Soft Computing (2020) 24:11297–11314
https://doi.org/10.1007/s00500-020-05069-2

FOUNDATIONS

An n-state switching PSO algorithm for scalable optimization

Izaz Ur Rahman[1] · Muhammad Zakarya[1] · Mushtaq Raza[2] · Rahim Khan[1]

Published online: 15 June 2020

Abstract
Particle swarm optimization (PSO) is an optimization method that is most widely used to solve a number of problems in various fields such as engineering, economics and computer systems. However, due to its scalability and unsatisfying performance particularly for large-scale optimization problems; numerous PSO variants have been suggested so far, in the literature. This paper also proposes a new variant of the canonical PSO algorithm ('*N*-state switching PSO—NS-SPSO') that uses the evolutionary factor information to update particles velocities and, therefore, further enhance its performance. The evolutionary factor is derived by using the population distribution and the mean distance of each particle from the global best. The population distribution and the mean distance are determined through Euclidean distance. Moreover, algorithmic parameters such as inertia weight, and acceleration coefficients are assigned appropriate values at *N* stages (derived from exploration, exploitation, convergence and jumping out states) that improves the search efficiency and convergence speed. The proposed algorithm is applied to 12 widely used mathematical benchmark functions that demonstrate its best performance in terms of minimum evaluation error, fast convergence and low computational time. Besides these, seven high-dimensional functions and few other algorithms for large-scale optimization were considered to test the scalability of NS-SPSO algorithm. Our comparative results show that NS-SPSO performs well on low-dimensional problems and is promising for solving large-scale optimization problems. Furthermore, the proposed NS-PSO algorithm almost outperforms its closest rivals for various benchmarks.

Keywords Particle swarm optimization · Evolutionary factor · Large-scale optimization · scalability

1 Introduction

Optimization is a real-world problem that is illustrated as the minimization or maximization of an objective function according to some constraints. In order to optimize real-world problems, they must be mathematically formulated, first. Mathematically, all optimization problems can be divided into two types based on the derived cost function, i.e. linear and nonlinear. In the literature, three different approaches have been adopted to solve optimization problems, i.e. deterministic, analytical and stochastic (Elijah 2012). Moreover, all deterministic approaches are based on a given initial condition and/or assumption, whereas analytical approaches have some pre-defined targets for a particular problem. Both of these methods diverge if the dimensionality factor of an optimization problem increases, dramatically. Therefore, to resolve the issue of quick divergence, the third approach, i.e. stochastic is largely used to solve and find near optimum solutions to various real-world optimization problems.

Among stochastic approaches, a population-based technique known as PSO (particle swarm optimization) is widely used to solve the optimization problems (Kenndy and Eberhart 1995; Eberhart and Kennedy 1995). PSO is a powerful and modern swarm intelligence technique to solve the global optimization problems. The particles find the objective of real-world optimization problem by emulating the

Communicated by A. Di Nola.

✉ Izaz Ur Rahman
izaz@awkum.edu.pk

✉ Muhammad Zakarya
mohd.zakarya@awkum.edu.pk

Mushtaq Raza
mushtaq.raza@fe.up.pt

Rahim Khan
rahimkhan@awkum.edu.pk

[1] Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan

[2] Faculty of Engineering, University of Porto, Porto, Portugal

behaviours of fish schools and bird flocks. Population is referred as swarm and each individual agent is a candidate solution that is referred as particle. Generally, two approaches are used to initialize the swarm, i.e. random initialization of the population and swarm initialization according to the problem variables. Each particle i in a swarm has some attributes and it is associated with two vectors, i.e. velocity vector $v_i = [v_i^1, v_i^2, \ldots, v_i^D]$ and position vector $x_i = [x_i^1, x_i^2, \ldots, x_i^D]$, where D represents the problem dimensionality factor. Velocity and position are initialized randomly within the search space range. Throughout the development process, the velocity and position of ith particle are updated on dimension D according to the following two equations:

$$v_i(t+1) = v_i(t) + c_1 r_1(\text{pbest}_i(t) - x_i(t)) + c_2 r_2(\text{gbest}_i(t) - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where c_1 and c_2 represent acceleration coefficients (or weights) and have constant values equal to 2.0, usually. Furthermore, r_1 and r_2 are two random numbers, which are uniformly distributed, and belong to (0, 1).

The velocity update equation (i.e. Eq. 1) has three terms on the right-hand side in which the first term is called momentum or inertia weight. The second term is called cognitive part that represents the personal influence of the particle. The third term is called the social part that denotes the social and collective influence of the particle. The main role of the acceleration coefficients is to adjust the balance between exploitation and exploration of each particle in the search space during its movement. Moreover, $V_{\max}$ is also introduced to further constrain the movement of each particle within the boundary of the search space. The value of $V_{\max}$ is given according to the problem' minimum and maximum boundaries. Moreover, small values for $V_{\max}$ cause exploitation (local minima), and large values for $V_{\max}$ cause exploration (local maxima). Each particle memorizes its best position that has found in the history. The best position that is found by each particle itself is called personal best P_{best} position. However, the best position found by the whole swarm is called global best or G_{best}. In the beginning, each particle moves with random velocity in the search space. Subsequently, each particle dynamically fine-tunes its movement velocity matching to the experiences of its own and other participants. The particle position will be updated constantly until the final optimal solution is found or maximum number of iterations are reached. Figure 1 describes the main process of traditional PSO algorithm.

Like traditional evolutionary algorithms, sensitivity of parameter and premature convergence are the tightly coupled issues associated with the PSO algorithm. The proposed model, that is a novel variant of the traditional PSO scheme, is introduced to resolve the aforementioned issues and more likely enhancing its search capacity. The proposed module is focused on how to: (i) control parameters modification (i.e. weight of inertia and acceleration coefficients); (ii) design topological structures of the model that is used for updating the velocity; and (iii) form a hybrid of PSO and evolutionary algorithms (Van den Bergh and Petrus Engelbrecht 2006; Eberhart and Shi 2001, 2004; Ozcan and Mohan 1999; Cheng and Jin 2015; Robinson et al. 2002; Juang 2004). Besides these, evolutionary state estimation (ESE)-based approaches that divides the search space into several sub-stages are also suggested to improve PSO search efficiency and convergence speed (Zhan et al. 2009). However, the four states do not guarantee the suitability of such approaches to large-scale, scalable, optimization problems, in particular, high dimensional.

