RESEARCH ARTICLE

Simple gravitational particle swarm algorithm for multimodal optimization problems

Yoshikazu Yamanaka*, Katsutoshi Yoshida

Department of Mechanical and Intelligent Engineering, Utsunomiya University, Utsunomiya, Tochigi, Japan

* yyamanaka@cc.utsunomiya-u.ac.jp

Citation: Yamanaka Y, Yoshida K (2021) Simple gravitational particle swarm algorithm for multimodal optimization problems. PLoS ONE 16(3): e0248470. https://doi.org/10.1371/journal.pone.0248470

Editor: Seyedali Mirjalili, Torrens University Australia, AUSTRALIA

Received: April 16, 2020

Accepted: February 28, 2021

Published: March 18, 2021

Data Availability Statement: All relevant data are within the manuscript.

Funding: This work was supported by Japan Society for the Promotion of Science, https://www.jsps.go.jp, KAKENHI Grant Numbers JP17H06552 to YY; and KAKENHI Grant Numbers JP18H01391 to KY. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Abstract

In real world situations, decision makers prefer to have multiple optimal solutions before making a final decision. Aiming to help the decision makers even if they are non-experts in optimization algorithms, this study proposes a new and simple multimodal optimization (MMO) algorithm called the gravitational particle swarm algorithm (GPSA). Our GPSA is developed based on the concept of "particle clustering in the absence of clustering procedures". Specifically, it simply replaces the global feedback term in classical particle swarm optimization (PSO) with an inverse-square gravitational force term between the particles. The gravitational force mutually attracts and repels the particles, enabling them to autonomously and dynamically generate sub-swarms in the absence of algorithmic clustering procedures. Most of the sub-swarms gather at the nearby global optima, but a small number of particles reach the distant optima. The niching behavior of our GPSA was tested first on simple MMO problems, and then on twenty MMO benchmark functions. The performance indices (peak ratio and success rate) of our GPSA were compared with those of existing niching PSOs (ring-topology PSO and fitness Euclidean-distance ratio PSO). The basic performance of our GPSA was comparable to that of the existing methods. Furthermore, an improved GPSA with a dynamic parameter delivered significantly superior results to the existing methods on at least 60% of the tested benchmark functions.

Introduction

Multimodal optimization (MMO) algorithms [1] (also known as niching methods or techniques) can locate multiple global optima in a single run, which is essential for solving many scientific and engineering optimization problems, e.g., Toyota paradox [2], motor design [3], clustering validity functions [4], network modeling [5], truss-structure design [6], overlay network [7], multi-robot cooperation [8], wireless sensor network [9], object detection [10], and honeycomb core design [11]. Compared with unimodal optimization (UMO) algorithms that can locate just a single global optimum, MMO algorithms have several benefits in some real-world problems [1], where some factors can be difficult to model mathematically, e.g., degree of difficulty in manufacturing. Specifically, having multiple solutions with a similar quality will give a decision maker more options for consideration, with factors that are not captured in the mathematical model. Finding multiple solutions may also help to reveal hidden properties or relations of the problem, e.g., the distribution of the solution set in the problem space. Therefore, MMO algorithms provide richer information about the problem domain than the UMO algorithms.

To achieve the MMO algorithms, several well-known niching techniques have been developed since the 1970s, namely, crowding [12], deterministic crowding [13], fitness sharing [14], derating [15], restricted tournament selection [16], clustering [17], and speciation [18]. Initially, these niching techniques were developed for evolutionary algorithms (EAs) and genetic algorithms (GAs).

Since the 2000s, some MMO algorithms based on particle swarm optimization (PSO) [19–21] have been proposed [22–24]. These algorithms simply replace the global best of classic PSO with neighborhood best; hence, they can be categorized into a one-stage method like the classic PSO. Owing to their simplicity, they have been successfully applied to many real-world problems [4–11]. However, these one-stage MMO algorithms are known to perform poorly compared with state-of-the-art niching methods [24, 25].

Recently, two-stage MMO algorithms [26–29] won the multimodal optimization competitions held by the Congress on Evolutionary Computation (CEC) and the Genetic and Evolutionary Computation Conference (GECCO). The first stage of the two-stage niching separates the candidate solutions into subpopulations by a clustering algorithm [28]. In the second stage, each subpopulation seeks the optimum by a core algorithm, which utilizes a restart scheme and taboo archive. However, non-experts in optimization algorithms hesitate to apply such complicated algorithms to their real-world problems. Although the source codes of these algorithms are available, they may not be directly implementable in the production items of a firm. In such situations, engineers should understand and implement the algorithms by themselves [30]. Therefore, it seems that the existing MMO algorithms have a dilemma between searching performance and algorithmic complexity.

The present study aims to develop a simple and powerful algorithm, which can be easily understood and implemented even by non-experts in optimization algorithms and can outperform the existing one-stage methods. The present paper proposes a new, simple, and purely dynamical one-stage method called the gravitational particle swarm algorithm (GPSA). This method replaces the linear feedback term involving the global best in the classical PSO framework with the inverse-square gravitational force between the particles. Under this mechanism, sub-swarms will autonomously self-organize at the nearby optima, but a few particles will intermittently escape, thereby maximizing the coverage of widely distributed multiple optima.

As our GPSA automatically and dynamically generates the above-mentioned swarm behavior without any clustering algorithms, restart scheme, and taboo archive, it is tractable even for non-experts. Furthermore, it requires fewer computational resources and fewer tuning parameters than existing MMO algorithms [22–24, 26, 31, 32]. On the CEC 2013 niching test problems [33], our GPSA performed comparably to the existing methods [22, 25], and can significantly outperform them by assigning a dynamic parameter.

Our GPSA differs from the existing one-stage algorithms based on PSO [22–24]. In particular, our GPSA omits the algorithmic particle-selecting procedures—sorting and selecting personal-best or nearest best(s)—of these algorithms. It also differs from ring-topology PSO (RPSO) [25] because it does not use ring-topology. Furthermore, our GPSA differs from methods based on gravity force [34–36] that cannot detect multiple global optima. Although a gravity-force method that detects multiple global optima has been proposed [32], this method requires algorithmic particle-selecting procedures, which are omitted in our GPSA. In addition, other well-known niching techniques utilized in EAs and GAs [12–18] are not required in our GPSA.

The remainder of this paper is organized as follows. Section 2 discusses the MMO problems and the drawbacks of the existing methods [22–29, 31, 32]. Section 3 introduces our GPSA, describes its search mechanism, and demonstrates its niching capability on one- and two-dimensional functions. Section 4 evaluates the performance of our GPSA on the CEC 2013 niching test problems, and confirms the comparable performances of our GPSA and the existing one-stage methods. In Section 5, our GPSA is improved by assigning a dynamic parameter. The superiority of the improved version over the existing methods is demonstrated in this section. Section 6 concludes the paper.

Background and related work

MMO problems

Consider an optimization problem of the form

$$\underset{\boldsymbol{x}}{\text{Maximize}}\, f(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^d, \tag{1}$$

where $\boldsymbol{x}$ is a d-dimensional vector and $f : \mathbb{R}^d \to \mathbb{R}$ is a real-valued function. This study focuses on the case where multiple global optima satisfy (1), i.e.,

$$\boldsymbol{x}_k^* = \arg\max f(\boldsymbol{x}),\ k = 1, 2, \cdots, N^{\text{go}}, \tag{2}$$

where $\boldsymbol{x}_k^*$ is kth global optima and N^{go} is the number of these global optima. Such problems are said to be multimodal, namely, they are MMO problems [1]. Conversely, problems with a single global optimum ($N^{\text{go}} = 1$) are known as UMO problems.

Eq (1) can be solved by particle-swarm-based approaches by considering the motion of a swarm of N candidate solutions:

$$X(t) := \{\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_N(t)\},\ \boldsymbol{x}_i(t) \in \mathbb{R}^d,\ t = 0, 1, \cdots, t_{\max}, \tag{3}$$

where t is discrete time. The candidate solutions (called particles) explore the d-dimensional domain $\mathbb{R}^d$, seeking either multiple global optima (in MMO problems) or a unique optimum (in UMO problems). The performances of such approaches therefore depend on the swarm movements.

Classical PSO

PSO was originally proposed in [19] and [20], and numerous variants for improvements have been proposed. In a typical classical PSO [21], the motion of the ith particle $x_i(t)$ is recursively described as [37]:

$$\begin{cases} \boldsymbol{v}_i(t+1) = \omega \boldsymbol{v}_i(t) + c_1 \boldsymbol{r}_{1i}(t) \otimes (\boldsymbol{pb}_i(t) - \boldsymbol{x}_i(t)) + c_2 \boldsymbol{r}_{2i}(t) \otimes \boldsymbol{a}_i(t), & \text{(4a)} \\ \boldsymbol{x}_i(t+1) = \boldsymbol{v}_i(t+1) + \boldsymbol{x}_i(t),\ t = 0, 1, \ldots, t_{\max}, & \text{(4b)} \end{cases}$$

with

$$\boldsymbol{a}_i(t) = \boldsymbol{gb}(t) - \boldsymbol{x}_i(t), \tag{5}$$

where $\otimes$ denotes component-wise multiplication, $\boldsymbol{v}_i(t)$ is the velocity of the ith particle, and $\boldsymbol{r}_{1i}(t)$ and $\boldsymbol{r}_{2i}(t)$ are independent white random-process vectors whose components are uniformly distributed over [0, 1]. The personal bests, denoted by $\boldsymbol{pb}_i(t) \in \mathbb{R}^d$, are the highest-cost positions found by each particle thus far (up to time t) among $\boldsymbol{x}_i(0), \cdots, \boldsymbol{x}_i(t)$. The global best

$\boldsymbol{gb}(t) \in \mathbb{R}^d$ is the best solution found by any particle up to time t. $(\omega, c_1, c_2) \in \mathbb{R}^3$ are tuning parameters.

Eqs (4) and (5) imply that only single solutions are found, as the global feedback term (5) obtains a unique limit. This problem has already been demonstrated by [25].

Two-stage MMO algorithms

To overcome the problem of classical PSO, researches have proposed two extended PSO methods: niching migratory multi-swarm optimizer (NMMSO) [26] and multi-charged particle swarm optimization (mCPSO) [31]. These methods, called two-stage niching methods [28], separate the particles into sub-swarms by a clustering algorithm in the first stage, and perform sub-swarming searches of the optimum by the classical PSO in the second stage. Alternative two-stage methods, which utilize an evolutionary strategy in the second stage, include covariance matrix self-adaptation with repelling subpopulations (RS-CMSA) [27], the Hill-Valley evolutionary algorithm (HillVallEA) [28], and its improved variant (HillVallEA19) [29]. These two-stage methods also utilize additional sub-procedures, namely, a restart scheme in NMMSO, mCPSO, RS-CMSA and HillVallEA and a taboo archive in RS-CMSA.

Unfortunately, the complexity of these two-stage methods is daunting to engineers wishing to implement search algorithms in their production items. Although the source codes of these methods are available, they might not be directly implementable in the production items of a firm. In such situations, engineers should understand and implement the algorithms by themselves, as reported in [30].

One-stage MMO algorithms

Simpler niching methods than the two-stage methods are also available. Some of these algorithms are based on PSO; specifically, RPSO, fitness Euclidean-distance ratio PSO (FERPSO) [22], locally informed particle swarm (LIPS) [24], and species-based PSO (SPSO) [23]. Another niching method, called the niching gravitational search algorithm (NGSA) [32] uses gravitational forces. In the present study, these methods are called one-stage methods because they include no clustering algorithms. Table 1 summarizes the characteristics (computational complexity, number of tuning parameters, and algorithm type) of these one-stage methods.

