also make the search premature, which means that LM is lack of exploration capacity to jump out of the current search region. Besides, CM's and GM's behaviors on $G1$ at the 10th iteration are quite illuminating. At the 10th iteration, both of the solutions obtained by GM and CM are widely distributed in the whole search space, which indicates their slow convergent ability. However, compared with their corresponding solutions at the 5th iteration, most individuals mutated by GM move slightly towards the local minima (about 17 clusters in the search space), while those mutated by CM make long distance jump and thus own better local minima climbing capacity (about 12 clusters).

Overall, the illustrative results of the search dynamics verify that PM which combines the three distinct search characteristics of GM, CM and LM is such a good methodology to construct the more powerful mutation operators.

4.3. Performance verification on LJPP

To further demonstrate the effectiveness and efficiency of the proposed IIA, LJPP obtained from the Cambridge Cluster Database has been used as the test suit. Firstly, the proposed IIA algorithm has been compared with the IIA-1, IIA-2, IIA-3, IIA-4 on 2, 3, 10, 38 atoms and the optimal solutions of them are already known. The optimal energy (i.e., fitness) and the maximum iteration number of the search are also set in Table 10. With the number of atoms increasing, the superiority of IIA becomes more significant which suggests IIA is more suitable for solving large scale problems. Besides, it is worth reiterating that the experimental results of IIA-1, IIA-2, IIA-3 and IIA-4 on LJPP still support our previous statements. In order to evaluate IIA more overally, the convergence graph and the box-and-whisker diagrams of solutions are also depicted for 2, 10, 38 atoms in Figure 12. It is apparent that PM outperforms the other mutation operators in terms of convergence speed and solution qualities. Due to the small data set, we have expanded the dataset and the optimal solutions of some expanded atoms are not known. However, it does not affect the performance of comparing different algorithms. We compared IIA (PMDF) with IIA (PMGD) algorithms with atomic numbers of 2, 3, 10, 15, 20, 25, 30, 38 and 50, and the experimental results are shown in the table below. The statistical results obtained on the basis of 30 independent runs are summarized in Table 11. The number of atoms is up to 50 which means the dimension of the search space is 150. The experimental results recorded are the average fitness (Mean) and the standard deviation (Std) over 30 runs. It is obvious that IIA performs significantly better than its competitors for all the tested instances consistently. IIA can find the optimal solutions for the LJPP with 2 or 3 atoms, and the Std is the smallest which indicates the robustness of the search performance.

Table 12 gives the t-test values between IIA (PMDF) and the competitors including IIA-1, IIA-2, IIA-3 and IIA-4 (PMGD). All the values are less than 0.05 which means that the average values are equal for small probability. We believe that the average values of IIA (PMDF) are significantly different from IIA-1, IIA-2, IIA-3 and IIA-4. Thus, the proposed PM has been demonstrated to be an effective methodology which combines different single mutation operators together to be a more powerful one.

Table 10. Performance comparison between IIA (PMDF), IIA-1, IIA-2, IIA-3 and IIA (PMGD) in terms of solution qualities.

Atoms	2	3	10	38
Optimal Energy	-1.000000	-3.000000	-28.422532	-173.928427
T_{max}	100	200	2000	5000
IIA (PMDF)	**-1.0000 ± 2.45E-10**	**-3.0000 ± 5.39E-07**	**-28.4150±4.28E-03**	**-161.9648 ± 5.24E-01**
IIA-1 (CM)	-1.0000 ± 2.94E-10	-3.0000 ± 2.53E-06	-28.2046 ± 3.32E-01	-154.0841 ± 3.51E+00
IIA-2 (GM)	-1.0000 ± 2.89E-10	-3.0000 ± 4.08E-06	-28.2195 ± 3.35E-01	-154.3348 ± 1.55E+00
IIA-3 (LM)	-1.0000 ± 2.96E-10	-3.0000 ± 2.81E-06	-28.3192 ± 2.55E-01	-159.6461 ± 9.80E-01
IIA (PMGD)	-1.0000 ± 2.70E-10	-3.0000 ± 1.99E-06	-28.3918 ± 3.69E-02	-160.2347 ± 9.39E-01