In this paper, we introduce a novel N state switching PSO (NS-SPSO) algorithm. The NS-SPSO algorithm is based on the assumptions to further improve the performance of NS-MJPSO (Rahman 2016) by excluding some complex steps such as: (i) determine the jumping probability according to the domain knowledge; however, in practice, it is very difficult to have enough domain knowledge of the optimization problem; and (ii) reduce the computational complexity and burden induced by the Markov chain process. In the proposed NS-SPSO algorithm, particles velocities are updated purely according to the evolutionary factor value. The particle switches from one state to another state according to the assessment of its current evolutionary factor. Furthermore, the choice of each particle either: to stay in the current state; or to switch to other state, is made by how large the value of evolutionary factor is. First of all, the population distribution and the mean distance are determined by using Euclidean distance. The evolutionary factor is then derived by using the population distribution and the mean distance of each particle from the global best. In the population distribution, we determine how far the particles are away from each other and also their global optimums. The main states are described as exploration, exploitation, convergence and jumping-out states (Zhan et al. 2009). However, in the N states, we further divide the main four states into sub-states or stages. Each state is assigned a value in the range of (0, 1).

The algorithmic parameters such as inertia weights and acceleration coefficients are assigned appropriate weights according to each sub-state or stage. In the N states, N number of acceleration coefficients are assigned, but its appropriate value is selected during the evaluation process according to the current state. Subsequently, the algorithm converges to the optimum in just few iterations. For that reason, we have adopted the concept of linearly time decreasing inertia weight for the proposed NS-SPSO algorithm. Extensive simulations have been carried out to evaluate the performance of our proposed NS-SPSO algorithm through

applying it to 12 most widely used benchmark functions. The benchmark functions consist of six uni-modal and six multi-modal problems. The results produced by the NS-SPSO are then compared with NS-MJPSO and other state-of-the-art algorithms (PSO variants). The average/best evaluation values are shown in the tables and further illustrated in the graphical figures. The proposed algorithm has consumed the shortest computation time and also has produced second best results (in terms of accuracy) in comparison with all other variants—with the notable exception of NS-MJPSO. Furthermore, the proposed algorithm has solved the problem of premature convergence to some extent by the concept of state switching. The particle successfully learns from the population distribution about the neighbourhood and also the global best position. Then, the particles efficiently move towards the global optimum in shorter time. Moreover, the classification of N states has induced the balance between local and global search regions. Following are the main contributions of our work:

1. an N-state switching PSO is proposed that extends the four-states approach (Zhan et al. 2009) into N-state for large-scale optimization problem;
2. an additional parameter based on evolutionary switching is introduced;
3. an existing mechanism for calculating the inertia weight parameter (Shi and Eberhart 1998b) is being adopted; and
4. the proposed approach is evaluated on various lower and high-dimensional (uni-modal and multi-modal) benchmark functions.

The rest of work in this paper is organized as follows. Section 2 is dedicated to summarize the background literature, the structure of basic PSO algorithm and its developments towards the proposed work. A brief description of the problem is presented in Sect. 3. In Sect. 4, the structure of the proposed algorithm is presented. In the following Sect. 5, the performance of proposed NS-SPSO algorithm is thoroughly examined in comparison to the other well-known PSO variants. In Sect. 6, we have briefly summarized our proposed work along with several directions for further research.

2 Related work

PSO is a meta-heuristic, population-based algorithm which was first introduced by Kennedy and Eberhart in 1995 (Kennedy and Eberhart 1995; Eberhart and Kennedy 1995). The main concept is inspired by the swarm intelligent behaviour and choreography of birds flocking and fish schooling (Kennedy et al. 2001). PSO imitates the participant agents called particles to get to the optimum location in the search space. After the random initialization of population, the particles compare their current position to their neighbours and thus move to the new position. Basically, each particle is a candidate solution and it keeps track of the best places found by itself within the trial history. It is denoted as personal best or P_{best} symbolically, whereas the best value ever found by entire swarm is called global best or G_{best}. All particles are collectively named as swarm. PSO algorithm is based on two simple equations denoted as velocity update v_i, and position update x_i. A constant value 2 is used for the acceleration coefficients c_1 and c_2. Apart from that, two uniformly distributed random numbers denoted as rand_1 and rand_2 have also been used. The simplified structure, good quality solutions, quick convergence and algorithm reliability are the main characteristics that have attracted researchers in various fields. PSO has been applied to various real-world optimization problems (Eberhart and Shi 2001; Krohling and dos Santos Coelho 2006; Ho et al. 2008; Liu et al. 2007; Eberhart and Shi 2004; Wang et al. 2013; Hu et al. 2015) in the last two decades. Due to the limitations of getting trapped into local optimum and excessive evaluations the basic PSO has been further modified. Several variants have been developed with extra capabilities.

In PSO modified versions, the diversity of the swarm has been improved by introducing the various structures for topologies in Suganthan (1999) and Kennedy (1999). Kennedy and Mendes (2002) have proposed two different types of topologies named as ring and Von-Neumann topologies. A novel fully informed PSO (FIPS) has been proposed in Mendes et al. (2004). In FIPS, the particles learn from their peers with the best fitness in their neighbourhood. Another variant, the comprehensive learning PSO (CLPSO) has been developed in Liang et al. (2006). This algorithm has also contributed to the area of topological improvements of the PSO. The performance of the above-mentioned algorithms are investigated on various uni-modal and multi-modal benchmark functions. Another variant of PSO is developed in Qu et al. (2013) that derives the mean distance of particles in the local neighbourhood. This algorithm has been used for the problems having many local optima.

PSO algorithm is used in combination with the other techniques such as evolutionary techniques (Zhan et al. 2009), genetic algorithm (Robinson et al. 2002; Valdez et al. 2014) and ant bee colony (Shelokar et al. 2007). Additional parameters have been introduced into PSO algorithm. The concept of niching has been incorporated with PSO algorithm in Brits et al. (2007). Gaussian mutation has been introduced by Higashi and Iba (2003). Another adaptive particle swarm optimization (APSO) algorithm has been proposed (Zhan et al. 2009), evolutionary state information has been used as an additional term added to the process of basic PSO. Four evolutionary states S_1, S_2, S_3 and S_4 (exploration, exploitation, convergence and jumping out) have been introduced.