Focusing on the computational complexity, these existing methods can be broadly classified into two groups: RPSO with complexity $O(N)$; and FERPSO, LIPS, SPSO and NGSA with

Table 1. Characteristics of the existing one-stage niching methods.

	RPSO	FERPSO	LIPS	SPSO	NGSA
Complexity:					
$O(N)$ [a]	✓	n/a	n/a	n/a	n/a
$O(N^2+\alpha)$ [b]	n/a	✓	✓	✓	✓
The number of Parameters:					
	3	3	4	4	5
Type:					
Based on PSO	✓	✓	✓	✓	n/a
Using gravitational force	n/a	n/a	n/a	n/a	✓

"✓" and "n/a" indicate applicable and not applicable, respectively.

[a]the same as classical PSO

[b]obtained by calculating the Euclidean norm between the particles and by selecting or sorting of particles, where $\alpha = N$ or $N \log N$. N is the number of particles.

https://doi.org/10.1371/journal.pone.0248470.t001 complexity $O(N^2 + \alpha)$, where $\alpha = N, N \log N$. In the method of the former group (RPSO), the complexity is equivalent to that of the classical PSO (4). In the methods of the latter group, which calculate the Euclidean norm between particles, the complexity is required to be at least $O(N^2)$. In addition, FERPSO adds a complexity $O(N)$ for particle selection, whereas LIPS, SPSO and NGSA add a complexity $O(N \log N)$ for particle sorting.

With the exception of RPSO and FERPSO, the number of tuning parameters is one or two higher in these methods than in the classical PSO. The additional parameters are required for sorting the particles.

The comparison confirms RPSO as the simplest niching method, but RPSO is known to deliver poorer performance than the other methods [25].

The above discussion highlights the drawbacks of the existing high-performance niching methods, namely, the difficulties in understanding and implementing two-stage methods, and the high computational complexity and large number of tuning parameters in one-stage methods.

GPSA

To resolve the above drawbacks, the present study proposes a new and simple niching method called GPSA. The basic idea is that, if the particles in PSO attract each other, they can autonomously organize into sub-swarms and search the multiple global optimum without any additional procedures. Specifically, the classical global feedback term (5) is replaced with a term that introduces inverse-square gravitational forces between the particles, i.e.,

$$\boldsymbol{a}_i(t) = \sum_{k \neq i} \frac{1}{||\boldsymbol{d}_{ki}(t)||^2} \boldsymbol{u}_{ki}(t), \tag{6}$$

$$\boldsymbol{u}_{ki}(t) = \frac{\boldsymbol{d}_{ki}(t)}{||\boldsymbol{d}_{ki}(t)||}, \ \boldsymbol{d}_{ki}(t) = \boldsymbol{x}_k(t) - \boldsymbol{x}_i(t), \tag{7}$$

where $||\cdot||$ denotes the Euclidean norm, $\boldsymbol{d}_{ki}(t)$ is a displacement vector between the positions of the ith and kth particles, and $\boldsymbol{u}_{ki}(t)$ is the normalized vector of $\boldsymbol{d}_{ki}(t)$. Algorithm 1 shows the pseudocode of our GPSA.

The novelty of our GPSA is summarized below.

- Our method is a purely dynamical one-stage method, i.e., the particles are updated only by a dynamical system with no additional procedures. In contrast, the existing two-stage methods require a clustering algorithm, a restart scheme, or a taboo archive [26–29, 31].
- Unlike the existing one-stage methods [22–24, 32], our method requires no algorithmic procedure for selecting the social or nearest best(s).

Algorithm 1 GPSA

```
 Input $f$, $d$, $N$, $t_{\max}$, $\omega$, $c_1$, $c_2$
 Output The set of final personal bests {$\boldsymbol{pb}_1(t_{\max})$, $\boldsymbol{pb}_2(t_{\max})$, ..., $\boldsymbol{pb}_N(t_{\max})$}
1: procedure GPSA
2:   $t = 0$
3:   Initialize particles' position $\boldsymbol{x}_i(t) \in \mathbb{R}^d, i = 1, 2, \ldots, N$
4:   Initialize particles' velocity $\boldsymbol{v}_i(t) \in \mathbb{R}^d$
5:   Set personal bests via $\boldsymbol{pb}_i(t) = \boldsymbol{x}_i(t)$
6:   while $t < t_{\max}$ do
7:     for $i = 1, \ldots, N$ do
8:       Update particles' position and velocity via (4) with (6)
```

```
9:       if $f(\boldsymbol{x}_i(t+1)) > f(\boldsymbol{pb}_i(t))$ then        ▷ Update personal best
10:        $\boldsymbol{pb}_i(t+1) = \boldsymbol{x}_i(t+1)$
11:      else
12:        $\boldsymbol{pb}_i(t+1) = \boldsymbol{pb}_i(t)$
13:    $t = t + 1$
```

Therefore, our GPSA is advantaged by simplicity. Our GPSA can be easily understood and implemented by non-experts in optimization algorithms.

Fig 1 compares the characteristics (computational complexity and number of tuning parameters) of our GPSA with those of the existing one-stage methods. As evidenced in the figure, our GPSA has the same number of tuning parameters $(\omega, c_1, c_2) \in \mathbb{R}^3$ as RPSO and FERPSO, and fewer parameters than other one-stage methods. In addition, the complexity of our GPSA is $O(N^2)$, obtained by summing the complexities $O(N(N-1))$ of (6) and $O(N)$ of (4). This complexity is higher than in the $O(N)$ method, but lower than in the $O(N^2+\alpha)$ methods, where $\alpha = N$ or $N \log N$. Therefore, in terms of the number of tuning parameters and complexity, our GPSA is simpler than these existing one-stage methods except RPSO.

Although the nominal performance of our GPSA is apparently inferior to that of RPSO in Fig 1, the actual performances of GPSA and RPSO are comparable. Furthermore, after minor improvements, our GPSA significantly outperforms RPSO. This performance will be demonstrated after the following examples.

One-dimensional examples

Our proposed GPSA dynamically organizes the niching behavior of the sub-swarms. This mechanism is clarified through illustrative examples in the present and following sub-sections.

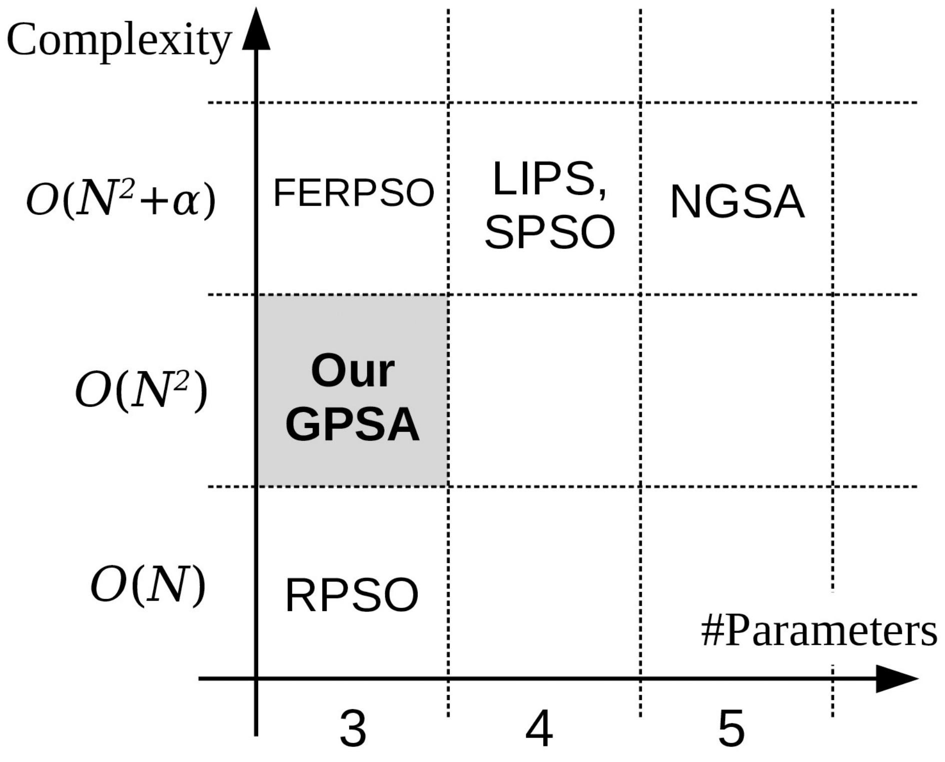

Fig 1. Comparison of the characteristics (computational complexity and number of tuning parameters, #parameters) of our proposed GPSA and existing one-stage niching methods.

https://doi.org/10.1371/journal.pone.0248470.g001

Fig 2 shows the motions produced by a two-particle GPSA solving a one-dimensional UMO problem, $f(x) = -x^2$ (an inverted sphere). Here the GPSA parameters were set to $\omega = 0.729$ and $c_1 = 1.49445$. These values are known to prevent particle divergence in the classical PSO [38], and are used even in the existing niching PSOs [22, 25]. The newly introduced parameter c_2 was set to 0.05.

During the initial phase ($0 \leq t \leq 23$), the particles were mutually attracted by gravity. At some close enough distance ($t = 23$), they appeared to repel one another. Such particle repulsion behavior, called a repulsive flip here, is considered as a gravity-induced motion. Roughly speaking, ignoring the randomness and setting $\omega = c_1 = 0$ in (4a), the one-dimensional gravitational force (6) becomes

$$a(t) := a_1(t) = -a_2(t) = \frac{x_2(t) - x_1(t)}{d(t)^3},\ d(t) = |x_2(t) - x_1(t)|, \tag{8}$$

so the positions of the two close particles are updated as

$$x_1(t+1) = x_1(t) + c_2 a(t),\ x_2(t+1) = x_2(t) - c_2 a(t). \tag{9}$$

If $d(t) = \sqrt[3]{c_2}$, the particles are swapped as

$$x_1(t+1) = x_1(t) + c_2 a(t) = x_1(t) + c_2 \frac{x_2(t) - x_1(t)}{c_2} = x_2(t). \tag{10}$$

Similarly, we obtain

$$x_2(t+1) = x_1(t), \tag{11}$$

as shown in Fig 3(A). In addition, when

$$d(t) = \sqrt[3]{c_2} + \epsilon_+,\ \ 0 < \epsilon_+ \ll \sqrt[3]{c_2}, \tag{12}$$

there is what we call an attractive flip (Fig 3(B)), because

$$d(t+1) = |x_2(t+1) - x_1(t+1)| = |x_2(t) - x_1(t) - 2c_2 a(t)| = \left|\frac{d(t)^3 - 2c_2}{d(t)^2}\right| < \sqrt[3]{c_2}. \tag{13}$$

In contrast,

$$d(t) = \sqrt[3]{c_2} - \epsilon_+ \tag{14}$$

generates a repulsive flip because $d(t+1) > \sqrt[3]{c_2}$ (Fig 3(C)). From (13), we established that $d(t+1) = |\frac{d(t)^3 - 2c_2}{d(t)^2}| \to \infty\ \ (d(t) \to 0)$. In other words, the repulsive flip strengthens as $d(t)$ approaches 0.

After the repulsive flip at $t = 23$ (Fig 2(A)), each particle underwent different damped oscillations around $pb_i(t)$, because the second term in (4a) imposes a linear restoring force. After the second repulsive flip at $t = 40$, repeated mutual repulsive flips of the particles were followed by individual dumped oscillations (Fig 2(B)). During this time, the particle's personal bests, $pb_1(t)$ and $pb_2(t)$, rapidly converged to the problem optimum $x = 0$ (Fig 2(C)), confirming that our GPSA can solve the UMO problem.

