International Journal of Computational Intelligence Systems (2023) 16:160
https://doi.org/10.1007/s44196-023-00326-2

RESEARCH ARTICLE

Brain Storm Optimization Algorithm with an Adaptive Parameter Control Strategy for Finding Multiple Optimal Solutions

Yuhui Zhang[1] · Wenhong Wei[1] · Shaohao Xie[2] · Zijia Wang[3]

Received: 18 May 2023 / Accepted: 27 August 2023

Abstract
Real-world optimization problems often have multiple optimal solutions and simultaneously finding these optimal solutions is beneficial yet challenging. Brain storm optimization (BSO) is a relatively new paradigm of swarm intelligence algorithm that has been shown to be effective in solving global optimization problems, but it has not been fully exploited for multimodal optimization problems. A simple control strategy for the step size parameter in BSO cannot meet the need of optima finding task in multimodal landscapes and can possibly be refined and optimized. In this paper, we propose an adaptive BSO (ABSO) algorithm that adaptively adjusts the step size parameter according to the quality of newly created solutions. Extensive experiments are conducted on a set of multimodal optimization problems to evaluate the performance of ABSO and the experimental results show that ABSO outperforms existing BSO algorithms and some recently developed algorithms. BSO has great potential in multimodal optimization and is expected to be useful for solving real-world optimization problems that have multiple optimal solutions.

Keywords Brain storm optimization · Adaptive parameter control strategy · Dynamic parameter control strategy · Multimodal optimization.

1 Introduction

Multimodal optimization aims at finding multiple optimal solutions of a given problem. In scientific research, engineering design, and logistics management, it is common to encounter problems with multiple optimal solutions [1–3]. Finding all the optimal solutions of these problems can bring benefits to the problem owners. First, they can have a better understanding of the problem by inspecting the common structure of the optimal solutions. Second, alternative solutions are available when the deployed solution failed due to some physical restrictions. For example, in logistics, the planned path may be blocked due to traffic accidents or road maintenance. In this case, we can switch to other paths if there are multiple optimal paths available [4].

The purpose of this study is to develop an efficient algorithm for multimodal optimization through the design of an adaptive parameter control strategy. It is worth noting that finding multiple optimal solutions to a problem can be challenging. Traditional single point based iterative algorithms require multiple runs, each with a different starting point, which can be time-consuming and inefficient. In contrast, population-based search algorithms may be more suitable for the task of simultaneously locating multiple solutions. Population-based search algorithms can be divided into two categories: evolutionary algorithms (EAs) and swarm intelligence (SI) algorithms. EAs are inspired by the evolution of life, while SI algorithms are inspired by the collective behavior of social animals. Some famous paradigms of EA include genetic algorithm (GA) [5, 6], genetic programming (GP) [7], differential evolution (DE) [8, 9], and evolution strategy (ES) [10, 11]. The SI category includes algorithms like particle swarm optimization (PSO) [12, 13], ant colony optimization (ACO) [14, 15], and artificial bee colony (ABC) [16, 17], just to name a few. The fundamental principle of EA and SI algorithms is to effectively solve problems through

✉ Yuhui Zhang
yhzhang@dgut.edu.cn

1 School of Computer Sciences and Technology, Dongguan University of Technology, Dongguan, Guangdong, China

2 School of Electronic Information, Shantou Polytechnic, Shantou, Guangdong, China

3 School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, Guangdong, China

Published online: 26 September 2023

information sharing. This is often achieved by the cooperation and/or competition between individuals. EA and SI have found wide spread applications in real-world scenarios [18]. The population-based search mechanism endows EA and SI with the potential to locate all the optima of a problem in a single run. It is possible to divide the whole population into several subpopulations and assign these subpopulations to different search areas for different optima. To facilitate the division, maintenance, and update of subpopulations, various niching techniques have been developed [1]. They help stabilize the search process so that the subpopulations can move steadily toward their targets. With the assist of niching techniques, population-based algorithms have become the mainstream approach for multimodal optimization.

Existing SI paradigms are generally inspired by the collective behavior of animals, birds, fishes, or insects. However, it is promising to develop more effective optimization algorithms by taking inspiration from human activities. After all, humans are the most intelligent living beings on earth. Motivated by this, Shi [19] proposed a novel brain storm optimization (BSO) algorithm that mimics the human brainstorming process. In recent years, BSO has been shown to be effective in solving various types of optimization problems [20, 21]. However, the power of BSO in solving multimodal problems have not been fully exploited. In BSO, new candidate solutions (new ideas) are produced by a creating operator, which adds random perturbations to existing solutions. The scale of perturbation is control by a parameter that changes dynamically with the number of iterations. In the context of multimodal optimization, it is common to observe that different subpopulations are in different evolutionary states and require different step size parameters. Therefore, the performance of BSO can be further improved by incorporating an adaptive parameter control strategy. Motivated by the above finding, in this paper, we develop an adaptive brain storm optimization (ABSO) algorithm that makes modifications to the step size parameter according to the feedback of the creating operator during the entire search process. In the primitive BSO, two levels of randomness (one uniform random value and one Gaussian random value) are added to the step size when producing new candidate solutions, which makes the adaptation of control parameters very difficult. Because it is hard to decide whether generating better solutions is credited to the random noise or to the correct setting of the control parameter. To tackle the challenge, we devise a simplified version the of the creating operator, in which the randomness only comes from a standard Gaussian distribution. In addition, an adaptive parameter control strategy is developed to adjust the step size parameter. Every individual has its own step size sampled from a Gaussian distribution with mean μ_k, where μ_k is a meta-parameter. To capture the landscape information and meet the requirement of multiple optima finding, the meta-parameter μ_k is constantly updated according to the success history of the creating operator. In addition to the adaptive parameter control strategy, we develop a systematic approach to dynamic strategy generation. The proposed approach can create convex, linear, and concave dynamics by making small changes to the configuration. Three dynamic control strategies derived from the approach are investigated in this paper. Comprehensive experiments have been conducted on benchmark problems to examine the effectiveness of the proposed strategies. The experimental results confirm that the adaptive control strategy can find suitable parameter settings for different types of multimodal problems. With the assist of the adaptive control strategy, ABSO compares favorably with several recently developed algorithms.

The remainder of this paper is organized as follows. Section 2 briefly reviews some classical niching techniques and some recently developed multimodal optimization algorithms. Then, the fundamental principle and detailed procedures of the BSO algorithm are presented. Section 3 is devoted to the description of the proposed ABSO with an adaptive parameter control strategy. Experiments are conducted in Sect. 4 to comprehensively examine the performance of the proposed algorithm. Finally, Sect. 5 concludes this paper and points out some future research directions.

2 Background

In this section, we first review several popular niching techniques which are used to induce multiple convergence behavior in population-based algorithms. A short survey on some recently developed multimodal optimization algorithms is provided subsequently. Then, the BSO algorithm is described in detail. Finally, a variant of BSO specifically designed for multimodal optimization is introduced.

2.1 Niching Techniques

Note that population-based search algorithms are designed for global optimization. With global selection operators, all the individuals will eventually gather around a single optimal/suboptimal point. This tendency is against the intention of multimodal optimization, in which the goal is to simultaneously find multiple solutions. To induce multiple convergence behavior in population-based algorithms, researchers have come up with many ideas to guide the individuals evolve toward different optima. The developed techniques are commonly referred to as "niching". The fundamental principle of niching techniques is to divide the population into subpopulations according to some neighborhood concepts and competitions between individuals are only allowed within neighborhoods [22, 23]. In this way, we

can avoid genetic drift phenomenon and maintain population diversity over the entire search process.

There are several classical niching techniques developed in the early stage of multimodal optimization research. The most prominent ones include fitness sharing [24], crowding [25], clearing [26], and speciation [27]. Fitness sharing modifies the fitness of an individual by taking account the population density within a circle (hypersphere in a high dimensional space). Crowding poses a restriction on the competition between offspring and parents. The newly produced offspring is only allowed to compete with its neighboring solutions. Clearing is more aggressive than crowding. For each niche, clearing removes all the individuals except the best one. Speciation introduces a procedure to divide the population into species. Each specie evolves separately with its own reproduction and replacement operators. The advantage of the classical niching techniques is that they are conceptually simple and easy to implement. However, their effectiveness is influenced by the niching parameters, which are difficult to determined without a priori knowledge of the problem being handled.

The classical techniques often serve as basic building blocks for more advanced techniques. Qu et al. [28] developed a neighborhood mutation operator for DE and combines it with fitness sharing, crowding, and speciation. In the neighborhood mutation operator, when producing an offspring solution for the ith individual $\mathbf{X}_i$, three individuals selected from the neighborhood of $\mathbf{X}_i$ are combined using the DE/rand/1 strategy. Gao et al. [29] applied a clustering technique to divide the population into subpopulations. In addition, a self-adaptation strategy is developed to automatically adjust the control parameters of DE. The clustering technique and the self-adaptation strategy are then integrated with crowding DE (CDE) and species-based DE (SDE). The neighborhood mutation operator and the self-clustering approach greatly enhance the local convergence capability of DE, but we need to first determine the neighborhood size and the cluster size. Epitropakis et al. [30] designed a dynamic archive technique to store found optima and adopts an adaptive parameter control strategy to increase search efficiency of DE. The introduction of dynamic archive can avoid the risk of mistakenly replacing potential optima. However, the management of the archive is not an easy task. Biswas et al. [31] proposed a local information sharing mechanism for inducing niching behavior. The mechanism is then integrated with CDE and SDE. A parent-centric mutation operator with normalized neighborhood is developed in [32]. The new operator is combined with synchronous crowding replacement rule to achieve stable niching behavior. The information sharing mechanism and parent-centric mutation operator can induce efficient niching behavior, but they involves a number of manually determined parameters. Traditional niching techniques require a large amount of distance calculation, which significantly increase the time complexity of EAs. To alleviate the problem, Zhang et al. [33] developed a fast niching technique based on locality sensitive hashing. A faster version of NCDE (termed Fast-NCDE) is developed by integrating the new technique with NCDE. Ma et al. [34] developed an improved ABC (IABC) algorithm that incorporates the crowding selection technique. Moreover, two search mechanisms are designed to help enhance population diversity and better explore the search space.