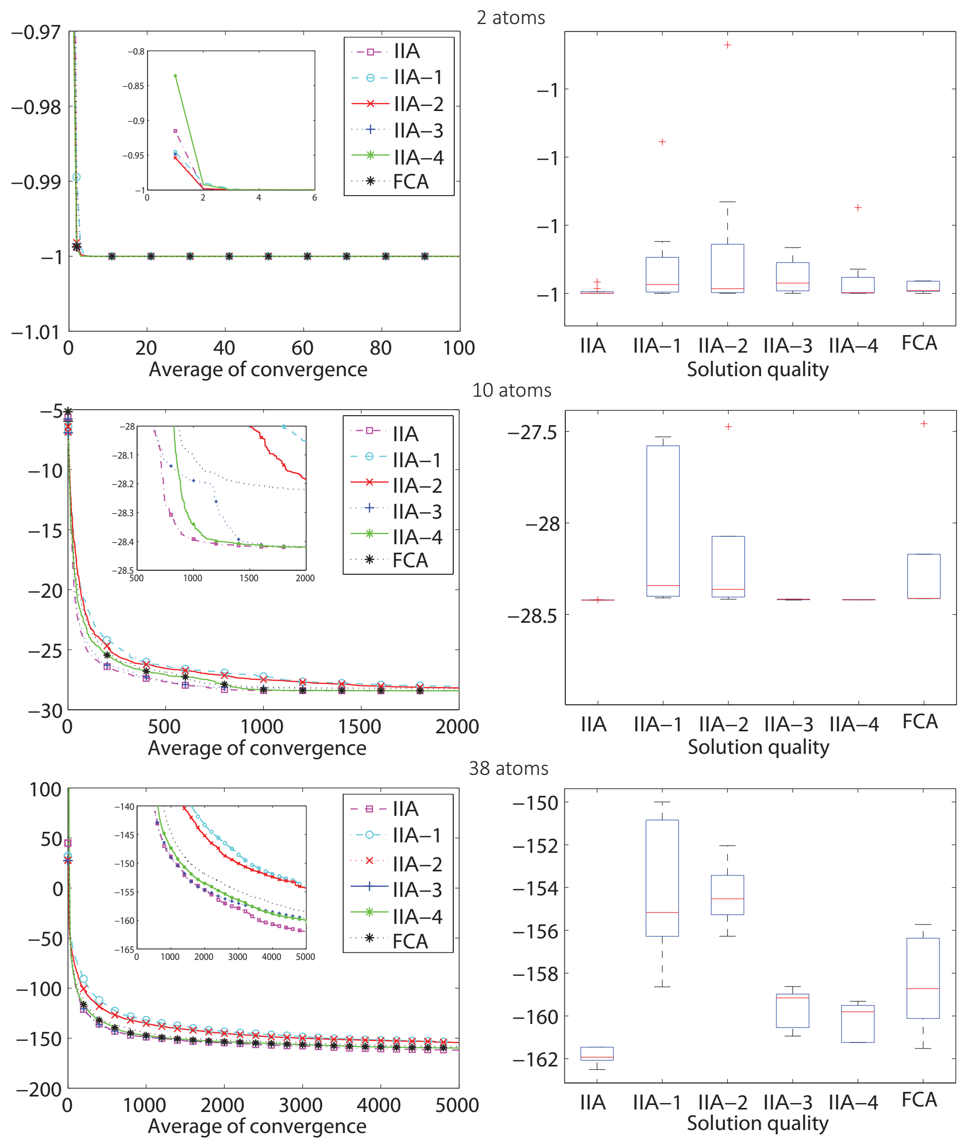

Figure 12. The convergence graphs and box-and-whisker diagrams of solutions obtained by all compared algorithms for the LJPP with 2, 10, 38 atoms.

Table 11. Performance comparison between IIA (PMDF) and IIA (PMGD) in terms of solution qualities on LJPP problems.