The search motions of our GPSA also effectively solve MMO problems. Fig 4 shows the motions of six particles searching for the three global optima of the function $f(x) = \max\{-|x-4|^2, -|x|^2, -|x+4|^2\}$. The particles successfully found all three solutions.

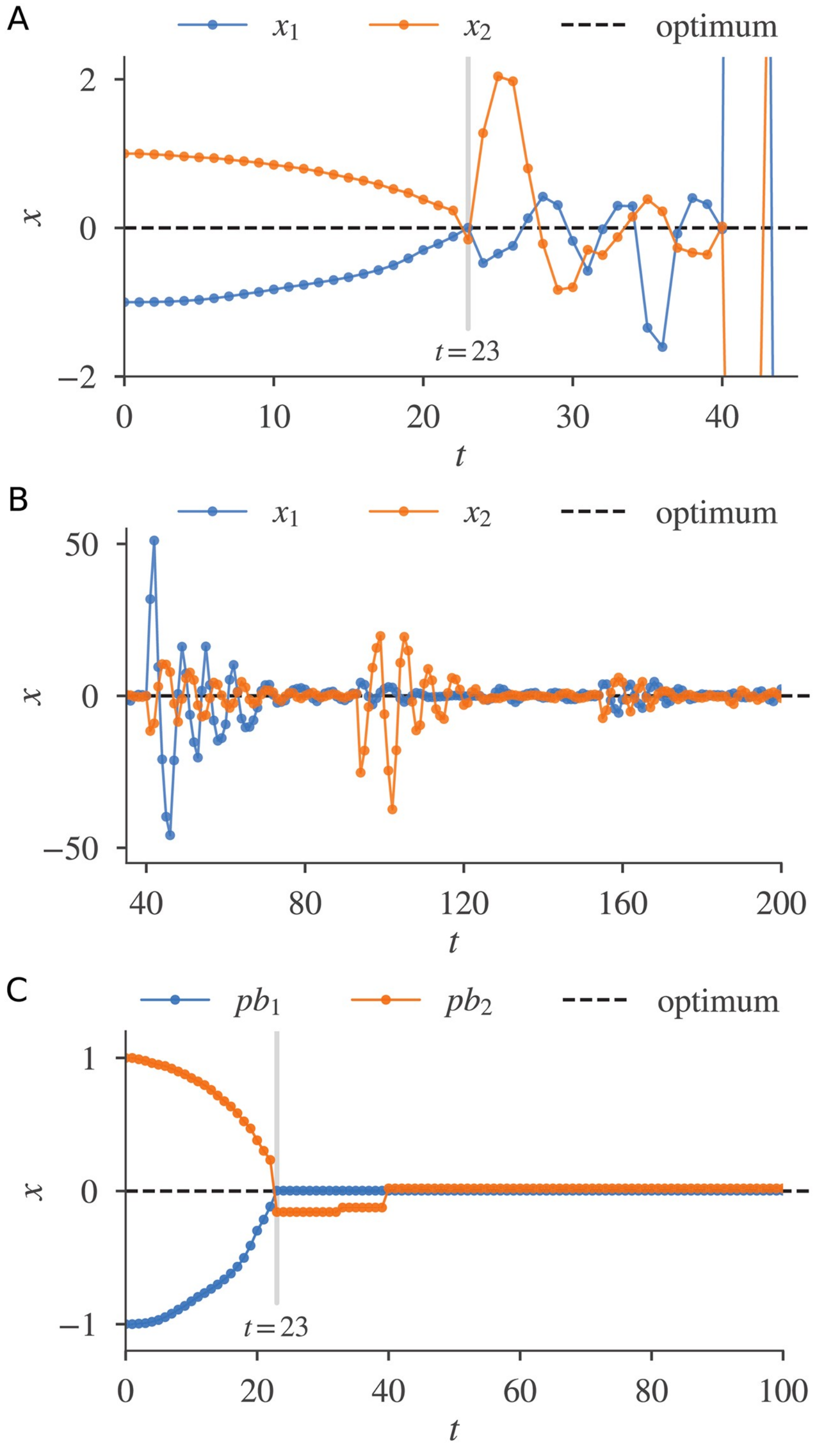

Fig 2. A two-particle GPSA solving a one-dimensional UMO function. (A) Particle positions ($0 \leq t \leq 45$). (B) Particle positions ($35 \leq t \leq 200$). (C) Personal bests.

https://doi.org/10.1371/journal.pone.0248470.g002

Initially, our GPSA autonomously and dynamically generated multiple sub-swarms near two of the optima (Fig 4(A)). A repulsive flip between $x_5(t)$ and $x_6(t)$ at $t = 11$ then repelled $x_6(t)$, which underwent a large-amplitude damped oscillation. This repulsion and oscillation process helped the personal best $pb_6(t)$ to find the distant optimum $x = 0$ (Fig 4(B)).

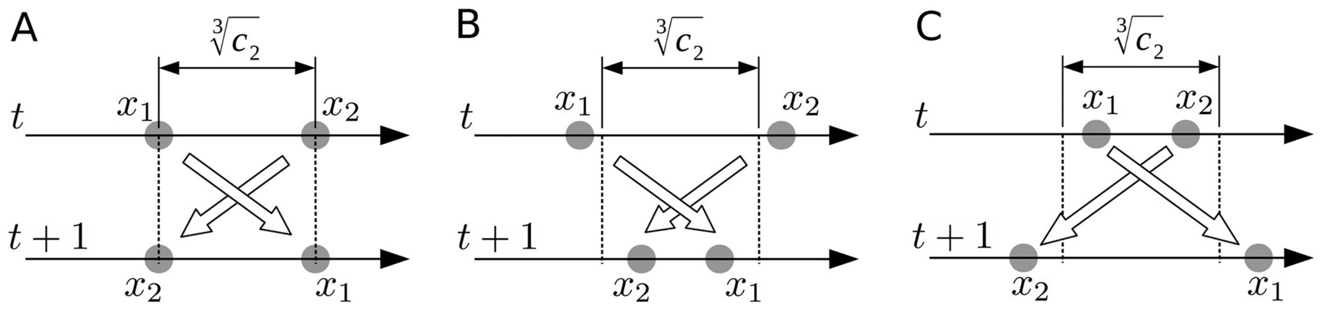

Fig 3. Different gravity-induced behaviors of two nearby particles. (A) Swap. (B) Attractive flip. (C) Repulsive flip.

https://doi.org/10.1371/journal.pone.0248470.g003

Two-dimensional example

Our proposed GPSA can also solve a two-dimensional MMO problem involving a two-dimensional Himmelblau function [33]. Fig 5 shows snapshots of a GPSA simulation run, starting from particles that were randomly placed in the right half-plane (Fig 5(A)). This initial spatial bias was designed to challenge the search for global optima in the left half-plane.

As clarified in Fig 5, the particles again autonomously and dynamically formed multiple sub-swarms. While most of the particles remained in the right half-plane, some escaped to the

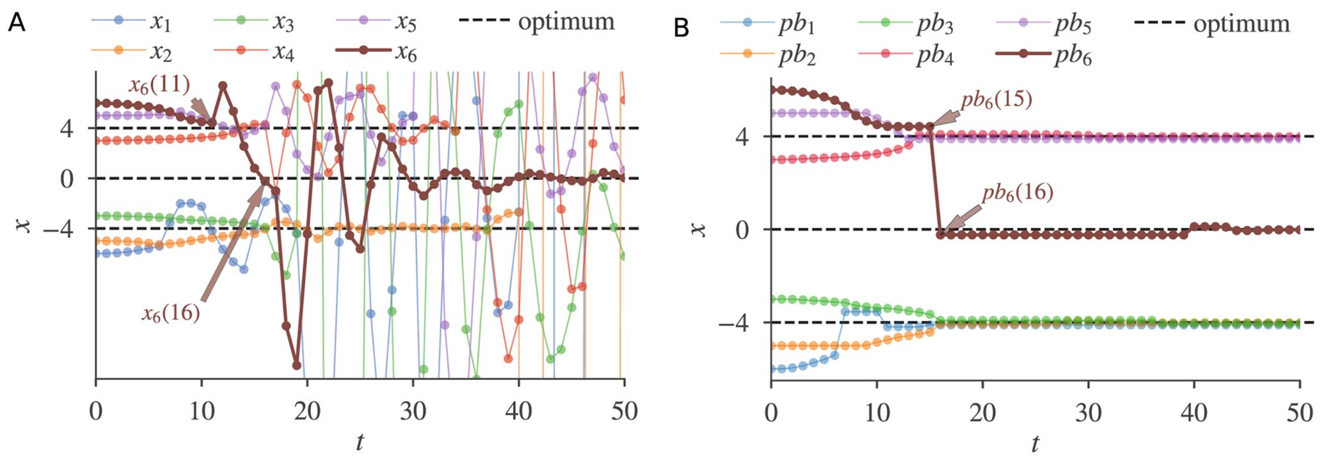

Fig 4. A six-particle GPSA solving a one-dimensional MMO function with three global optima. (A) Particle positions. (B) Personal bests.

https://doi.org/10.1371/journal.pone.0248470.g004

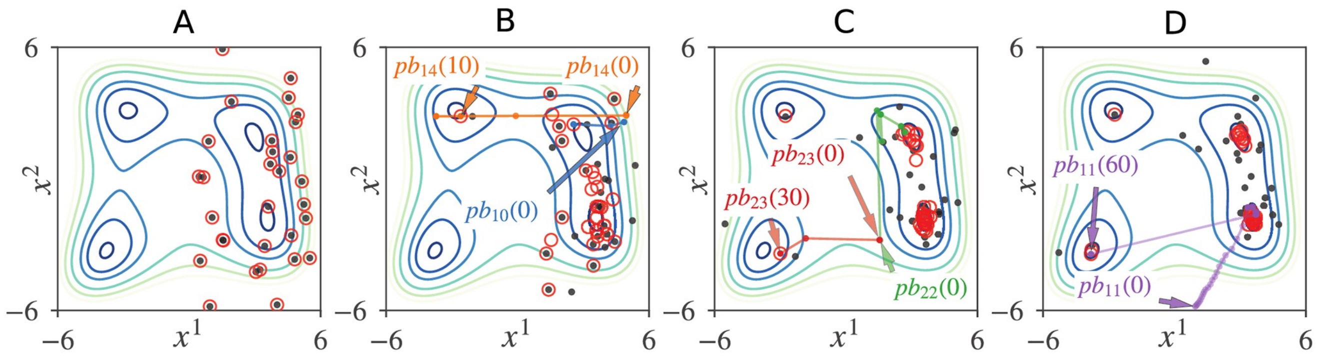

Fig 5. Snapshots of a GPSA run that found all four global optima of the Himmelblau function. x^j denotes the jth axis of position vector x, and the black and red circles indicate the positions and personal bests of particles, respectively. (A) $t = 0$. (B) $t = 10$. (C) $t = 30$. (D) $t = 60$.