More recently, many new techniques have been developed to tackle the challenges posed by multimodal optimization. To fully utilize the historical search information, Huang et al. [35] built a binary space partition (BSP) tree to structurally organize the space visiting information. A probabilistic niching strategy is defined to reinforce exploration and exploitation by utilizing the structurally organized information.The use of a BSP tree can greatly enhance the search efficiency, at the cost of increasing the algorithm's time complexity. Wang et al. [36] proposed an automatic niching technique based on affinity propagation clustering (APC). In addition, a contour prediction approach (CPA) and a two-level local search strategy (TLLS) are developed to accelerate convergence speed and increase solution accuracy. The APC method alleviates the need of specifying the number of clusters, but the CPA may not be suitable for the high-dimensional problems. A distributed individuals for multiple peaks (DIMP) framework is designed in [37]. A virtual population controlled by an adaptive range adjustment strategy is derived from each candidate solution to explore the search space. A lifetime mechanism and an elite learning mechanism (ELM) are embedded into the framework to increase population diversity and solution accuracy. Zhao et al. [38] proposed a local binary pattern (LBP) based adaptive DE (LBPADE) to efficiently solve multimodal optimization problems. A niching and global interaction (NGI) mutation operator is incorporated into LBPADE for effective exploration. Although the DIMP framework and the LBP mechanism are able enhance the performance of DE in finding multiple optima, they involves a number of new parameters that need to be fine-tune through trial-and-error. Sheng et al. [39] developed a niching memetic DE that adopts a niching competition strategy and a supporting archive strategy. The niching competition strategy is designed to encourage high potential niches for exploitation and low potential niches for exploration, while the supporting archive strategy is designed to help maintain potential optima and facilitate population evolution. The new strategy can help properly search the multimodal landscape, but it did not taken into account the niche size in its design. A double-layer-clustering speciation DE (DLCSDE) is developed in [40] to tackle multimodal optimization problems. The first level clustering is used to divide the whole population into subpopulations. The second level clustering groups the species seed in each subpopulation to perform

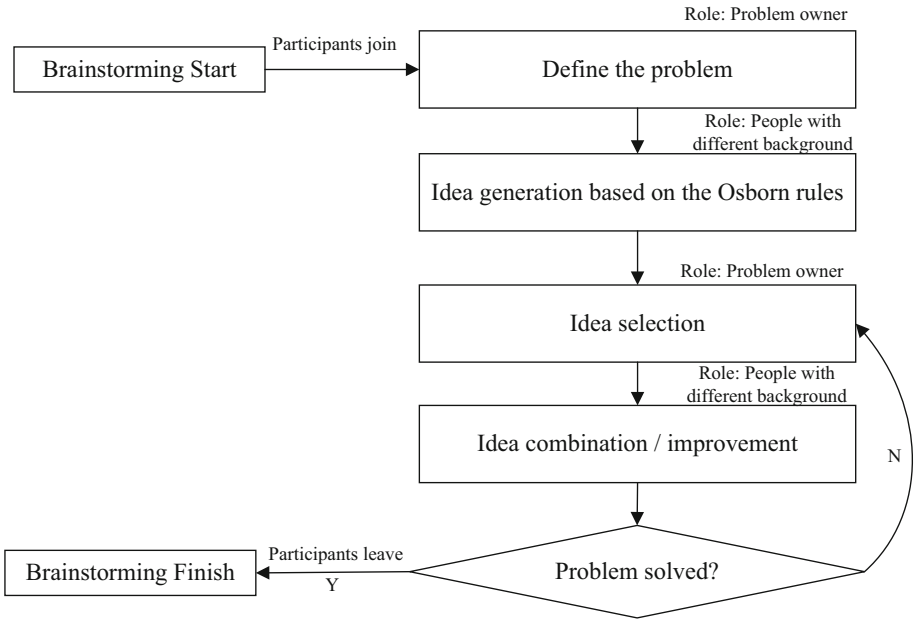

Fig. 1 Basic steps of brainstorming process

global search. The double-layer-clustering method is able to increase population diversity, but it inherits the sensitive parameters from the speciation method. Ahrari et al. proposed [41] a niching technique based on the concept of repelling subpopulation. Offspring of weaker subpopulations are kept some distance away from those of fitter subpopulations. The niching technique is incorporated into state-of-the-art evolution strategies to develop competitive multimodal optimization algorithms. The repelling method is very effective in distributing subpopulations to search for different optima. However, the definition of taboo regions in the repelling method requires additional parameters. A niching gray wolf optimizer (NGWO) that incorporates the personal best feature of PSO and a local search technique is proposed in [42]. The two new features help NGWO maintain a good balance between exploitation and exploration for solving multimodal optimization problems. NGWO achieves promising results on the benchmark problems. However, it may encounter difficulties when solving problems with irregular regions of attraction, since the fitness Euclidean-distance ratio is adopted in NGWO.

2.2 Brainstorm Optimization Algorithm

Most of the population-based search algorithms are inspired by the collaborative behavior of social animals. Since human being is the most intelligent and communicative being on earth, it is likely that algorithms inspired by human activities will have an edge over those inspired by other mechanisms. Driven by this motivation, Shi developed a BSO algorithm that mimics the problem-solving process of human beings [19]. When facing a difficult problem that cannot be solved by a single person, it is often helpful to gather a group of persons with different background to brainstorm. Great and unexpected solution can occur through collaboration of persons. Brainstorming is developed and systemized by Osborn and has been widely used in academia and industry for increasing creativity. There are four Osborn's rules to obey during the brainstorming process: (1) suspend judgment of ideas, (2) welcome unusual ideas, (3) combine and improve ideas, (4) the more ideas, the better. Based on the four rules, the basic steps of a brainstorming process are described in Fig. 1.

The detailed procedures of the brainstorming process are as follows. After defining the problem being solved, we gather a diverse group of people who can contribute different perspectives to the brainstorming session. Following ground rules like "no criticism", "quantity over quality", and "build on each other's ideas", the brainstorming session starts by encouraging participants to create and share their ideas. The purpose is to create as many ideas as possible. This is because the more ideas generated, the larger chance of finding high quality solutions. Then, the problem owner reviews the ideas and identify the most promising ones. Subsequently, new ideas are created by combining or improving the selected ideas. This helps to spark new ideas and generate more creative solutions. If the new solution solves the problem, then the brainstorming session ends. Otherwise, we start a new brainstorming session to find better ideas.

The procedures of BSO algorithm are given in Algorithm 1. Given a problem with D decision variables, BSO first generate a set of N initial ideas $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N$. Each idea $\mathbf{X}_i$ is a D-dimensional vector $[x_1, x_2, ..., x_D]$ representing a candidate solution. During the optimization process, the ideas are improved iteration by iteration through grouping, replac-

ing, and creating operators. In the grouping operator, N ideas are divided into M clusters by the k-means algorithm. The replacing operator randomly replaces cluster centers with newly generated ideas. In the creating operator, N new ideas are created one by one based on the current ideas. Each new idea is generated by using one or two clusters. After the cluster has been selected, new ideas are derived from the cluster center or random cluster members. The three operators are applied repeatedly until the termination criterion is met.

In the literature, BSO has been extended and enhanced to solve various optimization problems. El-Abd [43] developed a global-best BSO (GBSO) that uses the global best solution to guide the update of other solutions. Although a re-initialization scheme is used in GBSO, it may still be easy to get in local optima. Zhao et al. [44] designed four mutation strategies to improve the search capability of BSO and used a Q-learning mechanism to guide the selection of strategies. Noticing the importance of population diversity in algorithm design, Yu et al. [45] implemented two diversity measures to facilitate the adaptation of mutation strategies. Zhou et al. [46] developed a modified BSO algorithm with the step-size parameter adapts to the search ranges of each dimension. The defect of the approach is that it also introduces two new parameters. Although BSO have been improved in various aspects, not much work has been conducted to improve the solution update formula of BSO. In [47], Cheng et al. provided a comprehensive survey of BSO, which points out the recent developments of the BSO algorithms.

Algorithm 1 BSO

```
Randomly generate a set of N initial ideas X_1, X_2, ..., X_N and evaluate their fitness;
while termination criterion is not satisfied do
    Divide N ideas into M clusters C_1, C_2, ..., C_M using k-means;
    Record the best idea in each cluster as cluster center;
    if rand(0, 1) < p_replace then ▷ Probability of replacing a cluster center
        Randomly select a cluster center and replace the cluster center with a random idea;
    end if
    for i = 1 to N do
        if rand(0, 1) < p_one then ▷ Probability of generating a new idea based on a single cluster
            Randomly select a cluster C_j with probability p_j;
            if rand(0, 1) < p_onecenter then ▷ Probability of using the cluster center to generate a new idea
                Add a random value to the cluster center of C_j to generate a new idea Y_i;
            else
                Add a random value to a random idea of the selected cluster C_j to generate a new idea Y_i;
            end if
        else ▷ Probability of generating a new idea based on two clusters
            Randomly select two clusters C_j1 and C_j2;
            if rand(0, 1) < p_twocenter then ▷ Probability of using the cluster centers of two selected clusters to generate a new idea
                Combine the cluster centers of C_j1 and C_j2 to generate a new idea Y_i;
            else
                Combine two ideas randomly selected from C_j1 and C_j2 to generate a new idea Y_i;
            end if
        end if
        Evaluate Y_i and replace X_i with Y_i if Y_i is better;
    end for
end while
```

2.3 BSO for Multimodal Optimization

BSO is not specifically designed for multimodal optimization. To locate multiple optima in a single run, Dai et al. [48] developed an optima-identified framework (OIF) and integrated it with BSO. The resulting algorithm is named OIF-BSO. OIF-BSO makes four changes to BSO so that it can meet the requirement of multimodal optimization. The first change is the clustering technique, instead of k-means, a max-fitness clustering method (MCM) is used to divide the population into clusters. Second, instead of randomly replacing cluster centers with random new ideas, a modified disruption strategy (MDS) is used to classify the clusters into two categories, say, A and B. Cluster centers of category A are potential optima and will not be replaced by randomly generated new ideas. The third change is that new redistribution strategy is devised to create ideas. Considering that clusters in category A contain many redundant solutions (i.e., solutions around the identified potential optima), computational resources are allocated to clusters in category B to avoid unnecessary cost. The last change is that roulette wheel selection is used to pick random ideas from clusters in category B and crowding technique is used to restrict the comparison between newly generated ideas and existing ideas. Experimental results presented in [48] show that OIF-BSO is quite effective in solving benchmark multimodal optimization problems.