Each state has assigned an appropriate value from the fuzzy membership interval of the particle current state. The evolutionary factor E_f has been used to initialize the population distribution, and to measure the mean distance between the global best and other particles in the swarm. Four states have been described by taking population distribution information E_f in to account, which describes convergence, exploration, exploitation and jumping-out states, respectively. Fuzzy classification method is used for classifying the states, which results in some limitations of excessive computation of acceleration coefficients in each generation, swarm stagnation in the local optima, if the current global best is the local optimum and the last one is the complicated implementation of classification method. The authors state that dividing the search space into sub-stages increases search efficiency and convergence speed. However, the four states still do not guarantee to solve the local optimum and pre-mature convergence for high-dimensional optimization problems.

Furthermore, in the PSO performance studies, the computation time has been considered as the initial objective for improvement. Another aim that has been considered is solving the problem of local optima or premature convergence (Ho et al. 2008; Liu et al. 2007; Ciuprina et al. 2002; Liang et al. 2006). The given improved variants have been thoroughly investigated by applying them into numerous real-world problems. However, due to the nonlinear, multi-modal, high-dimensional and complex types of the real-world problems there is still a desirable room for further enhancement to the PSO algorithm. In response to that, the supplementary techniques have been merged to significantly control the parameters of PSO algorithm (Zhan et al. 2007, 2009; Tang et al. 2011). The topological structures have been improved to explore the search space, ensure global optimum and avoid premature convergence (Liang et al. 2006).

A novel hybrid type, switching PSO (SPSO), has been proposed in Tang et al. (2011). Evolutionary state information is used to find the mean distance of all the particles. Then Markov chain is applied to randomly switch particle within the four states according to certain transition probability. Furthermore, appropriate values of acceleration coefficients have been predefined for all states. The main consideration of the switching PSO is to establish the balance between local, global search region and converge quickly to the global optimum in few iterations. The switching mechanism has ensured that the particle will change its state according to certain probability and will not get trapped into local optimum prematurely. SPSO has shown the best performance for benchmark functions and genetic regulatory networks (GRN) application (Tang et al. 2011). Therefore, to make the existing SPSO robust and more accurate, we have proposed several modifications to the current SPSO algorithm.

In Weibo et al. (2018), a randomly occurring distributedly delayed PSO algorithm (RODDPSO) is suggested. In RODDPSO, the evolutionary state is computed through using the evolutionary factor. Based on the evolutionary state, the particle switches from one state to another. To reduce the chances of stagnation locally and to explore the search space, the time delays occurring randomly, which shows the previous P_{best} and G_{best} particles, are incorporated in the velocity update equation. Empirical evaluation suggests that RODDPSO outperforms some well-known PSO variants over eight benchmark functions. Previous studies show that PSO suffers from premature convergence, particularly, in problem relating to data clustering. RODDPSO has been evaluated for data clustering. Similarly, density-based PSO variants are presented for data clustering in Alswaitti et al. (2018) and Ling et al. (2016); and evaluated over various benchmark functions.

The proposed N state switching PSO algorithm (NS-SPSO) is the modified version of our previously developed algorithm NS-MJSPO described in Rahman (2016) and Rahman et al. (2020), where N is number of possible states that can be any positive value. In Rahman et al. (2020), the transition matrix based on the probability of each particle is used to predict the next state of the particle using Markovian jumping mechanism. Basically, the main idea is similar to four-state versions given in APSO and SPSO (Zhan et al. 2009; Tang et al. 2011). The NS-SPSO algorithm is based on evolutionary techniques. The N states are visualized as sub-states or stages of four states. Furthermore, the evolutionary states are described by calculating and then using the population distribution, mean distance using Euclidean space, maximum, minimum values and also the index of global best particle in the population distribution information. The inertia weight ω is an important parameter of the PSO algorithm, which has first been introduced by Shi and Eberhart (1998a). Shi and Eberhart (1998b) proposed the concept of linearly decreased inertia weight (LDIW) to compute the value of ω. It iteratively decreases the value of ω from 0.9 to 0.4 on basis of Eq. 9. The idea behind this decrease is possibly the state of the particle—in initial stages, the particle is far away the target; therefore, larger value for ω is proffered.

Besides these, PSO has been applied to a variety of optimization problems including machine learning techniques to improve feature selection of text and document clustering. These falls under the category of PSO application. For example, Abualigah and Khader (2017) and Abualigah et al. (2018) used PSO for feature selection, i.e. an unsupervised learning approach to choose a subset of most informative text features in order to improve the performance of the text clustering and minimize its computational time. The algorithm is known as FSPSOTC (Abualigah et al. 2018). The authors proposed 'H-FSPSOTC' a hybrid PSO algorithm using genetic operator to improve its efficiency (Abualigah and Khader 2017). Their results show that the proposed feature selection method

improved the text clustering outcomes through assisting the k-mean text clustering to make more similar groups.

In this work, we compute the inertia weight ω by combing the evolutionary factor and the time varying strategy (Shi and Eberhart 1999). Acceleration coefficients c_1 and c_2 both takes N number of tuned values. The proposed algorithm is then applied to 12 commonly used uni-modal and multi-model functions of various dimensions. The results are compared with some well-known and most cited algorithms. The proposed algorithm has performed well in terms of the shortest computation time and average/best evaluation values in comparison with all variants in comparison except NS-MJPSO in accuracy for most of the benchmark functions. However, in few problems some additional parameter tuning is required to improve the quality of solution in terms of better evaluation values.