https://doi.org/10.1371/journal.pone.0248470.g005

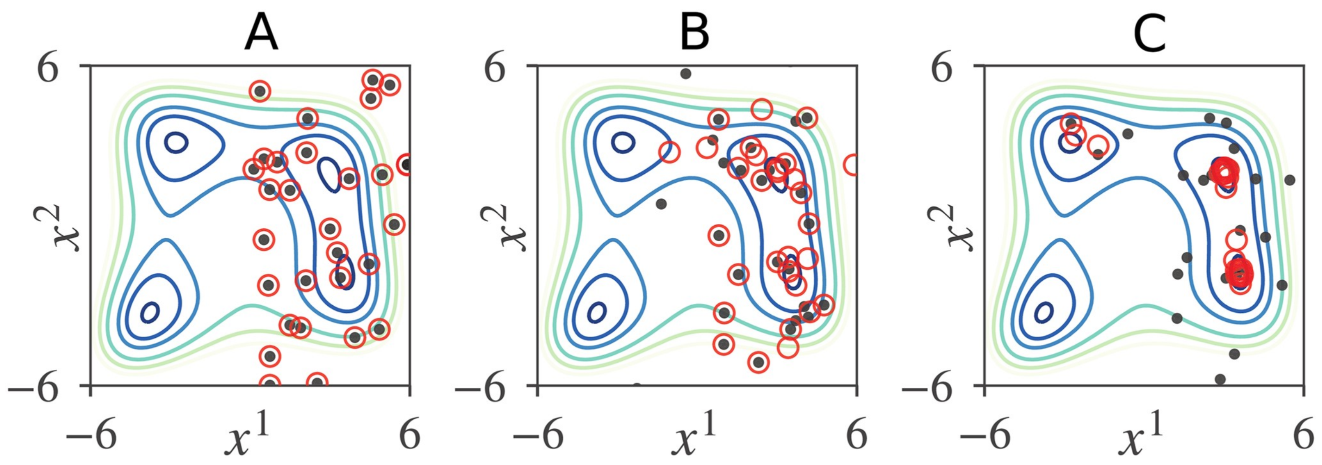

Fig 6. Snapshots of a GPSA run that found only three global optima of the Himmelblau function. (A) $t = 0$. (B) $t = 10$. (C)$t = 60$.

https://doi.org/10.1371/journal.pone.0248470.g006

left half-plane through the repulsive flips. In this run, the first repulsive flip occurred immediately (at $t = 1$) between $\boldsymbol{x}_{10}(t)$ and $\boldsymbol{x}_{14}(t)$, causing $\boldsymbol{pb}_{14}(t)$ to approach one of the distant global optima by $t = 10$ (Fig 5(B)). The $\boldsymbol{pb}_{23}(t)$ and $\boldsymbol{pb}_{11}(t)$ then found the remaining distant global optimum by $t = 60$ (Fig 5(C) and 5(D)).

Because our GPSA dynamics include random fluctuations, all global optima are not guaranteed to be found within a finite amount of time (see Fig 6). Nevertheless, as demonstrated in the following sections, the performances of our GPSA and our GPSA with minor improvements were comparable to and considerably superior to those of the existing methods, respectively.

Performance evaluation

Experimental setup

This section evaluates our GPSA on the twenty benchmark functions of the CEC 2013 niching-method competition [33]. These benchmark functions were proposed in 2013. However, they are still the latest ones because they have been utilized by the CEC and GECCO niching competitions held between 2013 and 2020. The benchmark functions are listed in Table 2. The terms d and N^{go} were introduced in (1) and (2), respectively, and D is the domain of the function. These benchmark functions are derived from twelve base-test functions, which are broadly classified into two types: functions with local optima and functions with no local optima. The former functions are the Five-Uneven-Peak Trap (f_1), the Uneven Decreasing Maxima (f_3), the Six-Hump Camel Back (f_5), Shubert (f_6, f_8), and the Composite Functions 1 (f_{11}), 2 (f_{12}), 3 ($f_{13}, f_{14}, f_{16}, f_{18}$), and 4 ($f_{15}, f_{17}, f_{19}, f_{20}$). The latter functions include the Equal Maxima (f_2), Himmelblau (f_4), Vincent (f_7, f_9) and the Modified Rastrigin (f_{10}). These functions over domain D are formally defined in [33]. For the external domain of D, the value of these functions was set to $-10^{10} \ll f_i(x), \forall \boldsymbol{x} \mid \boldsymbol{x} \in D, i = 1, \cdots, 20$ as the penalty value because our GPSA's particle can exceed the domain D due to the repulsive flip.

Following [33], the number of function evaluations in a single run was set to N^{fes} as listed in Table 2. The total number of particles was set to $N = 500$ for f_8 and f_9, which have many global optima, and to $N = 50$ for the remaining functions, as adopted in the existing one-stage methods, RPSO [25] and FERPSO [22]. Each of these particles was updated until $t_{\max} = N^{\mathrm{fes}}/N$. The number of runs was set to $N^{\mathrm{run}} = 100$, at least double the number of runs in previous studies [22, 24, 25, 33].

Table 2. Benchmark functions.

f	Base-test function	d	N^{go}	D	N^{fes}
f_1	Five-Uneven-Peak Trap	1	2	$[0, 30]$	5×10^4
f_2	Equal Maxima	1	5	$[0, 1]$	5×10^4
f_3	Uneven Decreasing Maxima	1	1	$[0, 1]$	5×10^4
f_4	Himmelblau	2	4	$[-6, 6]^2$	5×10^4
f_5	Six-Hump Camel Back	2	2	$[-1.9, 1.9] \times [-1.1, 1.1]$	5×10^4
f_6	Shubert	2	18	$[-10, 10]^2$	2×10^5
f_7	Vincent	2	36	$[0.25, 10]^2$	2×10^5
f_8	Shubert	3	81	$[-10, 10]^3$	4×10^5
f_9	Vincent	3	216	$[0.25, 10]^3$	4×10^5
f_{10}	Modified Rastrigin	2	12	$[0, 1]^2$	2×10^5
f_{11}	Composite Function 1	2	6	$[-5, 5]^2$	2×10^5
f_{12}	Composite Function 2	2	8	$[-5, 5]^2$	2×10^5
f_{13}	Composite Function 3	2	6	$[-5, 5]^2$	2×10^5
f_{14}	Composite Function 3	3	6	$[-5, 5]^3$	4×10^5
f_{15}	Composite Function 4	3	8	$[-5, 5]^3$	4×10^5
f_{16}	Composite Function 3	5	6	$[-5, 5]^5$	4×10^5
f_{17}	Composite Function 4	5	8	$[-5, 5]^5$	4×10^5
f_{18}	Composite Function 3	10	6	$[-5, 5]^{10}$	4×10^5
f_{19}	Composite Function 4	10	8	$[-5, 5]^{10}$	4×10^5
f_{20}	Composite Function 4	20	8	$[-5, 5]^{20}$	4×10^5

d is the number of dimensions, N^{go} is the number of global optima, D is the domain of the function, and N^{fes} is the number of allowed function evaluations.

https://doi.org/10.1371/journal.pone.0248470.t002

Performance metrics

The performance was evaluated by two metrics used in [33]. The first was the peak ratio (PR), which measures the average fraction of the global optima found per run and is given by

$$\mathrm{PR} = \frac{1}{N^{\mathrm{run}}} \sum_{k=1}^{N^{\mathrm{run}}} \frac{n_k^{\mathrm{go}}}{N^{\mathrm{go}}}, \tag{15}$$

where n_k^{go} is the number of global optima found during the kth run. The n_k^{go} was determined by a standard method provided in [33], which considers that a new global optimum is detected when $\boldsymbol{pb}_i(t)$ in (4) satisfies the following two conditions. First, the $\boldsymbol{pb}_i(t)$ should be further than any of the already discovered global optima in terms of the distance specified in [33]. Second, the difference between the fitness values of the $\boldsymbol{pb}_i(t)$ and the global optima provided in [33] should be less than the accuracy level ϵ. Following [33], the accuracy levels in this study were varied as $\epsilon \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

The second metric was the success rate (SR), which measures the probability of finding all global optima during the same run. The measure is given by

$$\mathrm{SR} = \frac{N^{\mathrm{succ}}}{N^{\mathrm{run}}}, \tag{16}$$

where N^{succ} is the number of runs in which all the global optima are found.

The benchmark functions and performance evaluation were implemented in the code provided by the CEC 2013 competition organizers. Obtainable from https://github.com/mikeagn/CEC2013.

Table 3. PR results of our GPSA with different parameters ($c_2 = 10^{-2}$, 10^{-4}, and 10^{-6}).

	f_1, $D = [0, 30]$			f_2, $D = [0, 1]$			f_3, $D = [0, 1]$			f_4, $D = [-6, 6]^2$		
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}
10^{-1}	0.895	**1.000**	0.995	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.975	0.008
10^{-2}	0.210	0.610	**0.970**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.948	0.005
10^{-3}	0.020	0.055	**0.615**	0.996	**1.000**	**1.000**	0.980	**1.000**	**1.000**	0.640	**0.833**	0.003
10^{-4}	0.005	0.010	**0.335**	0.812	0.998	**1.000**	0.770	0.990	**1.000**	0.100	**0.583**	0.003
10^{-5}	0.000	0.000	**0.190**	0.422	0.894	**0.996**	0.320	0.750	**1.000**	0.018	**0.158**	0.000
	f_5, $D = [-1.9, 1.9] \times [-1.1, 1.1]$			f_6, $D = [-10, 10]^2$			f_7, $D = [0.25, 10]^2$			f_8, $D = [-10, 10]^2$		
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}
10^{-1}	**1.000**	**1.000**	0.945	**0.173**	0.039	0.003	**0.480**	0.345	0.100	**0.005**	0.000	0.000
10^{-2}	**1.000**	**1.000**	0.895	**0.162**	0.036	0.003	**0.480**	0.310	0.049	**0.000**	**0.000**	**0.000**
10^{-3}	0.990	**1.000**	0.880	**0.141**	0.027	0.003	**0.480**	0.302	0.036	**0.000**	**0.000**	**0.000**
10^{-4}	0.455	**1.000**	0.860	**0.101**	0.018	0.002	**0.467**	0.295	0.032	**0.000**	**0.000**	**0.000**
10^{-5}	0.065	0.775	**0.830**	**0.039**	0.016	0.001	**0.309**	0.289	0.031	**0.000**	**0.000**	**0.000**
	f_9, $D = [0.25, 10]^3$			f_{10}, $D = [0, 1]^2$			f_{11}, $D = [-5, 5]^2$			f_{12}, $D = [-5, 5]^2$		
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}
10^{-1}	**0.360**	0.212	0.024	0.981	**0.997**	0.975	**0.667**	0.425	0.017	**0.570**	0.200	0.013
10^{-2}	**0.344**	0.139	0.001	0.422	**0.996**	0.975	**0.547**	0.422	0.015	**0.289**	0.195	0.013
10^{-3}	**0.219**	0.127	0.000	0.066	0.647	**0.975**	0.128	**0.383**	0.015	0.091	**0.141**	0.009
10^{-4}	0.034	**0.114**	0.000	0.008	0.083	**0.933**	0.013	**0.153**	0.012	0.016	**0.071**	0.005
10^{-5}	0.001	**0.047**	0.000	0.001	0.006	**0.388**	0.002	**0.028**	0.005	0.000	**0.056**	0.000
	f_{13}, $D = [-5, 5]^2$			f_{14}, $D = [-5, 5]^3$			f_{15}, $D = [-5, 5]^3$			f_{16}, $D = [-5, 5]^5$		
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}
10^{-1}	**0.657**	0.057	0.003	**0.113**	0.002	0.000	**0.101**	0.000	0.000	**0.003**	0.000	0.000
10^{-2}	**0.383**	0.055	0.003	**0.070**	0.002	0.000	**0.059**	0.000	0.000	**0.002**	0.000	0.000
10^{-3}	**0.083**	0.050	0.003	**0.052**	0.002	0.000	**0.031**	0.000	0.000	**0.002**	0.000	0.000
10^{-4}	0.013	**0.037**	0.003	**0.013**	0.000	0.000	**0.005**	0.000	0.000	**0.000**	**0.000**	**0.000**
10^{-5}	0.000	**0.022**	0.003	**0.000**	**0.000**	**0.000**	**0.001**	0.000	0.000	**0.000**	**0.000**	**0.000**
	$f_{17}, f_{18}, f_{19}, f_{20}$, $D = [-5, 5]^d$											
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}									
10^{-1}	**0.000**	**0.000**	**0.000**	Average								
10^{-2}	**0.000**	**0.000**	**0.000**									
10^{-3}	**0.000**	**0.000**	**0.000**	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}						
10^{-4}	**0.000**	**0.000**	**0.000**	0.249	**0.269**	0.222						
10^{-5}	**0.000**	**0.000**	**0.000**									

D is the function domain as listed in Table 2. The best values for each function and accuracy-level among the c_2 values are indicated in bold.

https://doi.org/10.1371/journal.pone.0248470.t003

Effects of the parameter c_2

The performance sensitivity of our GPSA to the parameter c_2 was investigated while the other parameters were fixed at $\omega = 0.729$ and $c_1 = 1.49445$.