BSO is a relatively new meta-heuristic algorithm whose performance in multimodal optimization has not been fully exploited. Different from other meta-heuristic algorithms, BSO is developed by mimicking the brainstorming process of human being, which is the most intelligent being on earth. It is expected that BSO can achieve better performance than algorithms inspired by other metaphor. This paper intends to exploit the potential of BSO in solving multimodal optimization problems. Based on this consideration, we developed a BSO algorithm that incorporates a niching technique and an adaptive parameter control strategy, with the purpose of providing an effective off-the-shelf tool for solving various multimodal optimization problems.

3 Adaptive BSO Algorithm for Multimodal Optimization

In this section, we first describe the motivation of introducing adaptive parameter control strategy into BSO. Then, a success history-based adaptive control strategy is developed. In addition, we introduce a systematic approach to building dynamic control strategies. Three dynamic strategies derived from the systematic approach are investigated in the later experimental section. Finally, we present an adaptive BSO algorithm by integrating the proposed adaptive parameter control strategy with BSO.

3.1 Motivation of Adaptive BSO

In BSO, new ideas (candidate solutions) are derived from existing ones, which are selected from one or two clusters. More specifically, there are two branches of creating new ideas. One branch is to generate a new idea by adding a random noise to an existing one. Suppose that $\mathbf{X} = [x_1, x_2, ..., x_D]$ is the selected idea for reproduction, a new idea $\mathbf{Y} = [y_1, y_2, ..., y_D]$ is generated in the following manner:

$$y_i = x_i + \xi_i \times n_i \tag{1}$$

where n_i is a random value drawn from a Gaussian distribution with mean μ and variance σ. ξ_i is a coefficient that rescales the random value. It is computed as follows:

$$\xi_i = \text{sigmoid}((0.5 \times \text{gen} - \text{MaxGen})/k) \times \text{random}(0, 1) \tag{2}$$

where gen is the current number of generations and MaxGen is the maximum number of generations. k is a control parameter of BSO that changes the slope of the linear input. The sigmoid function is a transfer function that maps the input into range (0, 1). The function random(0, 1) returns a random uniform number within (0, 1).

The second branch of generating a new idea by combing two existing ideas. Suppose that $\mathbf{X}_1$ and $\mathbf{X}_2$ are the selected ideas for reproduction, a combined idea $\mathbf{X}$ is first generated as follows:

$$\mathbf{X} = r\mathbf{X}_1 + (1 - r)\mathbf{X}_2 \tag{3}$$

where r is a random real value within the interval [0, 1]. After generating $\mathbf{X}$, formula (1) is used to create a new idea $\mathbf{Y}$. Therefore, the core of new idea generation lies in formula (1). It can be observed that there are two levels of randomness. The first level of randomness comes from the computation of ξ_i and the second level of randomness comes from the Gaussian random number.

When creating new ideas, there is a control parameter k that influences the distance between the new idea and the old idea. In multimodal optimization, different problems may have different regions of attractions. It is possible that these optimal regions are of different shapes and sizes. Some problems may require small step sizes while the others require large step sizes. A single parameter setting cannot meet the requirement of multiple optima finding task. Therefore, we need a way to adjust the step size parameter.

3.2 Adaptive and Dynamic Adjustment of the Step Size Parameter

There are two challenges when adjusting the parameter k. The first challenge is that the effect of k on the step size is indirect. The parameter k will change the slope of the linear input of the sigmoid function, which will further change the value of ξ_i. Then, ξ_i is multiplied by another random Gaussian value n_i and added to the existing candidate solution. The product of ξ_i and n_i gives the final perturbation to create new ideas. The second challenge is that there are two levels of randomness in creating new ideas. This makes the credit assignment process very difficult when performing parameter adaptation. It is hard to tell the creation of a new successful idea is due to the randomness or due to the correct setting of the parameter k.

To overcome the challenges, we introduce a new creating operator that involves a direct control parameter to adjust the step size. There is only one level of randomness. Specifically, given a reference idea, a new idea is generated in the following manner:

$$y_i = x_i + \text{sigmoid}(k) \times n_i \tag{4}$$

where k is the step size parameter and n_i is a Gaussian random value with mean 0 and variance 1. Different from the original creating operator shown in (1), the only randomness comes from the random Gaussian value n_i. In addition, the input of the sigmoid function is the control parameter itself, not influenced by the current number of generations. This simplification makes it much easy to adjust the control parameter based on the feedback information collected in the optimization process.

Note that the suitable parameter setting varies from problem to problem. To increase the search efficiency and robustness, we devise a success history-based parameter adaptation scheme. Instead of having only one global parameter setting, each candidate solution $\mathbf{X}_i$ has its associated control parameter k_i. We assume that they are produced by a hidden variable μ_k. The hidden variable is adjusted iteration by iteration using the success history of creating better ideas. The detailed procedures are as follows. At the beginning of BSO, the hidden variable μ_k is initialized to 0. The parameter k_i for the ith candidate solution is drawn from a Gaussian distribution with mean μ_k and variance 0.8. During the opti-

mization process, a memory $\mathbf{M}$ is used to store the 'correct' parameter settings. Suppose that the candidate solution $\mathbf{X}_i$ is selected to create a new idea $\mathbf{Y}_i$, then the parameter k_i associated with $\mathbf{X}_i$ will be applied in formula (4). After $\mathbf{Y}_i$ has been created, we evaluate its fitness and compare it with its nearest neighbor $\mathbf{X}_{nn}$ in the current candidate set of ideas. If $\mathbf{Y}_i$ is better than its nearest neighbor $\mathbf{X}_{nn}$, $\mathbf{X}_{nn}$ will be replaced by $\mathbf{Y}_i$ and the value k_i is appended to the memory $\mathbf{M}$.

After N new ideas have been created, BSO enters the next iteration until the predefined termination criterion is met. Before creating next set of N ideas, we first update the hidden variable μ_k. The parameter k_i associated with each candidate solution is resampled. Specifically, we use the 'correct' samples in memory $\mathbf{M}$ to update μ_k.

$$\mu_k = 0.9\mu_k + 0.1\frac{1}{|\mathbf{M}|}\sum_{k_i \in \mathbf{M}} k_i \tag{5}$$

The notation $|\mathbf{M}|$ denotes the number of samples in memory $\mathbf{M}$.

In addition to the adaptive control strategy, we introduce a systematic way to produce dynamic control strategies. In a dynamic control strategy, the parameter k is changed along with the number of fitness evaluations (FEs). We can produce a variable p that dynamically changes from 1 to 0 as FEs increases by using the following formula:

$$p = \left(\left(1 - \frac{\text{FEs}}{\text{MaxFEs}}\right)^u\right)^v \tag{6}$$

where MaxFEs is the predefined maximum number of fitness evaluations. u and v are user defined parameters used to change the convexity of the function plot. Figure 2 shows examples of three different settings, in which u and v are set to (1, 5), (1, 1), and (5, 1) respectively. By using different values of u and v, we can push the function plot toward upper left or lower right corner. Since p is in the range [0, 1], it is straightforward to rescale the range by using the following formula:

$$k = \text{LB} + p \times (\text{UB} - \text{LB})\,. \tag{7}$$

In this way, the control parameter k is dynamically decreased from UB to LB as the number of fitness evaluation increases.

3.3 Adaptive and Dynamic BSO

The adaptive parameter control strategy is combined with BSO to increase its search efficiency in solving multimodal optimization problems. The pseudo code of ABSO is presented in Algorithm 2. Compared with BSO, there are several major changes. Since the cluster centers are potential optimal solutions, the replacing operator may mistakenly replace them with random new ideas, leading to unexpected loss of found optima. Therefore, the replacing operator is removed in ABSO. Considering that the k-means algorithm is very time-consuming, the MCM developed in [48] is employed to divide the candidate solutions into clusters. Another major change is that after selecting one or two clusters for the creating operator, formula (4) is adopted to create new ideas, with the control parameter k adaptively adjusted in the optimization process. Finally, to increase the diversity maintenance capability, the crowding selection technique is used to restrict the comparison of the candidate solutions. The newly created ideas are compared with their nearest neighbors in the current set of candidate solutions. The dynamic versions of BSO (DBSO) are the same as ABSO, except that they use dynamic parameter control strategies.

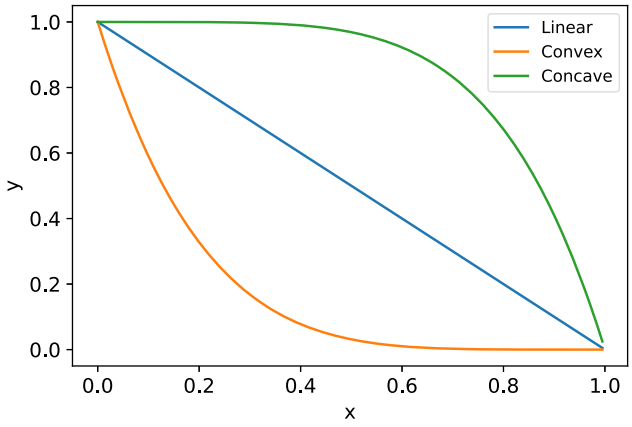

Fig. 2 Dynamic control strategies with convex, linear, and concave shapes

To illustrate the detailed steps of ABSO, a concrete example is presented in Fig. 3. Suppose that we are solving the equal maxima problem with the expression $f(x) = sin^6(5\pi x)$. The candidate set contains four solutions $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$ and $\mathbf{X}_4$. They are divided into two clusters (C_1 and C_2). We generate new solutions $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$, and $\mathbf{Y}_4$ according to the pseudo code shown in Algorithm 2. Subsequently, the crowding selection is conducted to update the candidate solution set and the 'correct' parameter value is stored in memory $\mathbf{M}$. Then, we update μ_k and enter next iteration if the termination criterion is not satisfied.