2.1 The basic framework of PSO algorithm

The PSO algorithm refers to the intelligent searching behaviour of all participants named as particles. The population of all particles is called swarm of size n, where each individual particle i is a candidate solution in the problem space. Each particle i holds two vectors quantities, the first one is the velocity of ith particle in Dth dimension and t time is represented as $v_i(t) = [(v_{i1}(t), v_{i2}(t), \ldots, v_{iD}(t))]$ and the second one is the position of the ith particle in Dth dimension and in time t is denoted as $x_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t))$, where D represents dimension of the solution search space. The swarm velocities and positions are initialized randomly with their respective boundaries $x_{\text{in}}(t) \in [x_{\min,n}, x_{\max,n}]$ $(1 \leq n \leq D)$ with $x_{\min,n}$ and $x_{\max,n}$ of the search space; where $V_{\max}$ is maximum velocity set to the 20% of the search space (Eberhart and Shi 2001). During the process of algorithm evaluation iteratively, the particle i with dth dimension is updated as follows.

$$v_i(t+1) = \omega v_i(t) + c_1 \text{rand}_1(\text{pbest}_i(t) - x_i(t)) + c_2 \text{rand}_2(\text{gbest}_i(t) - x_i(t)) \quad (3)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4)$$

where ω is called the inertia weight (Shi and Eberhart 1998a), c_1 and c_2 are denoted as acceleration coefficients (Eberhart and Kennedy 1995). Further, rand_1 and rand_2 are two uniformly distributed random numbers generated between [0, 1] (Kenndy and Eberhart 1995). Similarly, Pbest represents $\text{pbest}_i = (\text{pb}_{i1}, \text{pb}_{i2}, \ldots, \text{pb}_{iD})$. The personal best is the particle having the best fitness value found by the ith particle so far; and gbest means gbest denoted as $\text{gbest}_D = (\text{gb}_1, \text{gb}_2, \ldots, \text{gb}_D)$. The global best is the particle with the best fitness value found by the entire swarm. Notre that, n_{Best} is used for global best in the neighbourhood version, G_{Best} for the global version, and L_{Best} for the local version of the PSO. The particle's personal experience and its social interaction determines the direction towards its best position, iteratively. The movement of each and every particle in the search space and the influence of its parameter is shown in Fig. 1 (Weber and Van Noije 2012).

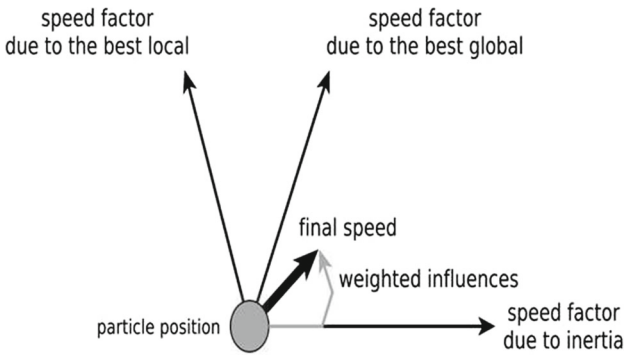

Fig. 1 The particle and parameter social learning behaviour (Rahman 2016) (this generalized figure denotes the movement of each particle that could be in any state out of the N states—as shown in Fig. 2)

3 Problem description

In traditional PSO, various issues, such as parameter sensitivity, getting stuck in local optima and weak robustness, affect its performance, particularly, for large-scale optimization problems. Therefore, rich literature suggests various PSO variants, in different problem domains, in order to: (i) enhance the search performance; and (ii) address one or more aforementioned issues. Among these, our previously proposed NS-MJPSO algorithm (Rahman 2016) has shown to be successful based on the assumptions that: (i) we know how to determine the jumping probability according to the prior knowledge; and (ii) we do not really care about the computational burden induced by the extra stage of the Markovian state jumping (Rahman 2016). In practice, however, it is quite often that we have less domain knowledge about the optimization problem and the computational burden is a concern. Therefore, it is essential to tackle this issue. Moreover, evolutionary state estimation and divisibility of the search space into several sub-stages does not guarantee fast convergence and mature convergence, in particular, for large-scale, high-dimensional, optimization problems. In this case, we have proposed another novel PSO algorithm, which is the NS-SPSO algorithm. For NS-SPSO algorithm, we update the velocity purely based on the evolutionary factors where the state switches from one to another according to the evaluation of its evolutionary factor. In other words, the possibility

for the state switching or staying is determined by how large the evolutionary factor is. Our proposed NS-SPSO algorithm is then examined through applications to some benchmark functions.

4 The novel N state switching PSO

This section elaborates the development of novel NS-SPSO for the enhancement of global search performance. A new switching parameter $\delta(t)$ is introduced in the basic PSO velocity update Eq. (5). The value of N states alongwith other parameters is initialized; where N represents the evolutionary state number and is described as $N \in \{4, 5, 6, \ldots, n-1, n\}$. The basic idea of state division is shown in Fig. 2. It is to be noted that the original four (4) state model—convergence, exploration, jump out and exploitation (Tang et al. 2011), is divided into different sub-states. Moreover, the likelihood of dividing a single state into multiple states is also possible such as (i) in exploitation state where particles are stuck or (ii) due to the utilization of small c_1 and large c_2 values, particles enter into the pre-mature convergence state where division of every state into at least two sub-state is mandatory which lead to $N = 5$ or 6. Note that, stucking a particle in a particular stage means that the algorithm pre-maturely converges without further optimizing the objective. In order words, the suboptimal value is computed in first few iterations and is repeatedly computed the same until the end. Furthermore, the convergence state means that the particle has already achieved its target position. The accuracy and search ability of a PSO based variant is significantly improved if a particular state is divided into multiple states. Furthermore, the proposed sub-state mechanism reduces the probability of skipping a particle during the transition process. Although sub-state mechanism resolves the aforementioned problem, a high computational overhead, i.e. in terms of search space and dimensionality of the problem, is major issue associated with the small state model. Note that, the division of a single state into multiple stages may not be beneficial in all cases, as investigated in Sect. 5.3. Therefore, it is essential to evaluate and estimate an appropriate number of states for a particular problem with respect to its dimensionality. The novel N state switching PSO (NS-SPSO) is investigated by applying to 12 uni-modal and multi-modal widely used benchmark functions (Liang et al. 2006; Suganthan et al. 2005) which are given in Sect. 5.

$$v_i(t+1) = \omega v_i(t) + c_1(\delta(t))r_1(t)(\text{pbest}_i(t) - x_i(t)) + c_2(\delta(t))r_2(t)(\text{gbest}_i(t) - x_i(t)), \quad (5)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (6)$$