Table 3 lists the PR-values obtained across all benchmark functions and the accuracy levels for $c_2 = 10^{-2}$, 10^{-4}, and 10^{-6}, where *D* is the function domain as listed in Table 2. The best values for each function and accuracy-level are highlighted in bold. The bottom section of the table summarizes the averaged *PR* obtained by summing the *PR*-values of all functions and accuracy levels and dividing by the total number of values.

At the accuracy level $\epsilon = 10^{-5}$, the PR of the functions with small domains (f_1, f_2, f_3, f_5, and f_{10}) increased with decreasing c_2, indicating that the exploitation performance of our GPSA improved as c_2 decreased. Note that as the accuracy level ϵ decreases, a stricter requirement is imposed on the exploitation performance [39].

In contrast, at the accuracy level $\epsilon = 10^{-1}$, the PR-values of the large-domain functions (f_4, f_6, f_7, f_8, f_9, and $f_{11}, \cdots, f_{16}$) increased with c_2, indicating that the exploration performance improved as c_2 increased.

The mechanism of these improvements can be explained by the effect of the gravitational force described in the one-dimensional example. Increasing the c_2 increases the distance between the particles generating the repulsive flip ($d(t) = \sqrt[3]{c_2} - \epsilon_+$ in (14)), elevating the frequency of repulsive flips and reducing that of the attractive motions.

These results clearly demonstrate that the balance between the exploitation and exploration performance of our GPSA can be controlled by c_2; specifically, a large c_2 is suitable for a global searching over a large domain, whereas a small c_2 favors local searching over a small domain.

Performance comparison with the existing one-stage methods

Next, the performance of our GPSA was compared with those of the existing methods, namely RPSO and FERPSO. These methods were selected for the following reasons. First, they are one-stage methods like our GPSA. Second, they replace the global feedback term (5) of the classical PSO with another term, similarly to our GPSA. Finally, they have the same number of tuning parameters as our GPSA (see Fig 1).

In RPSO, (5) is replaced by

$$\boldsymbol{a}_i(t) = \boldsymbol{lb}_i(t) - \boldsymbol{x}_i(t), \tag{17}$$

where $\boldsymbol{lb}_i(t) \in \mathbb{R}^d$ is the local best of the ith particle at iteration t. The local best is the highest-cost position found by the particle or one of its neighbors, i.e., the best among the personal bests $\boldsymbol{pb}_{i-1}(t)$, $\boldsymbol{pb}_i(t)$, and $\boldsymbol{pb}_{i+1}(t)$.

In FERPSO, (5) is replaced with:

$$\boldsymbol{a}_i(t) = \boldsymbol{nb}_i(t) - \boldsymbol{x}_i(t), \tag{18}$$

where $\boldsymbol{nb}_i(t) \in \mathbb{R}^d$ again represents a neighborhood personal best, but selected by maximizing the Euclidean-distance ratio (FER):

$$\mathrm{FER}_{ji} = \alpha \cdot \frac{f(\boldsymbol{pb}_j(t)) - f(\boldsymbol{pb}_i(t))}{||\boldsymbol{pb}_j(t) - pb_i(t)||}, \tag{19}$$

where $\alpha = ||s||/(f(\boldsymbol{gb}(t)) - f(\boldsymbol{x}_w(t)))$ is a scaling factor, $||s||$ is the size of the search space [22], $\boldsymbol{x}_w(t)$ is the least-fitted particle in the current population, and $\boldsymbol{pb}_i(t)$ and $\boldsymbol{pb}_j(t)$ are the personal bests of the ith and jth particles, respectively.

In this study, the tuning parameters of RPSO and FERPSO were set to $\omega = 0.729$ and $c_1 = c_2 = 1.49445$ consistent with [22, 25]. The other parameters and experimental conditions were consistent with those in the previous section.

Table 4 lists the SR-values in (16) obtained by our GPSA and the compared methods. The best and averaged values are indicated and summarized, as described in Table 3.

On the small-domain function f_{10}, our GPSA obtained remarkably higher SR-values than the compared methods. In particular, for $\epsilon = 10^{-1}$, 10^{-2}, 10^{-3} and 10^{-4} and $c_2 = 10^{-6}$, the SR-value of our GPSA exceeded 0.4, versus $SR = 0$ in the compared methods. On the other small-domain functions f_2 and f_3, the SR-values of our GPSA with $c_2 = 10^{-6}$ either outperformed or equaled those of the compared methods. These results showed that when c_2 is relatively small, the exploitation performance of our GPSA is comparable to those of other methods.

Table 4. SR results of our GPSA with $c_2 = 10^{-2}$, 10^{-4}, and 10^{-6}, RPSO, and FERPSO.

	f_1, $D = [0, 30]$					f_2, $D = [0, 1]$					f_3, $D = [0, 1]$				
	GPSA			RPSO	FERPSO	GPSA			RPSO	FERPSO	GPSA			RPSO	FERPSO
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}			$c_2 = 10^{-2}$,	10^{-4},	10^{-6}			$c_2 = 10^{-2}$,	10^{-4},	10^{-6}		
10^{-1}	0.80	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.84	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
10^{-2}	0.03	0.32	0.94	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.84	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
10^{-3}	0.00	0.01	0.39	**1.00**	**1.00**	0.98	**1.00**	**1.00**	0.84	0.90	0.98	**1.00**	**1.00**	**1.00**	**1.00**
10^{-4}	0.00	0.00	0.13	**1.00**	**1.00**	0.29	0.99	**1.00**	0.84	0.90	0.77	0.99	**1.00**	**1.00**	**1.00**
10^{-5}	0.00	0.00	0.03	**1.00**	**1.00**	0.01	0.52	**0.98**	0.79	0.89	0.32	0.75	**1.00**	**1.00**	**1.00**
	f_4, $D = [-6, 6]^2$					f_5, $D = [-1.9, 1.9] \times [-1.1, 1.1]$					f_{10}, $D = [0, 1]^2$				
	GPSA			RPSO	FERPSO	GPSA			RPSO	FERPSO	GPSA			RPSO	FERPSO
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}			$c_2 = 10^{-2}$,	10^{-4},	10^{-6}			$c_2 = 10^{-2}$,	10^{-4},	10^{-6}		
10^{-1}	**1.00**	0.92	0.00	0.94	0.87	**1.00**	**1.00**	0.89	**1.00**	**1.00**	0.79	**0.96**	0.71	0.00	0.00
10^{-2}	**1.00**	0.84	0.00	0.77	0.87	**1.00**	**1.00**	0.79	**1.00**	**1.00**	0.00	**0.95**	0.71	0.00	0.00
10^{-3}	0.18	0.46	0.00	0.63	**0.87**	0.98	**1.00**	0.77	**1.00**	**1.00**	0.00	0.00	**0.71**	0.00	0.00
10^{-4}	0.00	0.09	0.00	0.61	**0.87**	0.21	**1.00**	0.74	**1.00**	**1.00**	0.00	0.00	**0.42**	0.00	0.00
10^{-5}	0.00	0.00	0.00	0.58	**0.87**	0.00	0.58	0.69	**1.00**	**1.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	$f_6, f_7, f_8, f_9, f_{11}, \cdots, f_{20}$														
	GPSA			RPSO	FERPSO										
ϵ	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}												
10^{-1}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	Average									
10^{-2}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	GPSA			RPSO	FERPSO					
10^{-3}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	$c_2 = 10^{-2}$,	10^{-4},	10^{-6}							
10^{-4}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.14	0.19	0.19	0.23	**0.24**					
10^{-5}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**										

D is the function domain, as listed in Table 2. The best values for each function and accuracy-level among RPSO, FERPSO and our GPSA with $c_2 = 10^{-2}$, 10^{-4}, and 10^{-6} are highlighted in bold.

https://doi.org/10.1371/journal.pone.0248470.t004

On the large-domain functions, f_4 and f_5 with $\epsilon = 10^{-1}$ and 10^{-2} and $c_2 = 10^{-2}$, our GPSA performed at least as well as the compared methods. These results indicate that when c_2 is a relatively large, the exploration performance of our GPSA is comparable to those of other methods.

However, in terms of the averaged *SR* values, the other methods outperformed our GPSA. This degradation was primarily attributed to the dilemma of choosing between the exploitation and exploration performance in our GPSA.

Improvement of GPSA with a dynamic c_2

To resolve the exploration—exploitation dilemma of our GPSA, this section introduces a dynamic c_2 that further improves its performance.

Dynamic c_2 for our GPSA

Given our observations in the previous section, it is inferred that our GPSA's iteration should initially start with a large c_2 to enhance the exploration performance, and that c_2 should decrease as the iteration increments to improve the exploitation performance. Therefore, the present study proposes a method with a dynamic c_2, in which c_2 is provided as a nonlinear function of the iteration *t* based on the dynamic ω as used successfully in the classical PSOs

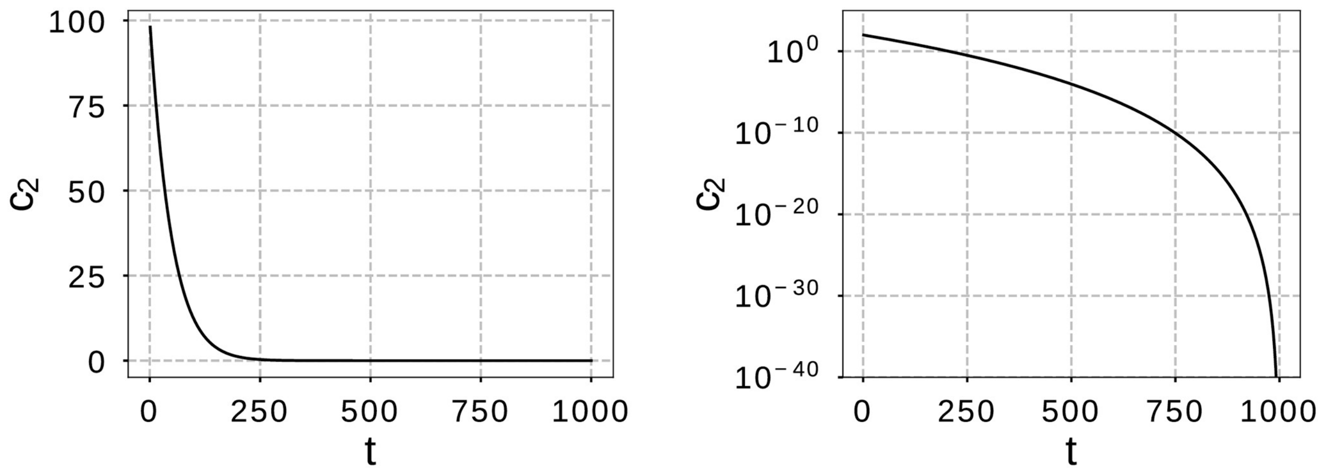

Fig 7. Dynamic c_2 as a function of iteration t, for $c_2^{\text{ini}} = 10^2$, n = 20, and $t_{\max}$ = 1000.

https://doi.org/10.1371/journal.pone.0248470.g007

[21, 40, 41], as follows:

$$c_2(t) = c_2^{\text{ini}}\left\{\frac{(t_{\max} - t)^n}{(t_{\max})^n}\right\}, \tag{20}$$

where c_2^{ini} is the initial c_2 at the beginning of a run and n is the nonlinear modulation index. In this study, these parameters were set to $c_2^{\text{ini}} = 10^2$ and n = 20 to cover the c_2 range analyzed in the previous section. Fig 7 shows the dynamic c_2 as a function of t for $t_{\max}$ = 1000, where the left graph is a linear plot and the right a semi-log plot. It is shown that the c_2 gradually decreased to zero as t increased, and c_2 used in Table 3 was covered. In this study, this modified GPSA is called a dynamic GPSA (DGPSA).