Atoms	IIA (PMDF)	IIA (PMGD)
	Mean ± Std	Mean ± Std
2	**-1.0000 ± 2.45E-10**	-1.0000 ± 2.70E-10
3	**-3.0000 ± 5.39E-07**	-3.0000 ± 1.99E-06
10	**-28.4150±4.28E-03**	-28.3918 ± 3.69E-02
15	**-52.3201±3.42E-03**	-51.7148 ± 5.81E-03
20	**-72.5542±2.61E-02**	-71.5932 ± 3.68E-03
25	**-95.1424±1.33E-02**	-94.8248±6.33E-02
30	**-121.5833±2.24E-01**	-120.5531±4.42E-01
38	**-161.9648 ± 5.24E-01**	-160.2347 ± 9.39E-01
50	**-220.9347± 1.75E-01**	-220.002 ± 6.74E-01

Table 12. Results of the t-test at a level of significance $\alpha = 0.05$ for IIA (PMDF) versus the competitors on LJPP.

Atoms	IIA-1 (CM)	IIA-2 (GM)	IIA-3 (LM)	IIA-4 (PMGD)
	t-test	t-test	t-test	t-test
2	4.87E-02	4.95E-02	3.83E-02	4.34E-02
3	2.31E-02	4.38E-03	3.78E-02	4.29E-02
10	3.64E-03	6.19E-03	4.06E-03	4.70E-02
38	3.56E-03	1.19E-04	4.02E-03	5.50E-03

Table 13. IIA compared to other bio-inspired optimization algorithms including GA and PSO in terms of computation time with the same iteration.

Func.	IIA	GA	PSO
Atoms			
2	**1.16E01**	2.02E-01	1.72E-01
3	**1.42E01**	1.18E-01	3.16E-01
10	**4.33E01**	5.369	3.704
15	**7.50E01**	1.716E01	1.497E01
20	**1.18E02**	2.83E01	2.48E01
25	**1.76E02**	4.169E01	3.825E01
30	**2.38E02**	5.610E01	5.317E01
38	**3.69E02**	8.557E01	8.706E01
50	**6.21E02**	1.432E02	1.452E02

Finally, Table 13 gives the computation times of the algorithm were calculated and compared for the algorithm at different numbers of atoms. Although the algorithm's computation time is slightly longer than the GA and PSO algorithms, the longitudinal comparison shows that the algorithm does not increase the time complexity of the algorithm too much as the number of atoms increases.

5. Conclusions

This paper proposes an effective immune algorithm (IIA) with parallel mutation (PM) and evaluates its performance on a two-dimension function, some benchmark functions and a number of Lennard-Jones potential instances. The experimental results show that IIA performs much better than its variant algorithms which use single mutation operators (i.e., CM, GM, LM) or a combination of CM and GM. The paper analyzes IIA in depth in terms of the user-defined parameters and the search dynamics of each incorporated mutation operator. The empirical evidences suggest that PM is an effective methodology which combines the three distinct but necessary search dynamics: 1) a fine-grained search ability derived from GM, 2) a coarse-grained search aptitude rooted in CM, and 3) a fast convergence manipulated by LM. The paper also investigates the setting for the ratios of operation probabilities. Some insights and guidelines in general have been concluded from the experimental results. The idea of PM can also be applied to the other population-based algorithms, such as evolutionary algorithms, swarm intelligence, etc. Future program on IIA will include the incorporation of more different mutation operators into PM and the self-adaption of the operation probabilities.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. A. Kumar, M. Nadeem, H. Banka, Nature inspired optimization algorithms: a comprehensive overview, *Evol. Syst.*, **14** (2023), 141–156. https://doi.org/10.1007/s12530-022-09432-6
2. K. Worden, W. J. Staszewski, J. J. Hensman, Natural computing for mechanical systems research: A tutorial overview, *Mech. Syst. Signal Process.*, **25** (2011), 4–111. https://doi.org/10.1016/j.ymssp.2010.07.013
3. S. C. Gao, Z. Tang, H. W. Dai, J. Zhang, An improved clonal algorithm and its application to traveling salesman problems, *IEICE Trans. Fundam.*, **E90-A** (2007), 2930–2938. https://doi.org/10.1093/ietfec/e90-a.12.2930
4. Y. Yang, H. Dai, S. C. Gao, Y. R. Wang, D. B. Jia, Z. Tang, Complete receptor editing operation based on quantum clonal selection algorithm for optimization problems, *IEEJ Trans. Electr. Electron. Eng.*, **14** (2018), 411–421. https://doi.org/10.1002/tee.22822
5. A. S. Muhamad, S. Deris, An artificial immune system for solving production scheduling problems: a review, *Artif. Intell. Rev.*, **39** (2013), 1–12. https://doi.org/10.1007/s10462-011-9259-1
6. F. M. Burnet, *The Clonal Selection Theory of Acquired Immunity*, Cambridge Press, 1959.