4 Experimental Study

In this section, we carry out experiments to evaluate the proposed approach. The performance of ABSO is examined on a set of benchmark problems and compared with several classic algorithms, as well as some recently developed

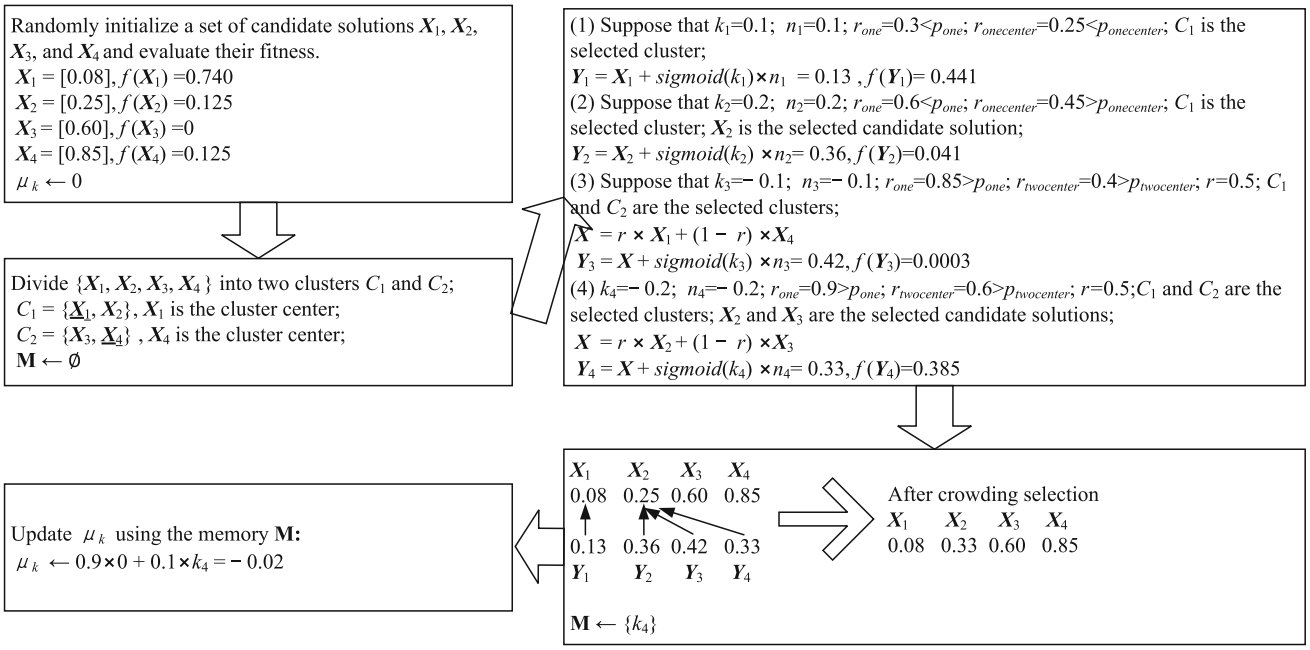

Fig. 3 Illustration of the procedures of ABSO

algorithms. We also investigate three versions of DBSO and compare them with ABSO to show the effect of parameter adjustment when solving different types of multimodal optimization problems.

4.1 Experimental Setup

4.1.1 Test Functions

We use a set of benchmark test functions to examine the performance of the proposed BSO with adaptive step size control strategy. The benchmark function set contains 20 multimodal functions with different characteristics. Detailed descriptions and mathematical formulations of the test functions can be found in [49]. The first ten functions are simple multimodal functions with regular optima distributions. The rest ten functions are complex composite functions with rugged landscapes, in which many local and global optima exist. It is very challenging to locate all the global optima of these composition functions. It is worth noting that all the test functions are to be maximized. In this context, the two terms "peak" and "optimum" are interchangeable.

4.1.2 Performance Measure

Two performance indicators, i.e., peak ratio (PR) and success rate (SR), are adopted to examine the performance of the algorithms. They are defined in the following manner.

(a) Peak ratio (PR): PR is the average percentage of peaks located by the algorithm over multiple independent runs. PR is calculated as follows:

$$\mathrm{PR} = \frac{\sum_{i=1}^{\mathrm{NR}} \mathrm{NFP}_i}{\mathrm{NKP} \times \mathrm{NR}}, \tag{8}$$

where NKP is the number of known peaks of the test function, NR is the total number of runs. NFP_i represents the number of found peaks in the ith run.

(b) Success rate (SR): SR is the ratio of the number of successful runs to the total number of runs. A successful run is a run in which all the optima are detected. SR is calculated as follows:

$$\mathrm{SR} = \frac{\mathrm{NSR}}{\mathrm{NR}}, \tag{9}$$

where the numerator NSR denotes the number of successful runs.

4.1.3 Algorithms in Comparison

The proposed ABSO is compared with some classical niching algorithms, as well as some recently developed algorithms to demonstrate the effect of the proposed adaptive and dynamic control strategies. Specifically, ABSO is compared with two PSO-based niching algorithms (i.e., r3pso-lhc [22] and LIPS [23]), three DE-based niching algorithms (i.e., NCDE [28], dADE/nrand/1 [30], and PNPCDE [32]), an improved artificial bee colony algorithm (IABC [34]), and a BSO algorithm

Algorithm 2 ABSO

Randomly generate a set of N initial ideas $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N$ and evaluate their fitness;
Initialize μ_k to 0;
while termination criterion is not satisfied **do**
  Divide N ideas into M clusters $C_1, C_2, ..., C_M$ using the max-fitness clustering method (MCM);
  Record the best idea in each cluster as cluster center;
  Set **M** (Memory) to empty;
  for $i = 1$ to N **do**
    Sample a step size parameter k_i from distribution $N(\mu_k, 0.8)$;
    if $rand(0, 1) < p_{one}$ **then** ▷ Probability of generating a new idea based on a single cluster
      Randomly select a cluster C_j with probability p_j;
      if $rand(0, 1) < p_{onecenter}$ **then** ▷ Probability of using the cluster center to generate a new idea
        Add a random value to the cluster center of C_j to generate a new idea $\mathbf{Y}_i$ according to (4) with parameter k_i;
      else
        Add a random value to a random idea of the selected cluster C_j to generate a new idea $\mathbf{Y}_i$ according to (4) with parameter k_i;
      end if
    else ▷ Probability of generating a new idea based on two clusters
      Randomly select two clusters C_{j1} and C_{j2};
      if $rand(0, 1) < p_{twocenter}$ **then** ▷ Probability of using the cluster centers of two selected clusters to generate a new idea
        Combine the cluster centers of C_{j1} and C_{j2} to generate a new idea $\mathbf{Y}_i$ according to (3) and (4) with parameter k_i;
      else
        Combine two ideas randomly selected from C_{j1} and C_{j2} to generate a new idea $\mathbf{Y}_i$ according to (3) and (4) with parameter k_i;
      end if
    end if
    Evaluate $\mathbf{Y}_i$ and find the nearest neighbor $\mathbf{X}_{nn}$ of $\mathbf{Y}_i$;
    if $\mathbf{Y}_i$ is better than $\mathbf{X}_{nn}$ **then**
      Replace $\mathbf{X}_{nn}$ with $\mathbf{Y}_i$;
      Append k_i to the set **M**;
    end if
  end for
  Update μ_k using **M** according to (5);
end while

(OIF-BSO [48]). r3pso-lhc is a ring topology PSO in which particles interact with their left and right neighbors. LIPS is a swarm optimizer with locally informed learning mechanism. dADE/nrand/1 is a DE-based niching algorithm with a new mutation operator, an adaptive parameter control strategy, and a dynamic archive. Both NCDE and PNPCDE adopt the crowding selection technique. NCDE integrates a neighborhood-based mutation operator while PNPCDE integrates a parent-centric mutation operator with normalized neighborhoods. IABC is an improved artificial bee colony algorithm with two new solution search mechanisms. OIF-BSO is a BSO algorithm with an optima-identified framework.

Table 1 Setting of MaxFEs for the test functions

Test function	MaxFEs
F1–F5	5.0×10^4
F6–F7	2.0×10^5
F8–F9	4.0×10^5
F10–F13	2.0×10^5
F14–F20	4.0×10^5

4.1.4 Parameter Settings

The termination criterion of the algorithms is defined by the maximum number of fitness evaluations (MaxFEs). The setting of MaxFEs for each benchmark function is given in Table 1. For complex composite functions, a larger number of fitness evaluations are provided. The total number of runs (NR) is set to 50, i.e., each algorithm is executed 50 times for each test function. For ABSO, the size of candidate solution set (the number of ideas) is fixed at 100 and the cluster size is set to 5. In the creating process, the probability of generating a new idea based one cluster (P_{one}) is set to 0.8. In the scenario of one cluster based creating, the probability of using the cluster center $P_{onecenter}$ is set to 0.4. In the scenario of two clusters based creating, the probability of using the cluster centers $P_{twocenter}$ is set to 0.5. ABSO does not introduce any new parameters. The parameters of ABSO are inherited from BSO. They are determined according to the recommendation of the developer of BSO [19]. Since the goal of the paper is to examine the effectiveness of the adaptive parameter control strategy (i.e., the strategy for automatic update of the step size parameter k), the other parameters are kept the same as those of BSO. It is a reasonable choice since the parameter settings of BSO have been widely examined by the developer and other researchers. The parameters of the compared algorithms are configured according to the corresponding publications.