Our DGPSA proposed above is more complicated than our GPSA, however, it is still simpler than the existing one-stage methods in Fig 1, except for RPSO (FERPSO, LIPS, SPSO and NGSA), the reason being that the computational complexity of our DGPSA is $O(N^2 + 1)$ obtained by summing the complexities $O(N^2)$ of original GPSA and $O(1)$ of (20), and is even lower than FERPSO, LIPS, SPSO, and NGSA. In addition, the number of tuning parameters of our DGPSA, $(\omega, c_1, c_2^{\text{ini}}, n) \in \mathbb{R}^4$, is fewer than or equivalent to that of LIPS, SPSO and NGSA.

Performance of our DGPSA

Table 5 shows the *PR*-values obtained by our DGPSA and the compared methods.

First, these values of our DGPSA were compared with our GPSA results in Table 3. On f_6, f_8, and $f_{12}, \cdots, f_{20}$, our DGPSA obtained higher *PR*-values at all accuracy levels than our GPSA results. Even on $f_1, f_2, f_4, f_5, f_7, f_9, f_{10}$, and f_{11}, our DGPSA also had higher *PR*-values at the strictest accuracy level $\epsilon = 10^{-5}$. Furthermore, the averaged *PR*-value increased by more than 2.3 times. These results clearly demonstrate that our DGPSA successfully solves the dilemma found in our GPSA.

Furthermore, our DGPSA obtained higher PR-values than the compared methods on $f_2, f_4, f_6, f_7, f_9, \cdots, f_{16}$, and f_{19}. In particular, on f_6 and f_7, the *PR*-values of our DGPSA were at least 1.9 times higher than those of the compared methods. The averaged *PR*-values also indicated the superiority of our DGPSA.

Table 6 shows the SR-values obtained by our DGPSA, all of which were superior or equal to those of the compared methods shown in Table 4.

Table 5. PR results of our DGPSA, RPSO, and FERPSO.

	f_1			f_2			f_3			f_4		
ϵ	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO
10^{-1}	**1.000**	**1.000**	**1.000**	**1.000**	0.968	0.980	**1.000**	**1.000**	**1.000**	**1.000**	0.985	0.968
10^{-2}	**1.000**	**1.000**	**1.000**	**1.000**	0.968	0.980	**1.000**	**1.000**	**1.000**	**1.000**	0.940	0.968
10^{-3}	**1.000**	**1.000**	**1.000**	**1.000**	0.968	0.980	**1.000**	**1.000**	**1.000**	**1.000**	0.903	0.968
10^{-4}	**1.000**	**1.000**	**1.000**	**1.000**	0.968	0.980	**1.000**	**1.000**	**1.000**	**1.000**	0.895	0.968
10^{-5}	**1.000**	**1.000**	**1.000**	**1.000**	0.958	0.978	**1.000**	**1.000**	**1.000**	**1.000**	0.885	0.968
	f_5			f_6			f_7			f_8		
ϵ	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO
10^{-1}	**1.000**	**1.000**	**1.000**	**0.948**	0.486	0.409	**0.430**	0.194	0.195	0.594	**0.607**	0.374
10^{-2}	**1.000**	**1.000**	**1.000**	**0.948**	0.468	0.402	**0.430**	0.194	0.195	0.586	**0.591**	0.360
10^{-3}	**1.000**	**1.000**	**1.000**	**0.948**	0.449	0.398	**0.430**	0.191	0.195	0.575	**0.580**	0.345
10^{-4}	**1.000**	**1.000**	**1.000**	**0.948**	0.436	0.394	**0.427**	0.182	0.194	0.561	**0.571**	0.331
10^{-5}	**1.000**	**1.000**	**1.000**	**0.948**	0.426	0.394	**0.415**	0.175	0.194	0.531	**0.565**	0.317
	f_9			f_{10}			f_{11}			f_{12}		
ϵ	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO
10^{-1}	**0.313**	0.213	0.179	**0.990**	0.677	0.569	**0.667**	0.598	0.633	**0.728**	0.360	0.500
10^{-2}	**0.257**	0.191	0.179	**0.990**	0.666	0.569	**0.667**	0.588	0.633	**0.718**	0.353	0.499
10^{-3}	**0.212**	0.168	0.177	**0.990**	0.648	0.568	**0.667**	0.580	0.633	**0.708**	0.341	0.496
10^{-4}	**0.193**	0.153	0.174	**0.990**	0.636	0.568	**0.667**	0.578	0.633	**0.699**	0.334	0.496
10^{-5}	**0.186**	0.140	0.164	**0.990**	0.618	0.564	**0.667**	0.575	0.633	**0.694**	0.333	0.496
	f_{13}			f_{14}			f_{15}			f_{16}		
ϵ	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO
10^{-1}	**0.660**	0.533	0.548	**0.658**	0.553	0.528	**0.343**	0.288	0.289	**0.577**	0.537	0.322
10^{-2}	**0.660**	0.522	0.548	**0.658**	0.547	0.528	**0.341**	0.284	0.289	**0.570**	0.530	0.322
10^{-3}	**0.660**	0.510	0.548	**0.658**	0.543	0.528	**0.329**	0.281	0.288	**0.570**	0.527	0.322
10^{-4}	**0.660**	0.505	0.548	**0.657**	0.540	0.528	**0.321**	0.280	0.288	**0.570**	0.520	0.322
10^{-5}	**0.660**	0.505	0.548	**0.645**	0.538	0.528	**0.316**	0.279	0.288	**0.570**	0.517	0.322
	f_{17}			f_{18}			f_{19}			f_{20}		
ϵ	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO	DGPSA	RPSO	FERPSO
10^{-1}	0.219	**0.244**	0.125	0.167	**0.250**	0.177	**0.125**	0.114	0.036	0.125	**0.134**	0.090
10^{-2}	0.211	**0.241**	0.125	0.167	**0.247**	0.177	**0.125**	0.114	0.036	0.125	**0.134**	0.090
10^{-3}	0.206	**0.240**	0.125	0.167	**0.247**	0.175	**0.125**	0.114	0.036	0.125	**0.134**	0.090
10^{-4}	0.206	**0.240**	0.125	0.167	**0.247**	0.175	**0.125**	0.114	0.036	0.125	**0.134**	0.090
10^{-5}	0.206	**0.239**	0.124	0.167	**0.247**	0.175	**0.125**	0.114	0.036	0.125	**0.134**	0.090
	Average											
	DGPSA	RPSO	FERPSO									
	0.619	0.523	0.494									

The best values for each function and accuracy level among the algorithms are indicated in bold.

https://doi.org/10.1371/journal.pone.0248470.t005

The results above show that our DGPSA outperformed the compared methods in terms of *PR* and *SR*.

Statistical tests

The statistically significant tests were not yet conducted in the results shown above. Here, the number of functions is counted for which our DGPSA significantly outperformed the others, as was done in [39].

Table 6. SR results of our DGPSA for all benchmark functions and all accuracy levels.

ϵ	f_1	f_2	f_3	f_4	f_5	f_6	f_{10}	f_{12}	$f_7, f_8, f_9, f_{11}, f_{13}, \cdots, f_{20}$	
10^{-1}	1.000	1.000	1.000	1.000	1.000	0.320	0.880	0.030	0.000	Average
10^{-2}	1.000	1.000	1.000	1.000	1.000	0.320	0.880	0.030	0.000	
10^{-3}	1.000	1.000	1.000	1.000	1.000	0.320	0.880	0.020	0.000	0.311
10^{-4}	1.000	1.000	1.000	1.000	1.000	0.320	0.880	0.020	0.000	
10^{-5}	1.000	1.000	1.000	1.000	1.000	0.320	0.880	0.020	0.000	

https://doi.org/10.1371/journal.pone.0248470.t006

Consider the following sets:

$$\mathcal{N}_i(\epsilon) := \{n_k^{\mathrm{go}}(f_i, \epsilon) \mid 1 \leq k \leq 100\}, \tag{21}$$

$$\overline{\mathcal{N}}_i(\epsilon) := \{\overline{n_k^{\mathrm{go}}}(f_i, \epsilon) \mid 1 \leq k \leq 100\}, \tag{22}$$

where $n_k^{\mathrm{go}}(f_i, \epsilon)$ is n_k^{go} in (15) obtained by our DGPSA on the function f_i ($i = 1, \cdots, 20$) and the accuracy level ϵ, $\mathcal{N}_i(\epsilon)$ is a set of the $n_k^{\mathrm{go}}(f_i, \epsilon)$ for all k, and $\overline{n_k^{\mathrm{go}}}(f_i, \epsilon)$ and $\overline{\mathcal{N}}_i(\epsilon)$ are those of the compared method. Then, to classify the f_i, the following sets were introduced:

$$\mathcal{F}^+(\epsilon) := \{f_i \mid m_i > \overline{m}_i, \quad \text{and} \quad T_i = 1\}, \tag{23}$$

$$\mathcal{F}^-(\epsilon) := \{f_i \mid m_i < \overline{m}_i, \quad \text{and} \quad T_i = 1\}, \tag{24}$$

$$\mathcal{F}^0(\epsilon) := \{f_i \mid T_i = 0\}. \tag{25}$$

Here, m_i and $\overline{m}_i$ indicate the median of $\mathcal{N}_i(\epsilon)$ and $\overline{\mathcal{N}}_i(\epsilon)$, respectively. T_i becomes unity when there is a significant difference between m_i and $\overline{m}_i$, and otherwise zero. This significant difference was determined by the Wilcoxon rank-sum test [42] with a significance level of 0.05, as used in [39]. $\mathcal{F}^+(\epsilon)$ and $\mathcal{F}^-(\epsilon)$ are the sets of f_i on which our DGPSA was significantly superior and inferior to the compared method, respectively. Also, $\mathcal{F}^0(\epsilon)$ is the set of f_i where there was no significant difference between our DGPSA and the compared method.

Table 7 shows the resulting numbers of functions $\#\mathcal{F}^+(\epsilon)$, $\#\mathcal{F}^-(\epsilon)$, and $\#\mathcal{F}^0(\epsilon)$, where A denotes the number of elements of a set A. Here, Sum(ϵ) indicates the sum of $\#\mathcal{F}^+(\epsilon)$, $\#\mathcal{F}^-(\epsilon)$,

Table 7. Resulting numbers of functions in $\mathcal{F}^+(\epsilon)$, $\mathcal{F}^-(\epsilon)$, and $\mathcal{F}^0(\epsilon)$.