7. G. J. V. Nossal, Negative selection of lymphocytes, *Cell*, **76** (1994), 229–239. https://doi.org/10.1007/978-1-4020-6754-9-11239
8. A. Perelson, Immune network theory, *Immunol. Rev.*, **110** (1989), 5–36. https://doi.org/10.1111/j.1600-065X.1989.tb00025.x
9. P. Matzinger, The danger model: a renewed sense of self, *Science*, **296** (2002), 301–305. https://doi.org/10.1126/science.1071059
10. F. Gu, J. Greensmith, U. Aickelin, Theoretical formulation and analysis of the deterministic dendritic cell algorithm, *Biosystems*, **111** (2013), 127–135. https://doi.org/10.1016/j.biosystems.2013.01.001
11. S. C. Gao, H. W. Dai, G. Yang, Z. Tang, A novel clonal selection algorithm and its application to traveling salesman problems, *IEICE Trans. Fundam.*, **E90A** (2007), 2318–2325. https://doi.org/10.1093/ietfec/e90-a.10.2318
12. B. H. Ulutas, S. Kulturel-Konak, A review of clonal selection algorithm and its applications, *Artif. Intell. Rev.*, **36** (2011), 117–138.
13. L. N. De Castro, F. J. Von Zuben, Learning and optimization using the clonal selection principle, *IEEE Trans. Evol. Comput.*, **6** (2002), 239–251. https://doi.org/10.1109/TEVC.2002.1011539
14. R. Shang, L. Jiao, F. Liu, W. Ma, A novel immune clonal algorithm for mo problems, *IEEE Trans. Evol. Comput.*, **16** (2012), 35–50. https://doi.org/10.1109/TEVC.2010.2046328
15. Y. Ding, Z. Wang, H. Ye, Optimal control of a fractional-order HIV-immune system with memory, *IEEE Trans. Control Syst. Technol.*, **3** (2012), 763–769. https://doi.org/10.1109/TCST.2011.2153203
16. P. A. D. Castro, F. J. Von Zuben, Learning ensembles of neural networks by means of a bayesian artificial immune system, *IEEE Trans. Neural Networks*, **22** (2011), 304–316. https://doi.org/10.1109/TNN.2010.2096823
17. G. Dudek, An artificial immune system for classification with local feature selection, *IEEE Trans. Evol. Comput.*, **6** (2012), 847–860. https://doi.org/10.1109/TEVC.2011.2173580
18. M. Hunjan, G. K. Venayagamoorthy, Adaptive power system stabilizers using artificial immune system, *IEEE Symp. Artif. Life*, **2007** (2007), 440–447.
19. M. Gui, A. Pahwa, S. Das, Analysis of animal-related outages in overhead distribution systems with Wavelet decomposition and immune systems-based neural networks, *IEEE Trans. Power Syst.*, **24** (2009), 1765–1771. https://doi.org/10.1109/TPWRS.2009.2030382
20. V. Cutello, G. Nicosia, M. Pavone, J. Timmis, An immune algorithm for protein structure prediction on lattice models, *IEEE Trans. Evol. Comput.*, **11** (2007), 101–117. https://doi.org/10.1109/TEVC.2006.880328
21. V. Cutello, G. Morelli, G. Nicosia, M. Pavone, G. Scollo, On discrete models and immunological algorithms for protein structure prediction, *Nat. Comput.*, **10** (2011), 91–102. https://doi.org/10.1007/s11047-010-9196-y
22. C. Vincenzo, N. Giuseppe, P. Mario, P. Igor, Protein multiple sequence alignment by hybrid bio-inspired algorithms, *Nucleic Acids Res.*, **39** (2011), 1980–1992. https://doi.org/10.1093/nar/gkq1052