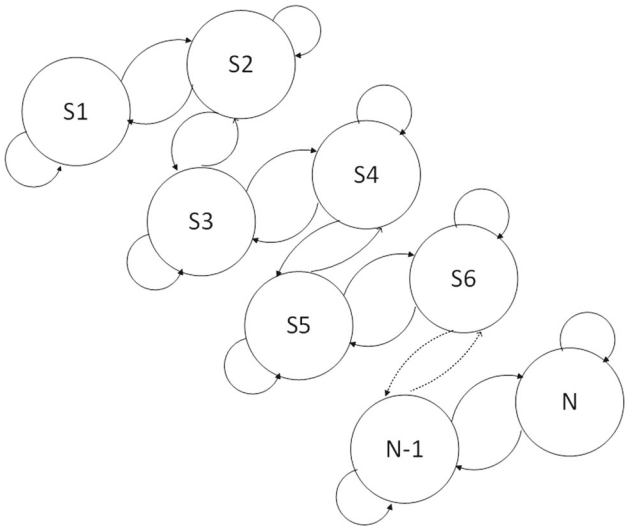

Fig. 2 The concept of N states using the Markov chain process (Rahman et al. 2020)

4.1 Prediction of evolutionary states

In the beginning of population distribution, the particles are dispersed in the search space. However, in the evolutionary process the particles group together iteratively in the later stages and find their local and global optimal positions in the search space. The extraction of information from the population distribution and using that for further describing the evolutionary state is an important research topic in PSO. Hence, the population distribution information in each generation is important to be recorded. A clustering based technique was introduced for evolutionary state estimation in Zhang et al. (2007) and Zhan et al. (2007), whereas fuzzy classification method is used for calculating four evolutionary states in Zhan et al. (2009).

In the first step of population distribution, the mean distance from the global best particle in the search space for each i is derived. The particles having smaller distance from the global best are close to the convergence state and it switches to the other state according to evolutionary factor. Furthermore, the particles located far away from the global best switch to another state with higher values of its parameters. The mean distance is calculated by using Euclidean matrix as follows (Zhan et al. 2009; Tang et al. 2011):

$$P_d(i) = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i(k) - \bar{x}_j(k))^2} \quad (7)$$

In Eq. (7), N represents the swarm size and D stands for dimensions of the problem. Evolutionary factor E_f has been introduced by Zhan et al. (2009), and it has further been used by Tang et al. (2011). Note that, the value of N-state

is pre-determined. Then, the computed value for E_f, based on the Euclidean distance, and comparing the E_f value with States(δ) (as shown in Eq. 8) describes the current state in the particular N-states. For example, if $N = 8$ and $0 \leq E_f < 2/N$ then the particle is in the second stage of the exploration state.

This paper presents a novel switching mechanism by extending from four states up to N states (Rahman 2016). It further divides the four states to possible sub-states. The sub-states smoothly describe the unit of association for particle in a particular state. By dividing into sub-states, we assume significant improvement in adopts more suitable values for its parameters. The sub-states represent the certain stages according to the value of N states. Hence, by increasing the number of states the performance of the algorithm will be improved in terms of accuracy in the evaluation results, but the computation burden will increase slightly. An auxiliary parameter δ, as described in Sect. 4.3, is used in the new velocity update Eq. (5) in Sect. 4. Initially, $c_1(\delta(0))$ and $c_2(\delta(0))$ with δ at position/state 0, are assigned 2. Then, the appropriate value for the c_1 and c_2 are automatically assigned during the runtime.

Here, we have derived the mean distance of all $P_d(i)$ by using Eq. (7) and find P_{dg} the global best particle (which corresponds to the usual global best of the traditional PSO gbest), $P_{d(max)}$ as the maximum mean distance and $P_{d(min)}$ as the minimum mean distances. Consider the values derived by using Eq. (7) in the population distribution and then compute the evolutionary factor using Eq. (8) (Fig. 3).

$$E_f = \frac{P_{dg} - P_{d(min)}}{P_{d(max)} - P_{d(min)}} \in [0, 1]$$

$$\text{States}(\delta) = \begin{cases} 1, & 0 \leq E_f < \frac{1}{N}, \\ 2, & \frac{1}{N} \leq E_f < \frac{2}{N}, \\ 3, & \frac{2}{N} \leq E_f < \frac{3}{N}, \\ \vdots & \vdots \\ N, & \frac{N-1}{N} \leq E_f < 1 \end{cases} \tag{8}$$

The division of solution space exploration approaches, which are found in other population-based optimization algorithms, such as APSO (Zhan et al. 2009; Tang et al. 2011), into sub-stages improve their convergence and, as well as, search efficiency. This is due to the fact that the entire population is intelligently managed and controlled according to particle's initial position. For example, if the particle is far away from its gbest, then maximum acceleration weight is assigned to speed up its movement towards the target position. Similarly, if particles are too close to their target position their speed is controlled accordingly.

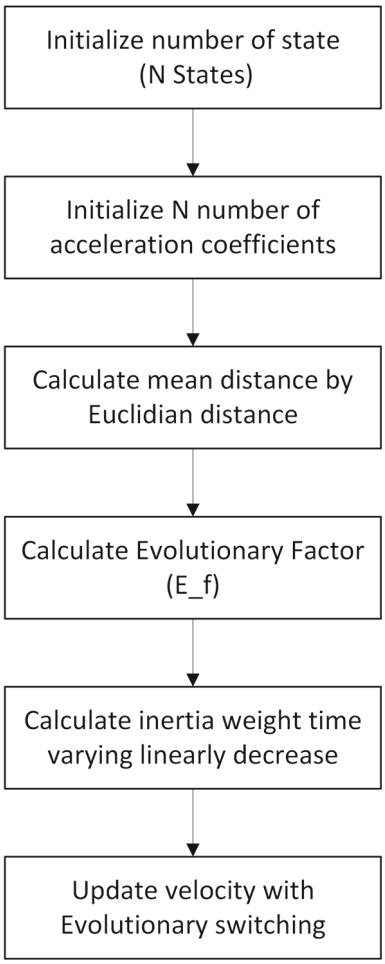

Fig. 3 The switching parameters flow diagram

4.2 Mechanism for inertia weight calculation

The inertia weight ω has the significant contribution in the control of PSO algorithm. It has the main influence on the global and local search performance. If the value of ω is small then it causes exploitation in the local search region. Large ω drag the swarm towards global search region. In this study, ω is computed by using the linearly decreasing strategy (LDIW) proposed in Shi and Eberhart (1999). In LDIW strategy, the value of ω is iteratively decreased from 0.9 to 0.4 based on the idea that in initial stages the particle needs larger values to control its movement towards gbest. The main reason to select this method is the suitability of the decreasing factor with the current state of the particle.