	RPSO versus DGPSA				
	$\epsilon = 10^{-1}$	$\epsilon = 10^{-2}$	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
$\#\mathcal{F}^+(\epsilon)$	12 (60%)	12 (60%)	12 (60%)	12 (60%)	12 (60%)
$\#\mathcal{F}^-(\epsilon)$	4 (20%)	3 (15%)	3 (15%)	4 (20%)	4 (20%)
$\#\mathcal{F}^0(\epsilon)$	4 (20%)	5 (25%)	5 (25%)	4 (20%)	4 (20%)
Sum(ϵ)	20 (100%)	20 (100%)	20 (100%)	20 (100%)	20 (100%)
	FERPSO versus DGPSA				
	$\epsilon = 10^{-1}$	$\epsilon = 10^{-2}$	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
$\#\mathcal{F}^+(\epsilon)$	16 (80%)	16 (80%)	16 (80%)	16 (80%)	16 (80%)
$\#\mathcal{F}^-(\epsilon)$	1 (5%)	1 (5%)	1 (5%)	1 (5%)	1 (5%)
$\#\mathcal{F}^0(\epsilon)$	3 (15%)	3 (15%)	3 (15%)	3 (15%)	3 (15%)
Sum(ϵ)	20 (100%)	20 (100%)	20 (100%)	20 (100%)	20 (100%)

$\#A$ denotes the number of elements of a set A, Sum(ϵ) indicates the sum of $\#\mathcal{F}^+(\epsilon)$, $\#\mathcal{F}^-(\epsilon)$, and $\#\mathcal{F}^0(\epsilon)$, and the values in the parentheses indicate the composition ratios of the resulting number to the Sum(ϵ).

https://doi.org/10.1371/journal.pone.0248470.t007

and $\#\mathcal{F}^0(\epsilon)$. The values in parentheses are the composition ratios of the resulting number to the Sum(ϵ).

Focusing on the comparison between RPSO and our DGPSA shown in the top of Table 7, it is understood that $\#\mathcal{F}^+(\epsilon)$ were equal to 12 for all ϵ, and that their composition ratios were 60% $(= \#\mathcal{F}^+(\epsilon)/\mathrm{Sum}(\epsilon) \times 100\ \%)$. From the definition of (23), these results show that our DGPSA was significantly superior to RPSO for 60% of the benchmark functions. On the other hand, $\#\mathcal{F}^-(\epsilon)$ were equal to or less than 4, and their composition ratios were 20% or less. From the definition of (24), these results show that our DGPSA was significantly inferior to RPSO in at most 20% of the benchmark functions.

Next, focusing on the comparison between FERPSO and our DGPSA shown in the bottom of the table, it is shown that $\#\mathcal{F}^+(\epsilon)$ were equal to 16 and that their composition ratios were 80%, as well as that $\#\mathcal{F}^-(\epsilon)$ were equal to 1 and their composition ratios were 5%. This indicates that our DGPSA was significantly superior and inferior to FERPSO for 80% and 5% of the benchmark functions, respectively.

Therefore, these results clearly demonstrate that our DGPSA was significantly superior to the compared one-stage methods in at least 60% of the benchmark functions.

Runtime comparison

Fig 8 shows the boxplot of runtime performed by our DGPSA, RPSO and FERPSO for one-, five-, ten-, and twenty-dimensional functions: $f_1, f_{16}, f_{18}, f_{20}$. The simulation conditions were consistent with those of the performance comparisons in the previous sub section. In Fig 8, **** indicates the statistically significant difference with p-value $<10^{-4}$ between our DGPSA and the compered method, which were tested by Wilcoxon rank-sum test [42]. All algorithms were implemented by Python and executed by the same physical computer (Lenovo ThinkPad T480s, Intel Core i5 processor with 24GB Memory).

Focusing on the runtime comparison between our DGPSA and FERPSO, the runtime of our DGPSA was statistically significantly shorter than that of FERPSO. The mechanism of these results can be explained by the computational complexity described in Fig 1. Focusing on the comparison between our DGPSA and RPSO, our DGPSA was considerably inferior to RPSO for the one-dimensional function f_1. On the other hand, for the high-dimensional functions f_{16}, f_{18}, and f_{20}, our DGPSA showed a statistically significantly shorter runtime than those of RPSO.

These results demonstrated that although our DGPSA has higher computational complexity than RPSO (see Fig 1), its execution time was significantly shorter than RPSO for the high-dimensional functions.

Performance comparison with the existing two-stage methods

The performance of our DGPSA was also compared with those of the existing two-stage methods, namely, RS-CMSA and HillVallEA19, that are the winner of the competition on niching methods held in GECCO 2017 and 2019. Table 8 compares the PR-values of our DGPSA for $\epsilon = 10^{-5}$ (reproduction from Table 5) with those of the two-stage methods in [29, 43], where the benchmark problems and simulation conditions were consistent with those of this study. Table 8 also shows the statistical test results between our DGPSA and the existing two-stage methods, which was conducted similar to the previous section (Statistical test). The original results of RS-CMSA and HillVallEA19 were obtained from the repository provided by the competition organizers https://github.com/mikeagn/CEC2013. The symbol (0) indicates that there was no statistically significant difference between our DGPSA and the compared

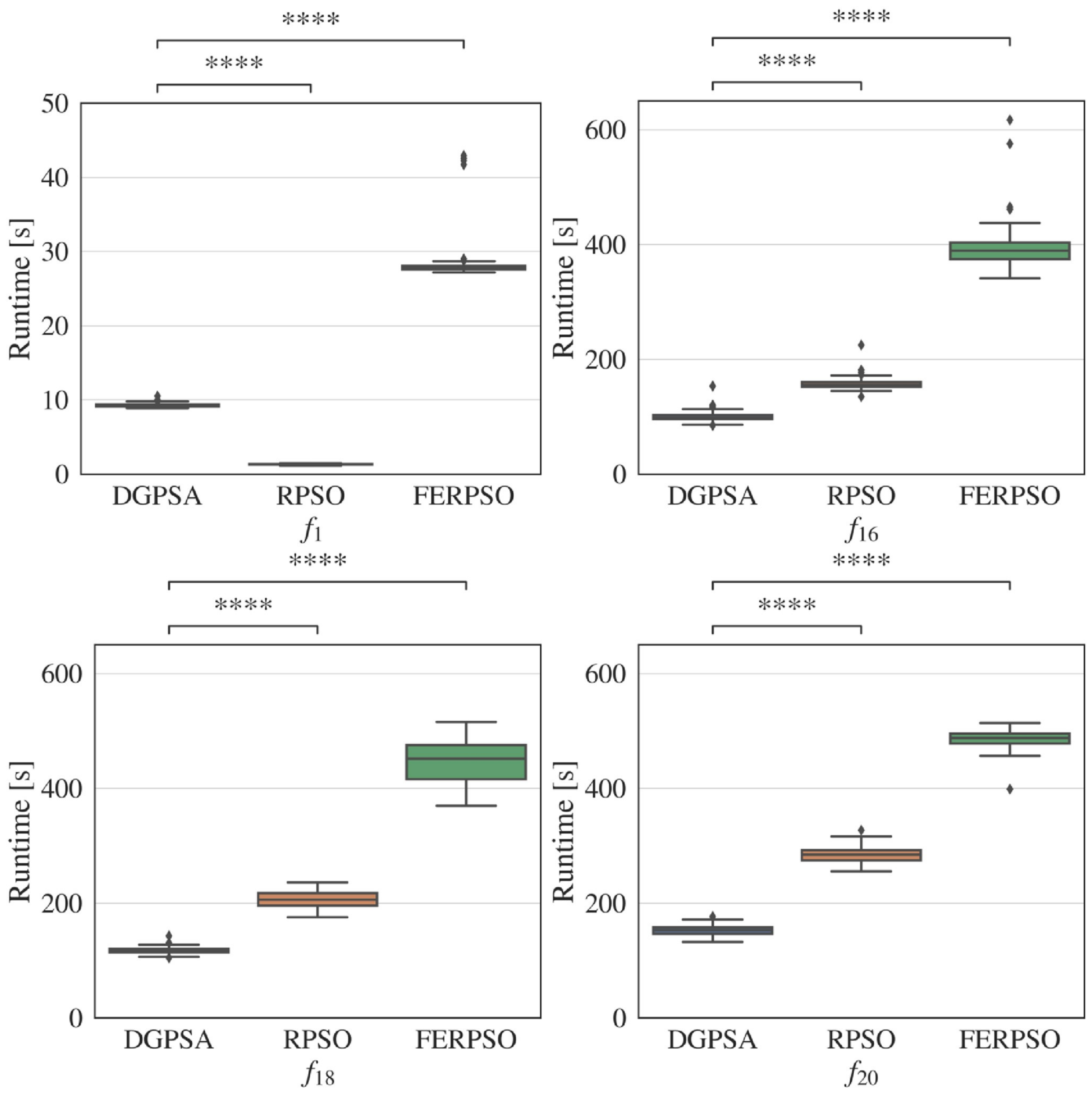

Fig 8. Comparison of runtime of our DGPSA and existing one-stage methods, where **** indicates significantly difference between the runtime of our DGPSA and that of the compered method with *p*-value $<10^{-4}$.

https://doi.org/10.1371/journal.pone.0248470.g008

method. The symbol (–) indicates that our DGPSA was significantly inferior to the compared method with a significance level of 0.05.

Our DGPSA performed comparably to the existing two-stage methods on functions f_1, f_2, f_3, f_4, and f_5 with no statistical difference. Although the *PR*-values for the other functions were significantly inferior in our DGPSA than in the two-stage methods, our one-stage DGPSA is suitable for non-experts in optimization algorithms who wish to implement an MMO algorithm into the production items (as described in GPSA's section).

Table 8. *PR* results of our DGPSA and the existing two-stage methods with $\epsilon = 10^{-5}$.

f	DGPSA	RS-CMSA	HillVallEA19	f	DGPSA	RS-CMSA	HillVallEA19
f_1	1.000	1.000 (0)	1.000 (0)	f_{11}	0.667	0.997 (-)	1.000 (-)
f_2	1.000	1.000 (0)	1.000 (0)	f_{12}	0.694	0.948 (-)	1.000 (-)
f_3	1.000	1.000 (0)	1.000 (0)	f_{13}	0.660	0.997 (-)	1.000 (-)
f_4	1.000	1.000 (0)	1.000 (0)	f_{14}	0.645	0.810 (-)	0.917 (-)
f_5	1.000	1.000 (0)	1.000 (0)	f_{15}	0.316	0.748 (-)	0.750 (-)
f_6	0.948	0.999 (-)	1.000 (-)	f_{16}	0.570	0.667 (-)	0.687 (-)
f_7	0.415	0.997 (-)	1.000 (-)	f_{17}	0.206	0.703 (-)	0.750 (-)
f_8	0.531	0.871 (-)	0.975 (-)	f_{18}	0.167	0.667 (-)	0.667 (-)
f_9	0.186	0.730 (-)	0.972 (-)	f_{19}	0.125	0.503 (-)	0.585 (-)
f_{10}	0.990	1.000 (-)	1.000 (-)	f_{20}	0.125	0.483 (-)	0.482 (-)
				average	0.612	0.856	0.892

The symbol (0) indicates that there was no statistically significant difference between our DGPSA and the compared method and (−) indicates that our DGPSA was significantly inferior to the compared method, with a significance level of 0.05.

https://doi.org/10.1371/journal.pone.0248470.t008

Conclusions

This study proposed a new MMO algorithm called GPSA, which replaces the global feedback term in the classical PSO by an inverse-square gravitational force term between the particles, in order to resolve the drawbacks of existing MMO algorithms, namely, the difficulties in understanding and implementing two-stage methods, and the high computational complexity, large number of tuning parameters and the limited performance in one-stage methods. The proposed GPSA is a simple and purely dynamical algorithm, which is distinct from the existing two- and one-stage MMO methods because of absence of clustering algorithms, restart schemes, taboo archives, and algorithmic procedures for selecting the social and nearest best(s).