23. S. C. Gao, R. L. Wang, M. Ishii, Z. Tang, An artificial immune system with feedback mechanisms for effective handling of populationsize, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, **E93A** (2010), 532–541. https://doi.org/10.1587/transfun.E93.A.532

24. T. Luo, A clonal selection algorithm for dynamic multimodal function optimization, *Swarm Evol. Comput.*, **50** (2019), 1980–1992.

25. W. W. Zhang, W. Zhang, G. G. Yen, H. L. Jing, A cluster-based clonal selection algorithm for optimization in dynamic environment, *Swarm Evol. Comput.*, **50** (2019), 1–13. https://doi.org/10.1016/j.swevo.2018.10.005

26. H. Zhang, J. Sun, T. Liu, K. Zhang, Q. Zhang, Balancing exploration and exploitation in multiobjective evolutionary optimization, *Inf. Sci.*, **497** (2019). https://doi.org/10.1016/j.ins.2019.05.046

27. N. Khilwani, A. Prakash, R. Shankar, M. K. Tiwari, Fast clonal algorithm, *Eng. Appl. Artif. Intell.*, **21** (2008), 106–128. https://doi.org/10.1016/j.engappai.2007.01.004

28. X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.*, **3** (1999), 82–102. https://doi.org/10.1109/4235.771163

29. C. Y. Lee, X. Yao, Evolutionary programming using mutations based on the levy probability distribution, *IEEE Trans. Evol. Comput.*, **8** (2004), 1–13. https://doi.org/10.1109/TEVC.2003.816583

30. M. Gong, L. Jiao, L. Zhang, Baldwinian learning in clonal selection algorithm for optimization, *Inf. Sci.*, **180** (2010), 1218–1236. https://doi.org/10.1016/j.ins.2009.12.007

31. A. M. Whitbrook, U. Aickelin, J. M. Garibaldi, Idiotypic immune networks in mobile-robot control, *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, **37** (2007), 1581–1598. https://doi.org/10.1016/j.ins.2009.12.007

32. S. Gao, H. W. Dai, J. C. Zhang, Z. Tang, An expanded lateral interactive clonal selection algorithm and its application, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, **E91A** (2008), 2223–2231. https://doi.org/10.1093/ietfec/e91-a.8.2223

33. V. Stanovov, S. Akhmedova, E. Semenkin, Selective pressure strategy in differential evolution: Exploitation improvement in solving global optimization problems, *Swarm Evol. Comput.*, **50** 2018, 1–14. https://doi.org/10.1016/j.swevo.2018.10.014

34. R. M. Hoare, *Structure and Dynamics of Simple Microclusters*, John Wiley Sons, Inc., 2007.

35. K. Deep, M. Arya, *Minimization of Lennard-Jones Potential Using Parallel Particle Swarm Optimization Algorithm*, Springer Berlin Heidelberg, 2010.

36. M. R. Hoare, Structure and dynamics of simple microclusters, *Adv. Chem. Phys.*, (1979), 49–135.

37. J. A. Northby, Structure and binding of lennard-jones clusters, *J. Chem. Phys.*, **87** (1987), 6166–6177. https://doi.org/10.1063/1.453492

38. G. Xue, Improvement on the northby algorithm for molecular conformation: Better solutions, *J. Global Optim.*, **4** (1994), 425–440.