4.2 Overall Performance

The experimental results (PR and SR values) achieved by the algorithms are presented in Table 2. The best PR is highlighted in bold. From the table, it can be observed that ABSO is able to obtain the best results on 16 out of 20 problems. The first five problems F1–F5 are relatively easy to solve since they only contain a small number of decision variables. The algorithms are able to find all the optimal solutions of these problems in most of the runs. F6–F9 are problems with many peaks. The peak number of F9 even exceeds the population size. Therefore, it is difficult to locate all the optimal solutions. F6 and F8 have very sharp peaks and they pose great challenges to the fine-search capability of the algorithms. IABC performs the best on these problems owing to

Table 2 Peak ratio and success rate of the multimodal optimization algorithms

Function	r3pso-lhc			LIPS			dADE/nrand/1			NCDE		
	PR	SR	*p* value	PR	SR	*p* value	PR	SR	*p* value	PR	SR	*p* value
F1	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	3.27E−01
F2	0.996≈	0.980	3.27E−01	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00
F3	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00
F4	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00
F5	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00
F6	0.713+	0.000	2.87E−16	0.757+	0.000	3.91E−16	1.000−	1.000	2.51E−04	0.146+	0.000	2.36E−16
F7	0.366+	0.000	5.87E−18	0.489+	0.000	4.29E−16	0.496+	0.000	1.59E−15	0.688−	0.000	2.43E−03
F8	0.112+	0.000	6.18E−18	0.200+	0.000	6.35E−18	0.449≈	0.020	6.16E−01	0.504−	0.000	5.85E−04
F9	0.096+	0.000	5.97E−18	0.124+	0.000	6.30E−18	0.153+	0.000	7.76E−18	0.267−	0.000	9.36E−17
F10	0.933+	0.380	4.32E−11	0.992+	0.900	2.30E−02	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00
F11	0.700+	0.000	1.67E−20	0.863+	0.240	1.85E−12	0.667+	0.000	1.03E−22	0.797+	0.180	5.76E−14
F12	0.648+	0.000	1.64E−18	0.823+	0.140	1.08E−12	0.900+	0.420	1.45E−05	0.330+	0.000	2.74E−19
F13	0.643+	0.000	4.56E−20	0.690+	0.000	5.19E−18	0.670+	0.000	1.22E−20	0.593+	0.000	3.98E−19
F14	0.563+	0.000	1.20E−10	0.627+	0.000	3.92E−05	0.667+	0.000	3.42E−06	0.637+	0.000	1.51E−07
F15	0.213+	0.000	1.43E−18	0.418+	0.000	1.25E−12	0.493+	0.000	6.02E−06	0.265+	0.000	6.51E−20
F16	0.030+	0.000	8.15E−22	0.203+	0.000	7.79E−21	0.623+	0.000	1.23E−04	0.637+	0.000	1.80E−03
F17	0.028+	0.000	2.21E−19	0.208+	0.000	5.11E−15	0.300+	0.000	1.33E−10	0.248+	0.000	9.03E−18
F18	0.000+	0.000	6.11E−22	0.060+	0.000	5.87E−20	0.453+	0.000	7.39E−14	0.290+	0.000	9.53E−20
F19	0.000+	0.000	3.81E−21	0.008+	0.000	1.16E−20	0.048+	0.000	2.63E−19	0.095+	0.000	4.14E−19
F20	0.000+	0.000	8.65E−21	0.000+	0.000	8.65E−21	0.088+	0.000	3.99E−07	**0.245**−	0.000	2.68E−03
B/E/W	15/5/0			15/5/0			12/7/1			10/6/4		
	PNPCDE			IABC			OIF-BSO			ABSO		
	PR	SR	*p* value	PR	SR	*p* value	PR	SR	*p* value	PR	SR	
F1	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**	1.000	
F2	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**	1.000	
F3	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**	1.000	
F4	**1.000**≈	1.000	1.00E+00	0.995≈	0.980	3.27E−01	**1.000**≈	1.000	1.00E+00	**1.000**	1.000	
F5	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**	1.000	
F6	0.343+	0.000	3.07E−16	**0.947**≈	0.480	3.07E−03	0.661+	0.000	2.90E−16	**0.947**	0.760	
F7	0.604+	0.000	1.74E−03	**0.724**−	0.000	9.53E−08	0.534+	0.000	2.10E−13	0.648	0.000	
F8	0.074+	0.000	6.61E−18	**0.619**−	0.000	3.99E−07	0.379≈	0.000	1.54E−01	0.407	0.000	
F9	0.256−	0.000	1.80E−15	**0.369**−	0.000	6.74E−18	0.148+	0.000	2.67E−17	0.211	0.000	
F10	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**≈	1.000	1.00E+00	**1.000**	1.000	
F11	0.827+	0.120	9.57E−16	0.663+	0.000	1.64E−22	0.667+	0.000	1.03E−22	**0.990**	0.940	
F12	0.525+	0.000	1.32E−19	0.135+	0.000	3.30E−20	0.715+	0.000	4.43E−20	**0.978**	0.820	
F13	0.667+	0.000	4.59E−21	0.613+	0.000	2.48E−19	0.667+	0.000	4.59E−21	**0.940**	0.640	
F14	0.663+	0.000	2.54E−06	0.667+	0.000	3.42E−06	0.667+	0.000	3.42E−06	**0.737**	0.060	
F15	0.270+	0.000	1.08E−19	0.363+	0.000	5.33E−17	0.348+	0.000	1.93E−16	**0.593**	0.000	
F16	0.600+	0.000	6.70E−07	0.663≈	0.000	3.27E−01	**0.667**≈	0.000	1.00E+00	**0.667**	0.000	
F17	0.225+	0.000	3.54E−17	0.153+	0.000	1.18E−18	0.228+	0.000	5.01E−14	**0.460**	0.000	
F18	0.190+	0.000	7.46E−21	0.400+	0.000	1.60E−16	0.143+	0.000	1.87E−19	**0.640**	0.000	
F19	0.125+	0.000	3.81E−21	0.100+	0.000	7.86E−20	0.258+	0.000	3.03E−12	**0.463**	0.000	
F20	0.125+	0.000	3.76E−11	0.060+	0.000	6.73E−14	0.133+	0.000	7.75E−04	0.213	0.000	
B/E/W	13/6/1			9/8/3			12/8/0			NA		

The notation "+" represents that the PR value achieved by ABSO is significantly better than that of the corresponding algorithm, while the notation "−" denotes the opposite. The differences are detected using the Wilcoxon rank-sum test at significant level $\alpha = 0.05$. The last row of the table summarizes the B/E/W counts of ABSO against its competitor

Table 3 Rankings of the algorithms revealed by the Friedman's test

Algorithm	Ranking
ABSO	2.65
OIF-BSO	3.625
dADE/nrand/1	4.05
IABC	4.5
NCDE	4.5
PNPCDE	4.775
LIPS	5.25
r3pso-lhc	6.65

its two new search mechanisms, followed by NCDE. F10 is a problem whose peaks have regular ball shape and are evenly distributed in the search space. ABSO, dADE/nrand/1, NCDE, PNPCDE, IABC, and OIFBSO succeeded in finding all the peaks for F10. The remaining ten test problems are much more difficult to solve since they have irregular peaks and there exist many local optimal solutions. This feature poses great challenges on the algorithms' exploration capability, as well as the capability of maintaining found optimal solutions. The performance of ABSO is very competitive on the complex problems since the proposed adaptive strategy can find the most suitable step sizes through the feedback information collected in the optimization process. In addition, the integration of the crowding technique enhances the optima maintenance capability of ABSO. As a consequence, ABSO is able to locate more peaks even though the peaks are of different shapes and sizes. Finally, the Wilcoxon signed rank test is used to check the existence of significant differences between the numerical results of ABSO and its compared algorithms. The obtained p values are reported in Table 2. The notations "≈", "+", and "−" indicate that the PR values of ABSO are similar to, significantly better than, and significantly worse than that of the corresponding algorithm respectively. As can be concluded from the table, the proposed algorithm outperforms its opponents by at least nine test problems. To investigate the overall performance of the algorithms, the Friedman's test is conducted. The rankings of the algorithms obtained by the Friedman's test are listed in Table 3. According to the Friedman's test, ABSO has the highest ranking, followed by OIF-BSO, dADE/nrand/1, and IABC.

4.3 Convergence Speed

In this subsection, we study the convergence speed of the algorithms. In the context of multimodal optimization, the convergence speed of an algorithm is defined by the number of fitness evaluations required to locate all the global optima. In cases that the algorithm cannot locate all the global optima, the predefined number of fitness evaluations (MaxFEs) is counted as the convergence speed. Table 4 lists the convergence speed of the algorithms on test problems F1–F6. The results are averaged over 50 independent runs and the best results are highlighted in bold. Since F1–F5 are relatively easy to solve. The algorithm that tends toward exploitation will have an edge over other algorithms. As can be observed from Table 4, r3pso-lhc has the fastest conver-

Table 4 Convergence speed of the algorithms on test functions F1–F6

Algorithm	Func	F1	F2	F3	F4	F5	F6
r3pso-lhc	Avg	**200.00**	1896.00	**1020.00**	**6806.00**	**2650.00**	196454.00
	Std	**0.00**	564.26	**495.18**	**6214.45**	**344.24**	24822.00
LIPS	Avg	**200.00**	**1618.00**	1212.00	9316.00	3664.00	200000.00
	Std	**0.00**	**520.27**	702.75	1250.34	749.34	0.00
dADE/nrand/1	Avg	214.52	8742.22	1760.94	28555.98	6632.52	166917.48
	Std	4.14	3151.03	951.60	10310.97	2125.69	50648.81
NCDE	Avg	350.00	3028.00	2268.00	11232.00	4088.00	200000.00
	Std	222.93	1423.94	2561.21	3012.34	771.40	0.00
PNPCDE	Avg	**200.00**	2134.00	1066.00	23938.00	4916.00	200000.00
	Std	**0.00**	603.19	773.72	5648.85	1304.20	0.00
IABC	Avg	209.50	2677.54	1789.92	31556.04	5019.36	179195.64
	Std	37.61	1262.27	1550.16	5699.34	1654.65	28085.09
OIF-BSO	Avg	468.68	4009.06	3466.80	34238.40	13490.58	200000.00
	Std	194.04	2203.60	3901.36	4980.45	4101.65	0.00
ABSO	Avg	296.00	4940.00	3260.00	21000.00	13832.00	**84560.00**
	Std	111.28	2732.47	3345.56	1671.17	2056.16	**62724.73**