$$\omega = (\omega_{max} - \omega_{min}) \times \frac{\text{iter}}{\text{maxiter}} + \omega_{min} \tag{9}$$

Here, we have initialized $\omega = 0.9$ as maximum and 0.5 values as minimum. The value of inertia weight ω is linearly decremented with time, so as to accelerate the velocity

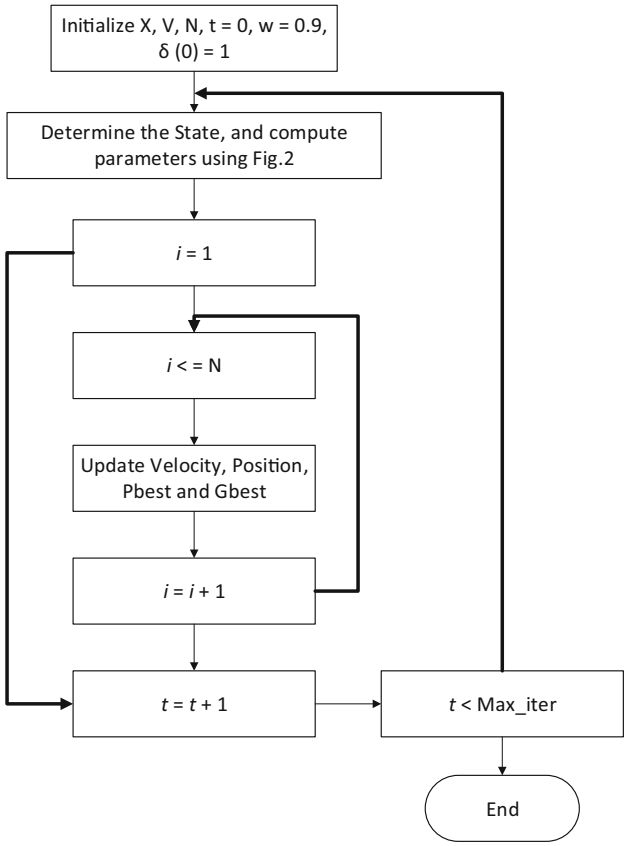

Fig. 4 *N* state switching PSO algorithm flowchart

of each particle according to its current position. Further developments and investigation of various methods regarding inertia weight have been briefly described in Rahman (2016). The complete structure of NS-SPSO is described by the following flowchart in Fig. 4, and the steps are shown in Algorithm 1:

4.3 Selection of acceleration coefficients

In the proposed NS-SPSO algorithm, the acceleration coefficients are selected and adjusted manually according to the problem. N number of acceleration coefficients are required. For instance, if $N = 4$ then we have to initialize four values for each acceleration coefficient $C = [2, N]$. Each value is designated to a particular state. The N acceleration coefficient values are pre-initialized. Initially, $c_1(\delta(0))$ and $c_2(\delta(0))$ are assigned 2. Then the appropriate value for the acceleration coefficient is automatically assigned during the program execution time. The strategy for selecting the acceleration coefficients for each state is described in Tang et al. (2011) as follows:

In the proposed NS-SPSO technique, the large value of E_{f} describes the state as *jumping-out-state*. As the particle has the intention to jump from the local optimum towards the global optimum; a large value of social learning factor $c_2(\delta(4)) = 2.2$ and smaller value of cognitive factor $c_1(\delta(4)) = 1.8$ are assigned. Subsequently, the particle flies towards the global best region very quickly. According to this strategy, the proposed algorithm converges to global optimum in few iterations.

Similarly, a relatively small value of E_f describes the current state in the *exploration-state*, according to that a large value of $c_1(\delta(3)) = 2.2$ and smaller value of $c_2(\delta(3)) = 1.8$ are assigned to let the particle explore search spaces on its personal influence.

Moreover, in the *exploitation-state*, a large value of $c_1(\delta(2)) = 2.1$ and smaller value of $c_2(\delta(2)) = 1.9$ are pre-initialized. Slight changes have been made to preserve the balance in local and global search performance. Subsequently, in the *convergence-state* equal values to both $c_1(\delta(1)) = 2.0$ and $c_2(\delta(1)) = 2.0$ because all particles group together in the convergence-state. The steps in the proposed NS-SPSO algorithm are shown in Algorithm 1 and their explanation follows as given. From step 1 to step 5, all the required parameters are being initialized. In subsequent steps, the mean distance or Euclidean distance for each particle is computed from step 7 to step 16. Then, from step 17 to step 18 the evolutionary factor is computed. Next, based on the E_{f} value and States(δ) the current state is computed in step 19 to step 23. In step 24, the next state of each particle is predicted in step 24. Finally, in step 25, the particle is moved through updating its velocity and acceleration coefficients. Note that, step 7 to step 25 are repeated for each particle until all iterations are completed.

4.4 Computational complexity

The computational complexity of NS-SPSO depends on three various parts: (a) fitness evaluations; (b) mean-based population distribution—E_{f} calculation; and (c) learning of the swarm behaviour. In respect of (a), time cost of the fitness evaluation is dependent on the problem size (dimension), which is beyond the scope of our current work. Therefore, we describe the time complexity of the proposed algorithm with respect to (b) and (c).