39. D. J. Wales, J. P. K. Doye, Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms, *J. Phys. Chem. A*, **101** (1998), 5111–5116. https://doi.org/10.1021/jp970984n

40. R. H. Leary, Global optima of lennard-jones clusters, *J. Global Optim.*, **11** (1997), 35–53. https://doi.org/10.1021/jp970984n

41. R. H. Leary, Global optimization on funneling landscapes, *J. Global Optim.*, **18** (2000), 367–383. https://doi.org/10.1023/A:1026500301312

42. D. Daven, N. Tit, J. Morris, K. Ho, Structural optimization of lennard-jones clusters by a genetic algorithm, *Chem. Phys. Lett.*, **256** (1996), 195–200. https://doi.org/10.1016/0009-2614(96)00406-X

43. B. Hartke, Efficient global geometry optimization of atomic and molecular clusters, *Eur. Phys. J. D*, **2006** (2006). https://doi.org/10.1007/0-387-30927-6-6

44. K. Deep, Shashi, V. K. Katiyar, Global optimization of lennard jones potential using newly developed real coded genetic algorithms, in *International Conference on Communication Systems and Network Technologies*, (2011), 614–618.

45. N. P. Moloi, M. M. Ali, An iterative global optimization algorithm for potential energy minimization, *Comput. Optim. Appl.*, **30** (2005), 119–132. https://doi.org/10.1007/s10589-005-4555-9

46. D. M. Deaven, K. M. Ho, Molecular geometry optimization with a genetic algorithm, *Phys. Rev. Lett.*, **75** (1995), 288–291. https://doi.org/10.1103/PhysRevLett.75.288

47. S. Darby, T. V. Mortimer-Jones, R. L. Johnston, C. Roberts, Theoretical study of cu-au nanoalloy clusters using a genetic algorithm, *J. Chem. Phys.*, **116** (2002), 1536–1550.

48. M. R. Hoare, Structure and dynamics of simple microclusters, *Adv. Chem. Phys.*, **40** (1979), 49–135. https://doi.org/10.1002/9780470142592.ch2

49. G. Xue, R. S. Maier, J. B. Rosen, Minimizing the lennard-jones potential function on a massively parallel computer, in *Proceedings of the 6th International Conference on Supercomputing*, ACM, (1992), 409–416.

50. D. Dasgupta, S. Yu, F. Nino, Recent advances in artificial immune systems: models and applications, *Appl. Soft Comput.*, **11** (2011), 1547–1587. https://doi.org/10.1016/j.asoc.2010.08.024

51. E. Hart, J. Timmis, Application areas of ais: The past, the present and the future, *Appl. Soft Comput.*, **8** (2008), 191–201. https://doi.org/10.1016/j.asoc.2006.12.004

52. V. Cutello, G. Nicosia, M. Pavone, Exploring the capability of immune algorithms: A characterization of hypermutation operators, in *Third International Conference on Artificial Immune Systems*, (2004), 263–276. https://doi.org/10.1007/978-3-540-30220-9-22

53. T. Jansen, C. Zarges, Analyzing different variants of immune inspired somatic contiguous hypermutations, *Theor. Comput. Sci.*, **412** (2011), 517–533. https://doi.org/10.1016/j.tcs.2010.09.027

54. X. Xu, J. Zhang, An improved immune evolutionary algorithm for multimodal function optimization, in *Third International Conference on Natural Computation*, (2007), 641–646.

55. X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.*, **3** (1999), 82–102. https://doi.org/10.1109/4235.771163

56. V. Cutello, G. Nicosia, M. Pavone, Real coded clonal selection algorithm for unconstrained global optimization using a hybrid inversely proportional hypermutation operator, in *Proceedings of the 2006 ACM symposium on Applied computing*, (2006), 950–954.

57. M. Crepinsek, S. H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, *ACM Comput. Surv.*, **45** (2013), 1–35. https://doi.org/10.1145/2480741.2480752

58. L. Jiao, Y. Li, M. Gong, X. Zhang, Quantum-inspired immune clonal algorithm for global optimization, *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, **38** (2008), 1234–1253. https://doi.org/10.1109/TSMCB.2008.927271

59. E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, *IEEE Congr. Evol. Comput.*, **2007** (2007), 4661–4667. https://doi.org/10.1109/4235.771163