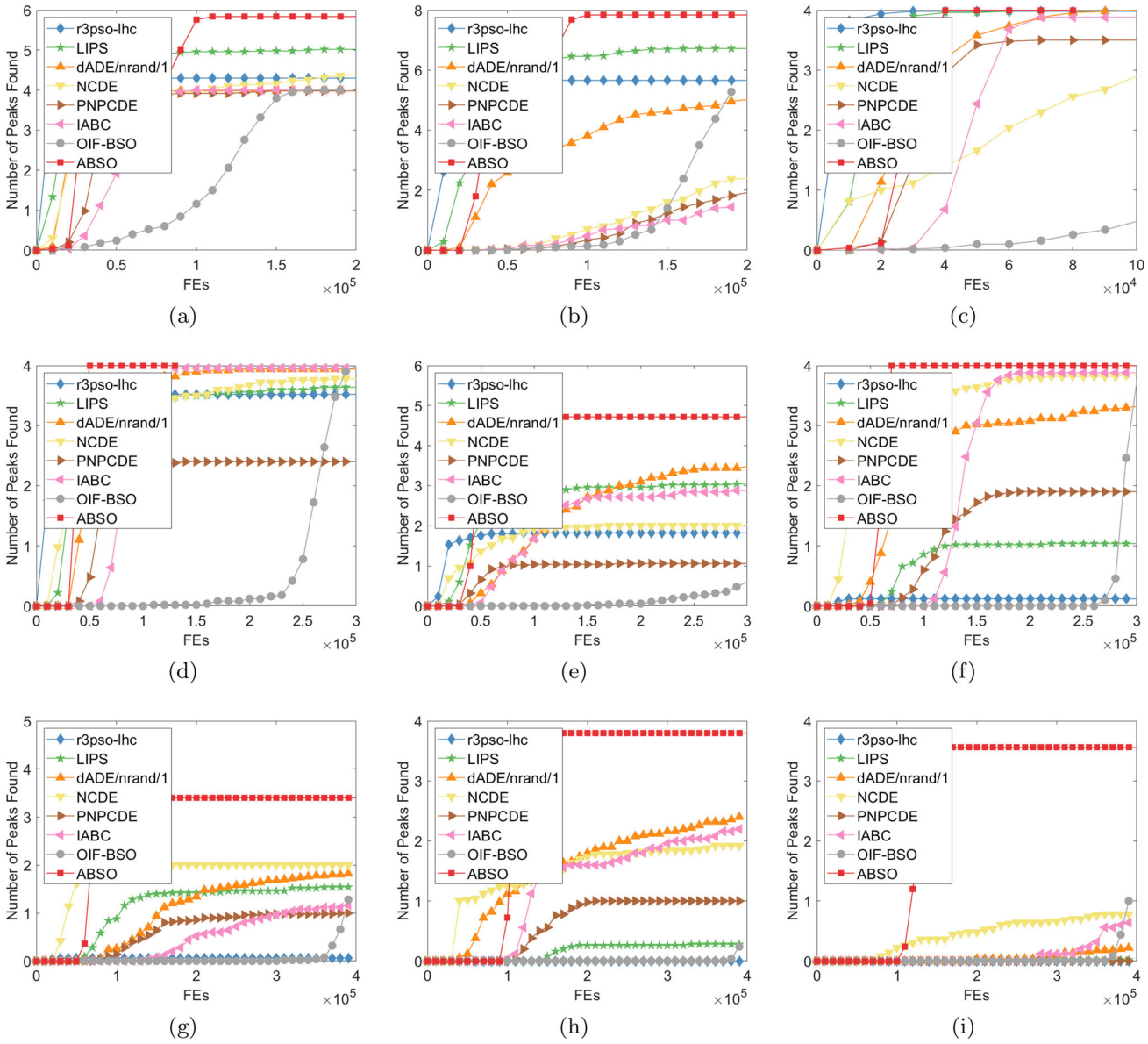

Fig. 4 Convergence graphs of the multimodal optimization algorithms. **a** F11, **b** F12, **c** F13, **d** F14, **e** F15, **f** F16, **g** F17, **h** F18, **i** F19

gence speed on four out of six test problems. LIPS performs the best on F2 while ABSO possesses the fastest speed on F6. The results of the algorithms on F7–F9 are reduced to the predefined MaxFEs and therefore not listed in the table. For the complex problem, the exploration capability plays an increasingly important role in high dimensional rugged landscapes. Note that it is very difficult to locate all the global optima, the convergence graphs of the algorithms are presented instead. Figure 4 shows the convergence graphs of the algorithms. The x-axis denotes the number of fitness evaluations, while the y-axis denotes the number of global optima found. It can be observed from the figure that ABSO is able to locate more optima than the compared algorithms using less fitness evaluations in majority of the problems F11–F19.

4.4 Effect of the New Creating Operator

In this subsection, we demonstrate the effectiveness of the new creating operator by comparing the results of ABSO and niching BSO (NBSO) with the original update formula (i.e., formula (1)). For ease of description, NBSO with the original update formula and different values of k is denoted as NBSO-ok. Specifically, we compare ABSO with NBSO-ok with five different values of k range from 1 to 20. The experimental results are tabulated in Table 5. The best results on each test problem are highlighted in bold.

From the table, it can be noted that the overall performance of NBSO-ok gradually increases with the value of k. However, their performance on problems with a large num-

Table 5 Comparison between ABSO and NBSO-ok with different values of k

Function	NBSO-o1		NBSO-o5		NBSO-o10		NBSO-o15		NBSO-o20		ABSO	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F2	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F4	0.880	0.560	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F5	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F6	0.036	0.000	0.544	0.000	0.832	0.020	0.972	0.580	**0.998**	0.960	0.947	0.760
F7	0.428	0.000	0.521	0.000	0.602	0.000	0.646	0.000	**0.648**	0.000	**0.648**	0.000
F8	0.000	0.000	0.077	0.000	0.226	0.000	0.322	0.000	**0.488**	0.000	0.407	0.000
F9	0.066	0.000	0.107	0.000	0.144	0.000	0.169	0.000	0.190	0.000	**0.211**	0.000
F10	0.527	0.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
F11	0.610	0.000	0.690	0.020	0.830	0.240	0.877	0.340	0.850	0.280	**0.990**	0.940
F12	0.060	0.000	0.735	0.040	0.955	0.680	**0.995**	0.960	**0.995**	0.960	0.978	0.820
F13	0.360	0.000	0.673	0.000	0.820	0.080	0.837	0.120	0.883	0.300	**0.940**	0.640
F14	0.517	0.000	0.647	0.000	0.667	0.000	0.677	0.000	0.687	0.000	**0.737**	0.060
F15	0.198	0.000	0.450	0.000	0.663	0.000	**0.690**	0.000	0.688	0.000	0.593	0.000
F16	0.073	0.000	0.193	0.000	0.593	0.000	**0.667**	0.000	**0.667**	0.000	**0.667**	0.000
F17	0.015	0.000	0.030	0.000	0.353	0.000	0.495	0.000	**0.513**	0.000	0.460	0.000
F18	0.007	0.000	0.037	0.000	0.047	0.000	0.087	0.000	0.307	0.000	**0.640**	0.000
F19	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.093	0.000	**0.463**	0.000
F20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.213**	0.000

ber of decision variables (i.e., F18–F20) is not satisfactory, despite the value of k. For NBSO-ok, a large range of values needs to be tested in order to find the most effective setting for problems with a large number of decision variables. In comparison, with the new creating operator and the adaptive parameter control strategy, ABSO is able to achieve competitive results on F18–F20. The overall performance of ABSO is better than that of NBSO-ok. This result reveals the benefit of the simplified creating operator shown in formula (4).

4.5 Effect of the Adaptive Step Size Control Strategy

In this subsection, we study the effect of the adaptive step size control strategy. To visualize the hidden variable μ_k, we plot the change of μ_k with respect to the grows of the number of fitness evaluations in Fig. 5. The results are averaged over 50 independent runs and the confidence intervals are determined by the standard deviation. From the figure, it can be observed that for most of the test problems, μ_k gradually decreases from 0 to negative values range from -10 to -25. An exception is that when solving F3, μ_k gradually increases from 0 to 10. The changing dynamic of μ_k varies from problem to problem. This indicates that for different types of multimodal problems, the most suitable parameter setting is different. It is a challenging and time-consuming task to manually choose the best setting through trial-and-error.

To check whether the adaptive strategy can enhance the algorithm performance, ABSO is compared with NBSO with fixed parameter settings. Seven different settings of the hidden variable μ_k, namely, 5, 0, -5, -10, -15, -20, and -25 are examined in the experiment. NBSO with these settings are denoted as NBSO-p5, NBSO-0, NBSO-m5, NBSO-m10, NBSO-m15, NBSO-m20, and NBSO-m25 respectively. Table 6 lists the experimental results of the algorithms, and the best PR values are in bold. It can be observed from the table that ABSO obtain the best PR values for 19 out of the 20 problems. The gap between the PR values of ABSO and NBSO with fixed setting of μ_k becomes larger when solving the complex composite problems F11–F20. This is because for these complex problems, different stages of the search process require different parameter values and the fixed parameter setting cannot meet the requirement of this optimization dynamic.