In respect of (b), the time required to calculate the evolutionary factor (E_{f}) of each particle is constant and is achieved using the Euclidean distance. For m particles, the time required for computing the population distribution is given by:

$$T_m = \mathcal{O}(m) \tag{10}$$

Moreover, behaviour learning is an essential process for updating the particle behaviour; and we assume it similar to other learning mechanisms in various PSO algorithms (Cheng and Jin 2015). As described in Cheng and Jin (2015),

Algorithm 1: State switching process

```
N ← Number of states ;
D ← Dimension ;
S ← Population ;
X ← Current position ;
C ← Acceleration coefficients ;
for each particle i do
    // calculate the mean distance using Equation (7) ;
    temp2 ← 0 ;
    for k ← 1 : S do
        temp ← 0 ;
        for j ← 1 : D do
            temp ← (X(i, j) − X(k, j))^2 + temp ;
        end for
        temp2 ← temp2 + √temp ;
    end for
    Dis_i ← temp2/S [which denotes the traditional PSO global
    best particle gbest];
    // calculate E_f and current state ;
    E_f ← (Dis_j − min(Dis))/(max(Dis) − min(Dis)) ;
    for i ← 0 :N States do
        if i/N States ≤ E_f < (i + 1)/N States then
            δ ← current state i ;
        end if
    end for
    predict the next state using δ and E_f ;
    update velocity using δ with respect to time ;
end for
```

'the time complexity of behaviour learning process is indispensable in a population-based stochastic search algorithm'. Therefore, for n-dimensional problem which consists of m number of particles, the time complexity of the learning process can be obtained as follows:

$$T_l = \mathcal{O}(m \times n) \tag{11}$$

Therefore, the total complexity T of the NS-SPSO algorithm can be described as:

$$T = \mathcal{O}(m(1+n)) \tag{12}$$

5 The experimental work

The proposed NS-SPSO has been evaluated over 12 commonly used benchmark functions that are given in Table 1 f_1 to f_{12} taken from Cheng and Jin (2015). Few of them are uni-modal (f_1 to f_5), and several are multi-modal (f_6 to f_{12}) as described in Cheng and Jin (2015) Initially, the proposed NS-SPSO is applied to the 12 benchmarks functions f_1 to f_{12} in 30 dimensions using different values for N. Then, the evaluation results are compared with published values of six state-of-the-art algorithms. All the variants were re-implemented and evaluated for 30 independent trials. The published results of all the variants are taken from Cheng and Jin (2015) for comparison. Additionally, the proposed algorithm were also tested for its scalability on high-dimensional functions in Sect. 5.4. These consist of various 50 dimensional shifted or rotated functions from f_{13} to f_{19}; and their dimensionality were set to 100, 500 and 1000. All the experimental work have been conducted on a PC with an Intel Core i5-3320M 2.6 GHz CPU and Microsoft Windows 10 Pro 64-bit system. The benchmark test experiments of the 12 uni-modal and multi-modal problems for the proposed NS-SPSO and other PSO variants in comparison are all coded in MATLAB *version R2015a*. It is also worth to be mentioned that because of the evolutionary control and switching techniques the algorithm converge to its optimum in the early stages of the function evaluations.

Mathematical benchmark functions $f_1 - f_{12}$ are taken as for testing high-dimensional problems. Out of those functions, $f_1 - f_5$ functions represent uni-modal. Function f_6 represents a step function that has a single minimum and is disjointed. Function f_7 represents a quadratic and noisy function, where random [0, 1] denotes a uniformly distributed arbitrary variable between [0, 1], whereas functions $f_8 - f_{12}$ describes multi-modal functions, with intent that the number of local minima grows exponentially in conjunction with the problem dimensionality factor (Yao et al. 1999; Törn and Žilinskas 1989). Such kind of problems emerge to be the most challenging class of problems for evolutionary optimization algorithms. Considering uni-modal functions for classical and fast evolutionary methods convergence rates are more fascinating and attractive as compared to the end optimized results, as other methods are specified for optimization uni-modal functions. However, in considering multi-modal functions, the end optimization results are much more significant and important as they contemplate the strength of algorithm in getting away from local optima and finding a near-global optimum position.

Various metrics were used to evaluate the performance of the proposed algorithm. For example, the optimization error denote the difference between the real optimum value and achieved value. Similarly, the near convergent rate specifies the smallest achievable value. Moreover, the computational time shows the wall clock time which an algorithm takes to converge. Similarly, the impact of number of states (N), for high-dimensional optimization functions, on the performance of the proposed NS-SPSO algorithm has been investigated through repeated experimentations. The discussion is consistent with the scalability of the proposed approach.

5.1 Performance analysis of NS-SPSO in benchmark functions

To analyse the performance of the proposed NS-SPSO algorithm over various benchmark functions, we compare it to

Table 1 The 12 common standard functions for experimentation using 30 dimensions (Rahman 2016)

Name	Function	Search range
Sphere	$f_1(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
Schwefel 2.22	$f_2(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
Schwefel 1.2	$f_3(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^n$
Schwefel 2.21	$f_4(\mathbf{x}) = f(x_1, \ldots, x_n) = \max_{i=1,\ldots,n} \lvert x_i \rvert$	$[-100, 100]^n$
Rosenbrock	$f_5(x, y) = \sum_{i=1}^{n} [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	$[-30, 30]^n$
Step	$f_6(\mathbf{x}) = \sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor^n$	$[-100, 100]^n$
Schwefel	$f_7(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = 418.9829d - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^n$
Rastrigin	$f_8(x, y) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	$[-5.12, 5.12]^n$
Ackley	$f_9(\mathbf{x}) = f(x_1, \ldots, x_n) = -a.\exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(cx_i)) + a + \exp(1)$	$[-32, 32]^n$
Griewank	$f_{10}(\mathbf{x}) = f(x_1, \ldots, x_n) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$	$[-600, 600]^n$
Penalized 1	$f_{11}(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y = 1 + \frac{1}{4}(x_i + 1)$	$[-50, 50]^n$
Penalized 2	$f_{12}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$

$$\text{In } f_{11} \text{ and } f_{12}, u(x_j, a, k, m) = \begin{cases} k(x_j - a)^m, x_j > a \\ 0, -a \le x_j \le a \\ k(-x_j - a)^m, x_j < -a \end{cases}$$

Note that, these functions are also scalable to the search dimensionality represented by n. Furthermore, the global optimum is zero for entire functions. These functions relate to uni-modal (f_1 to f_5) and multi-modal (f_6 to f_{12}) functions

Table 2 Parameter coefficients of the PSO variants for comparison—m, and R denote each swarm population size and regrouping period (Liang and Suganthan 2005); moreover χ denote the constriction coefficient parameter, as described in Mendes et al. (2004)