First, the types of niching behavior generated by our GPSA were investigated on simple one- and two-dimensional MMO problems. The findings are summarized below:

- Through mutual attraction via the inverse-square gravitational force, the particles dynamically formed into sub-swarms dynamically without algorithmic rules.
- The sub-swarms autonomously gathered near the multiple global optima, with individual particles performing damped oscillations about their respective personal bests.
- A few of the particles in the sub-swarms intermittently escaped, via the repulsive flip, and found distant global optima.

Next, our GPSA was compared with the existing MMO algorithms. The observations are summarized below:

- In the nominal performance comparison, our GPSA was confirmed as a simpler method than the existing methods, because it is less computationally complex and requires fewer tuning parameters than the existing high-performance methods.
- In the actual performance comparison, *PR* and *SR* were measured on the twenty CEC benchmark functions. The exploitation and exploration performances of our GPSA were comparable to those of the existing one-stage methods (RPSO and FERPSO).

Finally, an improved GPSA called DGPSA, which gradually decreases the parameter c_2 as the iterations proceed, was evaluated and the results are summarized below:

- Although our DGPSA is simpler than the existing one-stage methods, it outperformed the compared one-stage methods in both *PR* and *SR*.
- The well-known statistical test method, namely, the Wilcoxon rank-sum test, confirmed the superiority of our DGPSA over the compared one-stage methods on at least 60% of the benchmark functions.
- In comparison of runtime for high-dimensional functions, our DGPSA was significantly superior to the compared one-stage methods.
- Although our DGPSA was statistically inferior to the existing two-stage methods for high-dimensional functions, its simplicity enables its implementation as an MMO algorithm by non-experts.

Clearly, our proposed DGPSA resolves the shortcomings of the existing methods by virtue of its simple and purely dynamical algorithm that outperforms the existing one-stage methods (FERPSO and RPSO). Therefore, we believe that the proposed DGPSA is a more appropriate algorithm than the existing methods for the situation where non-experts in optimization algorithms understand and implement a MMO algorithm to solve the real world problems.

In the future, we plan to investigate the applicability of our GPSA to real-world optimization problems, including optimal structure design [6, 11] and the training of neural networks. We will also consider ways of optimizing the model parameters and applying dynamic inertia weight.

Acknowledgments

We appreciate the valuable feedback offered by Assoc. Prof. Munehisa Sekikawa at Utsunomiya University.

Author Contributions

Conceptualization: Yoshikazu Yamanaka, Katsutoshi Yoshida.

Data curation: Yoshikazu Yamanaka.

Formal analysis: Yoshikazu Yamanaka, Katsutoshi Yoshida.

Funding acquisition: Yoshikazu Yamanaka, Katsutoshi Yoshida.

Investigation: Yoshikazu Yamanaka.

Methodology: Yoshikazu Yamanaka, Katsutoshi Yoshida.

Project administration: Yoshikazu Yamanaka.

Resources: Yoshikazu Yamanaka, Katsutoshi Yoshida.

Software: Yoshikazu Yamanaka.

Supervision: Katsutoshi Yoshida.

Validation: Yoshikazu Yamanaka, Katsutoshi Yoshida.

Visualization: Yoshikazu Yamanaka.

Writing – original draft: Yoshikazu Yamanaka.

Writing – review & editing: Yoshikazu Yamanaka, Katsutoshi Yoshida.

References

1. Li X, Epitropakis MG, Deb K, Engelbrecht A. Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications. IEEE Transactions on Evolutionary Computation. 2017; 21(4):518–538. https://doi.org/10.1109/TEVC.2016.2638437
2. Ward A, Liker JK, Cristiano JJ, Sobek DK. The second Toyota paradox: how delaying decisions can make better cars faster. Sloan Management Review. 1995; 36(3):43–61.
3. Dong-Hyeok Cho, Hyun-Kyo Jung, Cheol-Gyun Lee. Induction motor design for electric vehicle using a niching genetic algorithm. IEEE Transactions on Industry Applications. 2001; 37(4):994–999. https://doi.org/10.1109/28.936389
4. Sun C, Liang H, Li L, Liu D. Clustering with a Weighted Sum Validity Function Using a Niching PSO Algorithm. In: 2007 IEEE International Conference on Networking, Sensing and Control. London, UK: IEEE; 2007. p. 368–373.
5. Kronfeld M, Dräger A, Aschoff M, Zell A. On the benefits of multimodal optimization for metabolic network modeling. In: GCB 2009—German Conference on Bioinformatics 2009. Bonn: Gesellschaft für Informatik e.V.; 2009. p. 191–200.
6. Luh GC, Lin CY. Optimal design of truss-structures using particle swarm optimization. Computers and Structures. 2011; 89(23-24):2221–2232. https://doi.org/10.1016/j.compstruc.2011.08.013
7. Liao J. Service Composition Based on Niching Particle Swarm Optimization in Service Overlay Networks. KSII Transactions on Internet and Information Systems. 2012; 6(4):1106–1127.
8. Zhang J, Gong D, Zhang Y. A niching PSO-based multi-robot cooperation method for localizing odor sources. Neurocomputing. 2014; 123:308–317. https://doi.org/10.1016/j.neucom.2013.07.025
9. Ma D, Ma J, Xu P. An adaptive clustering protocol using niching particle swarm optimization for wireless sensor networks. Asian Journal of Control. 2015; 17(4):1435–1443. https://doi.org/10.1002/asjc.1050
10. Mehmood S, Cagnoni S, Mordonini M, Khan SA. An embedded architecture for real-time object detection in digital images based on niching particle swarm optimization. Journal of Real-Time Image Processing. 2015; 10(1):75–89. https://doi.org/10.1007/s11554-012-0256-7
11. Gholami M, Alashti RA, Fathi A. Optimal design of a honeycomb core composite sandwich panel using evolutionary optimization algorithms. Composite Structures. 2016; 139:254–262. https://doi.org/10.1016/j.compstruct.2015.12.019
12. De Jong KA. Analysis of the behavior of a class of genetic adaptive systems. University of Michigan. Ann Arbor, Michigan, USA; 1975.
13. Mahfoud SW. Crowding and Preselection Revisited. In: Parallel Problem Solving from Nature 2. North Holland, Amsterdam; 1992. p. 27–36.
14. Sareni B, Krahenbuhl L. Fitness sharing and niching methods revisited. IEEE Transactions on Evolutionary Computation. 1998; 2(3):97–106. https://doi.org/10.1109/4235.735432
15. Beasley D, Bull DR, Martin RR. A Sequential Niche Technique for Multimodal Function Optimization. Evolutionary Computation. 1993; 1(2):101–125. https://doi.org/10.1162/evco.1993.1.2.101
16. Harik GR. Finding multimodal solutions using restricted tournament selection. In: Proceedings of the Sixth International Conference on Genetic Algorithms. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1995. p. 24–31.
17. Yin X, Germay N. A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization. In: Artificial Neural Nets and Genetic Algorithms. Vienna: Springer Vienna; 1993. p. 450–457.
18. Li JP, Balazs ME, Parks GT, Clarkson PJ. A Species Conserving Genetic Algorithm for Multimodal Function Optimization. Evolutionary Computation. 2002; 10(3):207–234. https://doi.org/10.1162/106365602760234081
19. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks. Perth, WA, Australia: IEEE; 1995. p. 1942–1948.
20. Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya, Japan: IEEE; 1995. p. 39–43.
21. Shi Y, Eberhart R. A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence. Anchorage, AK, USA: IEEE; 1998. p. 69–73.
22. Li X. A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation—GECCO'07. New York, USA: ACM Press; 2007. p. 78–85.

23. Li X. Adaptively Choosing Neighbourhood Bests Using Species in a Particle Swarm Optimizer for Multimodal Function Optimization. In: Deb K, editor. Genetic and Evolutionary Computation—GECCO 2004. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004. p. 105–116.

24. Qu BY, Suganthan PN, Das S. A distance-based locally informed particle swarm model for multimodal optimization. IEEE Transactions on Evolutionary Computation. 2013; 17(3):387–402. https://doi.org/10.1109/TEVC.2012.2203138

25. Li X. Niching Without Niching Parameters: Particle Swarm Optimization Using a Ring Topology. IEEE Transactions on Evolutionary Computation. 2010; 14(1):150–169. https://doi.org/10.1109/TEVC.2009.2026270

26. Fieldsend JE. Running Up Those Hills: Multi-modal search with the niching migratory multi-swarm optimiser. In: 2014 IEEE Congress on Evolutionary Computation (CEC). Beijing, China: IEEE; 2014. p. 2593–2600.

27. Ahrari A, Deb K, Preuss M. Multimodal Optimization by Covariance Matrix Self-Adaptation Evolution Strategy with Repelling Subpopulations. Evolutionary Computation. 2017; 25(3):439–471. https://doi.org/10.1162/evco_a_00182

28. Maree SC, Alderliesten T, Thierens D, Bosman PAN. Real-valued evolutionary multi-modal optimization driven by hill-valley clustering. In: Proceedings of the Genetic and Evolutionary Computation Conference on—GECCO'18. New York, New York, USA: ACM Press; 2018. p. 857–864.

29. Maree SC, Alderliesten T, Bosman PAN. Benchmarking the Hill-Valley Evolutionary Algorithm for the GECCO 2019 Competition on Niching. arXiv preprint; 2019. Available from: https://arxiv.org/abs/1907.10988.

30. Ishihara Y. Application of Multi-objective Optimization Method to HDD Production Lines. Toshiba review. 2016; 71(1):38–41.

31. Blackwell T, Branke J. Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation. 2006; 10(4):459–472. https://doi.org/10.1109/TEVC.2005.857074

32. Yazdani S, Nezamabadi-Pour H, Kamyab S. A gravitational search algorithm for multimodal optimization. Swarm and Evolutionary Computation. 2014; 14:1–14. https://doi.org/10.1016/j.swevo.2013.08.001

33. Li X, Engelbrecht A, Epitropakis MG. Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization. Evolutionary Computation and Machine Learning Group, RMIT University, Technical report; 2013.

34. Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: A Gravitational Search Algorithm. Information Sciences. 2009; 179(13):2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

35. Tsai HC, Tyan YY, Wu YW, Lin YH. Gravitational particle swarm. Applied Mathematics and Computation. 2013; 219(17):9106–9117. https://doi.org/10.1016/j.amc.2013.03.098

36. Blackwell TM, Bentley PJ. Dynamic Search with Charged Swarms. In: Proceedings of the Genetic and Evolutionary Computation Conference. June. New York City, New York; 2002. p. 19–26.

37. Poli R, Kennedy J, Blackwell T. Particle swarm optimization. Swarm Intelligence. 2007; 1(1):33–57. https://doi.org/10.1007/s11721-007-0002-0

38. Eberhart RC, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. vol. 1. La Jolla, CA, USA; 2000. p. 84–88.

39. Gong YJ, Zhang J, Zhou Y. Learning Multimodal Parameters: A Bare-Bones Niching Differential Evolution Approach. IEEE Transactions on Neural Networks and Learning Systems. 2018; 29(7):2944–2959.

40. Chatterjee A, Siarry P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Computers & Operations Research. 2006; 33(3):859–871. https://doi.org/10.1016/j.cor.2004.08.012

41. Shi Y, Eberhart RC. Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation. vol. 3. Washington, DC, USA: IEEE; 1999. p. 1945–1950.

42. Wilcoxon F. Individual Comparisons by Ranking Methods. Biometrics Bulletin. 1945; 1(6):80–83. https://doi.org/10.2307/3001968

43. Ahrari A, Deb K, Preuss M. Benchmarking Covariance Matrix Self Adaption Evolution Strategy with Repelling Subpopulations for GECCO 2017 Competition on Multimodal Optimization. COIN Report 2017014; 2017.