We also test the three dynamic control strategies (convex, linear, and concave) suggested in Sect. 3. UB and LB are set to 0 and -25 respectively. In this way, μ_k is gradually decreased from 0 to -25. Table 7 lists the experimental results. NBSO with the three dynamic strategies are denoted as DBSO-convex, DBSO-linear, and DBSO-concave respectively. From the table, it can be noticed that both the linear

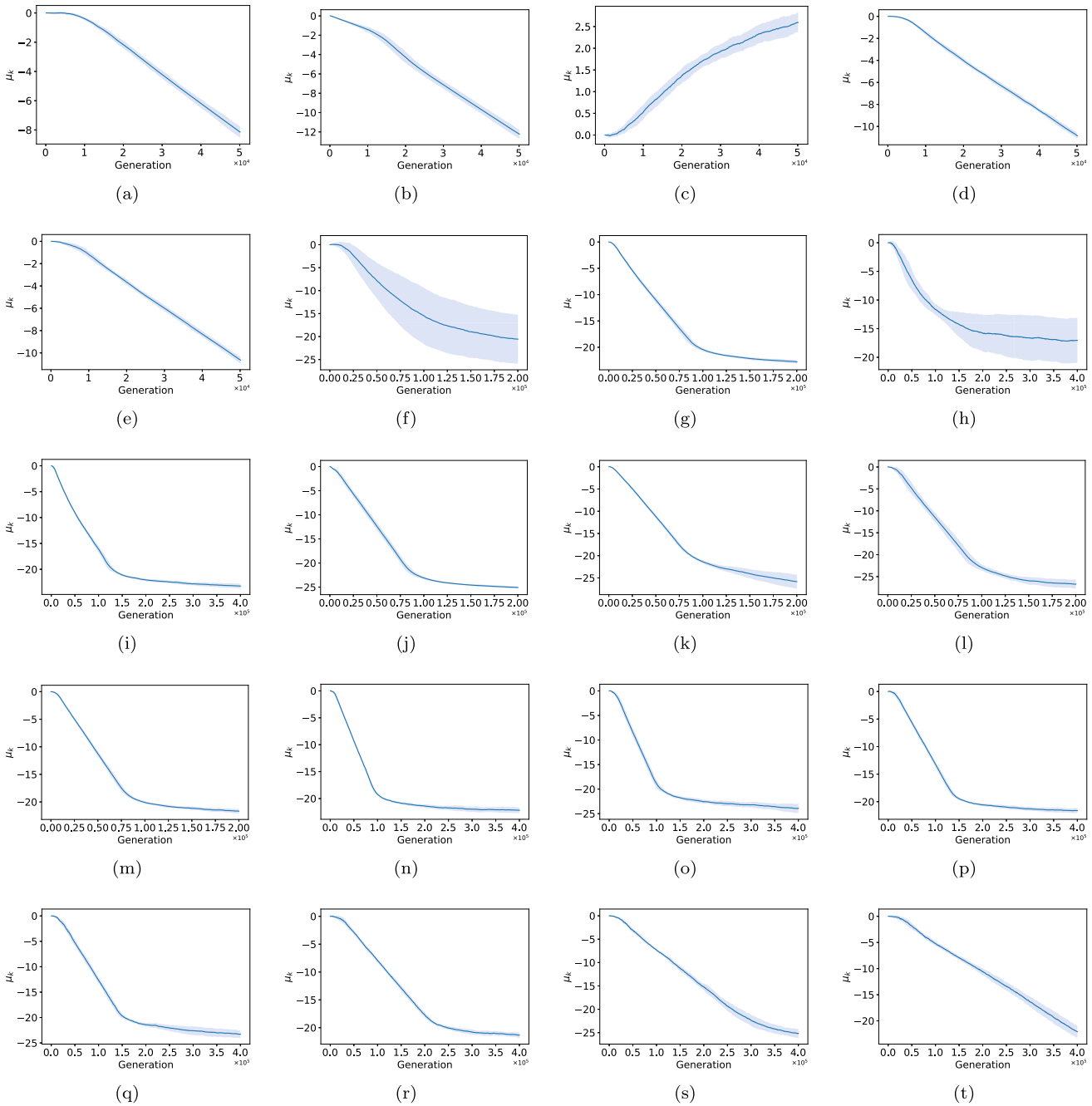

Fig. 5 Change of μ_k along with the number of FEs. **a–d** F1–F4, **e–h** F5–F8, **i–l** F9–F12, **m–p** F13–F16, **q–t** F17–F20

strategy and the convex strategy perform quite well on the test problems, while the performance of the concave strategy is relatively poor. According to the curves plotted in Fig. 5, the linear and convex strategies roughly match the changes of μ_k. The overall rankings of the algorithms provided by the Friedman's test are listed in Table 8. DBSO-convex has the highest ranking. DBSO-linear and ABSO are in the second and third places respectively. This result does not indicate that the dynamic control strategy is superior to the adaptive control strategy. To apply the dynamic control strategy, we first need to specify the upper bound UB and lower bound LB, which is a difficult task without a priori knowledge. In the experiment, UB and LB are determined by the numerical results returned by the adaptive control strategy.

5 Conclusion

In this paper, we developed an adaptive BSO algorithm that automatically adjusts the step size parameter according to the feedback information collected in the optimization pro-

Table 6 Comparison between ABSO and NBSO with fixed step size parameter

Function	NBSO-0		NBSO-p5		NBSO-m5		NBSO-m10		NBSO-m15		NBSO-m20		NBSO-m25		ABSO	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	1.000	1.000	1.000	1.000	1.000	1.000	0.630	0.340	0.570	0.300	0.660	0.460	0.530	0.280	**1.000**	1.000
F2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**1.000**	1.000
F3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.980	0.980	0.980	0.940	0.940	**1.000**	1.000
F4	0.085	0.000	0.010	0.000	1.000	1.000	0.810	0.420	0.245	0.000	0.255	0.000	0.220	0.000	**1.000**	1.000
F5	0.560	0.300	0.320	0.080	1.000	1.000	0.990	0.980	0.930	0.860	0.980	0.960	0.970	0.940	**1.000**	1.000
F6	0.000	0.000	0.000	0.000	0.734	0.000	0.609	0.000	0.002	0.000	0.001	0.000	0.001	0.000	**0.947**	0.760
F7	0.281	0.000	0.169	0.000	0.578	0.000	0.409	0.000	0.258	0.000	0.252	0.000	0.245	0.000	**0.648**	0.000
F8	0.000	0.000	0.000	0.000	0.000	0.000	0.188	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.407**	0.000
F9	0.010	0.000	0.002	0.000	**0.215**	0.000	0.090	0.000	0.022	0.000	0.019	0.000	0.020	0.000	0.211	0.000
F10	0.015	0.000	0.003	0.000	**1.000**	1.000	0.998	0.980	0.685	0.000	0.578	0.000	0.582	0.000	**1.000**	1.000
F11	0.010	0.000	0.000	0.000	0.667	0.000	0.627	0.000	0.250	0.000	0.207	0.000	0.250	0.000	**0.990**	0.940
F12	0.000	0.000	0.000	0.000	0.438	0.000	0.625	0.000	0.028	0.000	0.008	0.000	0.010	0.000	**0.978**	0.820
F13	0.003	0.000	0.000	0.000	0.660	0.000	0.537	0.000	0.187	0.000	0.187	0.000	0.190	0.000	**0.940**	0.640
F14	0.000	0.000	0.000	0.000	0.433	0.000	0.180	0.000	0.007	0.000	0.000	0.000	0.000	0.000	**0.737**	0.060
F15	0.000	0.000	0.000	0.000	0.188	0.000	0.173	0.000	0.030	0.000	0.005	0.000	0.013	0.000	**0.593**	0.000
F16	0.000	0.000	0.000	0.000	0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.667**	0.000
F17	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.460**	0.000
F18	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.640**	0.000
F19	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.463**	0.000
F20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.213**	0.000

Table 7 Comparison between ABSO and DBSO with different control dynamics

Function	ABSO		DBSO-linear		DBSO-convex		DBSO-concave	
	PR	SR	PR	SR	PR	SR	PR	SR
F1	**1.000**	1.000	1.000	1.000	1.000	1.000	0.990	0.980
F2	**1.000**	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F3	**1.000**	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F4	**1.000**	1.000	1.000	1.000	1.000	1.000	0.865	0.560
F5	**1.000**	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F6	0.947	0.760	0.963	0.520	**0.998**	0.960	0.637	0.000
F7	0.648	0.000	**0.678**	0.000	0.672	0.000	0.560	0.000
F8	0.407	0.000	0.447	0.000	**0.769**	0.000	0.172	0.000
F9	0.211	0.000	0.237	0.000	**0.244**	0.000	0.184	0.000
F10	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	1.000	1.000
F11	0.990	0.940	**1.000**	1.000	0.930	0.600	0.990	0.940
F12	0.978	0.820	0.980	0.840	**1.000**	1.000	0.860	0.220
F13	0.940	0.640	**0.987**	0.920	0.893	0.380	0.937	0.640
F14	0.737	0.060	**0.767**	0.060	0.740	0.000	0.717	0.040
F15	0.593	0.000	0.683	0.000	**0.735**	0.000	0.503	0.000
F16	0.667	0.000	**0.673**	0.000	0.667	0.000	0.657	0.000
F17	0.460	0.000	0.430	0.000	**0.545**	0.000	0.325	0.000
F18	0.640	0.000	0.630	0.000	**0.653**	0.000	0.433	0.000
F19	0.463	0.000	0.413	0.000	**0.465**	0.000	0.088	0.000
F20	0.213	0.000	0.168	0.000	**0.230**	0.000	0.000	0.000

Table 8 Rankings of ABSO and DBSO revealed by the Friedman's test

Algorithm	Ranking
DBSO-convex	1.875
DBSO-linear	2.05
ABSO	2.5
DBSO-concave	3.575

cess. A modified creating operator for producing new ideas is devised and incorporated into BSO to reduce the level randomness so that the control strategy can correctly assign credit of generating better candidate solutions to the setting of parameters. Experiments have been conducted on a set of benchmark multimodal optimization problems to examine the effect of the adaptive step size control strategy. The experimental results show that ABSO is very competitive compared with some recently developed algorithms. The performance of ABSO is superior to BSO with fixed parameter settings. With the adaptive control strategy, the step size parameter is constantly updated to increase the success rate of producing better ideas. In this way, ABSO can locate multiple optimal solutions with increased efficiency.

We also develop a systematic approach to the generation of dynamic strategies. The systematic approach can produce control strategies with decreasing curves of different shapes (convex, linear, and concave). According to the experimental results, if the control curves match the feature of the problems being solved, DBSO is capable of achieving higher performance than the adaptive control strategy.

The focus of this paper is to improve the performance of BSO in multimodal optimization through the adaptive adjustment of the step size parameter k. Although the experimental results reveal the effectiveness of the proposed adaptation scheme, the developed algorithm does not get rid of all the control parameters. We still need to find the most suitable settings for the other parameters through extensive trial-and-error. Therefore, ABSO still has room for improvement. It is possible to design more advanced adaptation scheme that simultaneously adjusts all the parameters. In this way, we can obtain an algorithm which is parameter-free and more robust. Furthermore, it is essential to apply the algorithm to more complex optimization problems to demonstrate its potential for real-world applications.

Author Contributions Conceptualization, methodology, writing-original draft preparation, YZ; writing-review and editing, WW; formal analysis, SX; supervision, ZW.

Funding This research was funded in part by the National Natural Science Foundation of China under Grant 62106046 and Grant 62106055, in part by the Guangdong Natural Science Foundation under Grant 2019A1515110474 and grant 2022A1515011825.

Availability of data and material The data presented in this study are available on request from the corresponding author.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Li, X., Epitropakis, M.G., Deb, K., Engelbrecht, A.: Seeking multiple solutions: an updated survey on niching methods and their applications. IEEE Trans. Evol. Comput. **21**(4), 518–538 (2016)
2. Huang, T., Gong, Y.-J., Kwong, S., Wang, H., Zhang, J.: A niching memetic algorithm for multi-solution traveling salesman problem. IEEE Trans. Evol. Comput. **24**(3), 508–522 (2019)
3. Hu, Y., Zhang, K.: Multimodal optimization evolutionary algorithm for RNA secondary structure prediction. In: The Fifth International Conference on Biological Information and Biomedical Engineering, Association for Computing Machinery, Hangzhou, China, pp. 1–7 (2021)
4. Huang, T., Gong, Y.-J., Zhang, Y.-H., Zhan, Z.-H., Zhang, J.: Automatic planning of multiple itineraries: a niching genetic evolution approach. IEEE Trans. Intell. Transp. Syst. **21**(10), 4225–4240 (2019)
5. Lotf, J.J., Azgomi, M.A., Reza, E.D.M.: An improved influence maximization method for social networks based on genetic algorithm. Phys. A Stat. Mech. Appl. **586**, 126480 (2022)
6. Aziz, R.M., Mahto, R., Goel, K., Das, A., Kumar, P., Saxena, A.: Modified genetic algorithm with deep learning for fraud transactions of ethereum smart contract. Appl. Sci. **13**(2), 697 (2023)
7. Devarriya, D., Gulati, C., Mansharamani, V., Sakalle, A., Bhardwaj, A.: Unbalanced breast cancer data classification using novel fitness functions in genetic programming. Expert Syst. Appl. **140**, 112866 (2020)
8. Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A., et al.: Differential evolution: a review of more than two decades of research. Eng. Appl. Artif. Intell. **90**, 103479 (2020)
9. Opara, K.R., Arabas, J.: Differential evolution: a survey of theoretical analyses. Swarm Evol. Comput. **44**, 546–558 (2019)
10. Hansen, N.: A global surrogate assisted CMA-ES. In: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, Prague, Czech Republic, pp. 664–672 (2019)
11. Biedrzycki, R.: Handling bound constraints in CMA-ES: an experimental study. Swarm Evol. Comput. **52**, 100627 (2020)
12. Shami, T.M., El-Saleh, A.A., Alswaitti, M., Al-Tashi, Q., Summakieh, M.A., Mirjalili, S.: Particle swarm optimization: a comprehensive survey. IEEE Access **10**, 10031–10061 (2022)

13. Houssein, E.H., Gad, A.G., Hussain, K., Suganthan, P.N.: Major advances in particle swarm optimization: theory, analysis, and application. Swarm Evol. Comput. **63**, 100868 (2021)
14. Rokbani, N., Kumar, R., Abraham, A., Alimi, A.M., Long, H.V., Priyadarshini, I., Son, L.H.: Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. Soft Comput. **25**, 3775–3794 (2021)
15. Zhou, X., Ma, H., Jianggang, G., Chen, H., Deng, W.: Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. Eng. Appl. Artif. Intell. **114**, 105139 (2022)
16. Ullah, A.: Artificial bee colony algorithm used for load balancing in cloud computing. IAES Int. J. Artif. Intell. **8**(2), 156 (2019)
17. Kaya, E., Gorkemli, B., Akay, B., Karaboga, D.: A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems. Eng. Appl. Artif. Intell. **115**, 105311 (2022)
18. Ali, S., Bhargava, A., Saxena, A., Kumar, P.: A hybrid marine predator sine cosine algorithm for parameter selection of hybrid active power filter. Mathematics **11**(3), 598 (2023)
19. Shi, Y.: Brain storm optimization algorithm. In: Advances in Swarm Intelligence: Second International Conference, ICSI 2011, Chongqing, China, June 12-15, 2011, Proceedings, Part I 2, pp 303–309. Springer (2011)
20. Zhan, Z., Zhang, J., Shi, Y., Liu, H.: A modified brain storm optimization. In: 2012 IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012)
21. Cheng, S., Qin, Q., Chen, J., Shi, Y.: Brain storm optimization algorithm: a review. Artif. Intell. Rev. **46**, 445–458 (2016)
22. Li, X.: Niching without niching parameters: particle swarm optimization using a ring topology. IEEE Trans. Evol. Comput. **14**(1), 150–169 (2009)
23. Qu, B.-Y., Suganthan, V., Das, S.: A distance-based locally informed particle swarm model for multimodal optimization. IEEE Trans. Evol. Comput. **17**(3), 387–402 (2012)
24. Goldberg, D.E., Richardson, J., et al.: Genetic algorithms with sharing for multimodal function optimization. In: Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, vol. 4149. Lawrence Erlbaum, Hillsdale (1987)
25. Thomsen, R.: Multimodal optimization using crowding-based differential evolution. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), vol. 2, pp. 1382–1389. IEEE (2004)
26. Pétrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 798–803. IEEE (1996)
27. Li, J.-P., Balazs, M.E., Parks, G.T., Clarkson, P.J.: A species conserving genetic algorithm for multimodal function optimization. Evol. Comput. **10**(3), 207–234 (2002)
28. Qu, B.-Y., Suganthan, P.N., Liang, J.-J.: Differential evolution with neighborhood mutation for multimodal optimization. IEEE Trans. Evol. Comput. **16**(5), 601–614 (2012)
29. Gao, W., Yen, G.G., Liu, S.: A cluster-based differential evolution with self-adaptive strategy for multimodal optimization. IEEE Trans. Cybern. **44**(8), 1314–1327 (2013)
30. Epitropakis, M.G., Li, X., Burke, E.K.: A dynamic archive niching differential evolution algorithm for multimodal optimization. In: 2013 IEEE Congress on Evolutionary Computation, pp. 79–86. IEEE (2013)
31. Biswas, S., Kundu, S., Das, S.: Inducing niching behavior in differential evolution through local information sharing. IEEE Trans. Evol. Comput. **19**(2), 246–263 (2014)
32. Biswas, S., Kundu, S., Das, S.: An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution. IEEE Trans. Cybern. **44**(10), 1726–1737 (2014)
33. Zhang, Y.-H., Gong, Y.-J., Zhang, H.-X., Tian-Long, G., Zhang, J.: Toward fast niching evolutionary algorithms: a locality sensitive hashing-based approach. IEEE Trans. Evol. Comput. **21**(3), 347–362 (2016)
34. Ma, S., Wang, Y., Zhang, S.: Improved artificial bee colony algorithm for multimodal optimization based on crowding method. J. Organ. End User Comput. (JOEUC) **34**(3), 1–18 (2022)
35. Huang, T., Gong, Y.-J., Chen, W.-N., Wang, H., Zhang, J.: A probabilistic niching evolutionary computation framework based on binary space partitioning. IEEE Trans. Cybern. **52**(1), 51–64 (2020)
36. Wang, Z.-J., Zhan, Z.-H., Lin, Y., Wei-Jie, Yu., Wang, H., Kwong, S., Zhang, J.: Automatic niching differential evolution with contour prediction approach for multimodal optimization problems. IEEE Trans. Evol. Comput. **24**(1), 114–128 (2019)
37. Chen, Z.-G., Zhan, Z.-H., Wang, H., Zhang, J.: Distributed individuals for multiple peaks: a novel differential evolution for multimodal optimization problems. IEEE Trans. Evol. Comput. **24**(4), 708–719 (2019)
38. Zhao, H., Zhan, Z.-H., Lin, Y., Chen, X., Luo, X.-N., Zhang, J., Kwong, S., Zhang, J.: Local binary pattern-based adaptive differential evolution for multimodal optimization problems. IEEE Trans. Cybern. **50**(7), 3343–3357 (2019)
39. Sheng, W., Wang, X., Wang, Z., Li, Q., Chen, Y.: Adaptive memetic differential evolution with niching competition and supporting archive strategies for multimodal optimization. Inf. Sci. **573**, 316–331 (2021)
40. Liu, Q., Du, S., van Wyk, B.J., Sun, Y.: Double-layer-clustering differential evolution multimodal optimization by speciation and self-adaptive strategies. Inf. Sci. **545**, 465–486 (2021)
41. Ahrari, A., Deb, K.: Multimodal optimization by evolution strategies with repelling subpopulations. In: Preuss, M., Epitropakis, M.G., Li, X., Fieldsend, J.E. (eds) Metaheuristics for Finding Multiple Solutions. Natural Computing Series. Springer, Cham, pp. 145–163 (2021)
42. Ahmed, R., Nazir, A., Mahadzir, S., Shorfuzzaman, M., Islam, J.: Niching grey wolf optimizer for multimodal optimization problems. Appl. Sci. **11**(11), 4795 (2021)
43. El-Abd, M.: Global-best brain storm optimization algorithm. Swarm Evol. Comput. **37**, 27–44 (2017)
44. Zhao, F., Hu, X., Wang, L., Zhao, J., Tang, J.J.: A reinforcement learning brain storm optimization algorithm (BSO) with learning mechanism. Knowl. Based Syst. **235**, 107645 (2022)
45. Yang, Yu., Gao, S., Wang, Y., Lei, Z., Cheng, J., Todo, Y.: A multiple diversity-driven brain storm optimization algorithm with adaptive parameters. IEEE Access **7**, 126871–126888 (2019)
46. Zhou, D., Shi, Y., Cheng, S.: Brain storm optimization algorithm with modified step-size and individual generation. In: Ying, T., Yuhui, S., Zhen, J. (eds.) Advances in Swarm Intelligence, pp. 243–252. Springer, Berlin (2012)
47. Cheng, S., Sun, Y., Chen, J., Qin, Q., Chu, X., Lei, X., Shi, Y.: A comprehensive survey of brain storm optimization algorithms. In: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, pp, 1637–1644 (2017)
48. Dai, Z., Fang, W., Tang, K., Li, Q.: An optima-identified framework with brain storm optimization for multimodal optimization problems. Swarm Evol. Comput. **62**, 100827 (2021)
49. Li, X., Engelbrecht, A., Epitropakis, M.G.: Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep (2013)