Algorithm	Inertia weight	Acceleration coefficients
LPSO	[0.9, 0.4]	$c_1 = c_2 = 2.0$
GPSO	[0.9, 0.4]	$c_1 = c_2 = 2.0$
DMS-PSO	0.729	$c_1 = c_2 = 1.49445, m = 3, R = 15$
FIPS	$\chi = 0.729$	$c_1 + c_2 = 4.1$
CLPSO	[0.9, 0.7]	$c_1 = c_2 = 1.49445$
NS-MJPSO	[0.9, 0.5]	$c_1 = [2, N], c_2 = [2, N], \phi = 0.9, N$
NS-SPSO	[0.9, 0.5]	$c_1 = [2, N], c_2 = [2, N], N$

the newly developed NS-MJPSO (Rahman et al. 2020; Rahman 2016); and five other variants of the PSO algorithm. All the variants have been coded and re-implemented for comparison purposes. The published values are used in the tables here produced by Cheng and Jin (2015). We compare NS-SPSO to our newly developed NS-MJPSO algorithm, as described in Rahman (2016); since we aim to further enhance its performance by reducing the computational burden. Our second comparative variant is the local-neighbourhood PSO (local-PSO) (Kennedy and Mendes 2002), third is the global best version (GPSO) (Shi and Eberhart 1999), fourth is the dynamic multi-swarm version of PSO (DMS-PSO) (Liang and Suganthan 2005; Cheng et al. 2013), fifth is the fully informed PSO (FIPS) (Mendes et al. 2004), and the sixth and last one is the comprehensive-learning PSO (CLPSO) (Liang et al. 2006). The required parameters along-with the values are described here in Table 2.

The proposed NS-SPSO has characterized outstanding performance on 9 out of 12 problems (i.e. f_1 to f_6 and f_8 to f_{12}) containing both uni-modal and multi-modal problems. We have shown the performance of the proposed NS-SPSO algorithm individually for each function in Table 3. More-

Table 3 The 12 standard test functions' optimization errors—for each procedure and test function, the primary row characterizes the statistical mean or average value (μ), while the subsequent row designates the statistical standard deviation (σ)

	NS-SPSO		NS-MJPSO		LPSO		FIPS		DMS-PSO		CLPSO		GPSO
f_1	6.86E−142	=	2.16E−150	+	4.89E−12	+	7.23E−70	+	3.81E−15	+	6.32E−19	+	5.56E−33
	1.71E−161		2.79E−161		1.80E−14		4.78E−71		4.97E−20		1.69E−19		3.30E−45
f_2	2.26E−100	–	7.93E−95	+	1.33E−08	+	9.99E−39	+	3.29E−11	+	7.49E−12	+	9.67E+00
	3.46E−95		**2.51E−98**		9.36E−10		2.71E−39		1.42E−14		4.70E−12		1.85E−28
f_3	1.54E−25	+	1.48E−23	+	2.75E+01	+	1.16E+00	+	8.35E+01	+	1.06E+03	+	2.22E+03
	1.87E−30		5.45E−27		8.10E+00		3.58E−01		1.06E+01		6.74E+02		4.44E−05
f_4	2.52E−20	+	3.04E−21	+	2.14E−02	+	2.42E+00	+	2.14E+00	+	4.50E+00	+	3.87E−05
	7.63E−24		1.48E−23		8.23E−03		3.60E−01		8.52E−01		3.32E+00		3.71E−06
f_5	1.36E−05	–	1.92E−04	+	6.27E+01	+	2.59E+01	+	3.86E+01	+	9.55E+00	+	1.31E+02
	1.47E−04		**8.47E−09**		7.32E+00		1.25E−01		2.74E−01		1.73E+00		3.84E−01
f_6	0.00E+00	=	0.00E+00	=	0.00E+00	=	0.00E+00	=	2.67E−01	=	0.00E+00	=	0.00E+00
	0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00		0.00E+00
f_7	3.25E−03	+	2.99E+03	+	1.87E+03	+	2.70E+03	+	− 2.55E−01	–	4.85E−13	+	5.52E+03
	5.62E−04		1.40E+03		9.51E+02		1.34E+03		− 7.66E+00		0.00E+00		2.82E+03
f_8	2.41E−03	–	0.00E+00	+	1.69E+01	–	4.25E+01	+	3.32E−02	+	6.13E−09	+	3.05E+01
	3.00E−04		0.00E+00		4.01E+00		2.70E+01		**1.78E−15**		4.23E−10		1.98E−07
f_9	1.45E−14	–	1.33E−14	+	2.87E−06	–	7.16E−15	+	1.49E−08	+	2.98E−10	–	9.25E−01
	1.33E−14		**6.22E−15**		7.68E−08		**6.22E−15**		6.84E−11		1.24E−10		**6.22E−15**
f_{10}	5.33E−15	–	0.00E+00	+	3.94E−03	–	4.48E−09	–	7.22E−03	+	2.20E−12	–	1.05E−02
	4.55E−16		0.00E+00		2.44E−13		0.00E+00		0.00E+00		7.22E−15		0.00E+00
f_{11}	4.15E−02	=	1.57E−32	+	6.97E−15	=	1.57E−32	+	3.46E−03	+	3.09E−20	=	3.46E−03
	1.57E−32		1.57E−32		2.45E−16		1.57E−32		1.24E−21		9.08E−21		1.57E−32
f_{12}	2.20E−03	=	1.35E−32	+	8.45E−14	=	3.66E−04	+	4.76E−03	+	4.17E−19	=	1.43E−16
	1.35E−32		1.35E−32		3.36E−15		1.35E−32		1.47E−18		1.09E−19		1.35E−32
+/=/–		**3/4/5**		**11/1/0**		**6/3/3**		**10/1/1**		**10/1/1**		**7/3/2**	

The smallest value for μ denotes the best outcome and, therefore, good algorithm; furthermore, the ranking formula was used to rank the NS-SPSO algorithm against other variants

over, Figs. 5and 6 visually show the convergence rates of various algorithms over various benchmark function—the lower the curve, the minimum is the value. We have executed the algorithms for 30 independent trails due to the randomness of the algorithms results. In the empirical results, we store the mean, small evaluation errors and standard deviation for each function in all trails. In Table 3, the statistical results are given over 30 independent trails for the newly developed NS-SPSO and all other comparison algorithms. To simplify our findings, a ranking method is developed to show their significance. Over the 30 runs' population of data, a statistical two-tailed *t test* was carried out with 95% confidence interval, i.e. significance level $\alpha = 0.05$ (Zakarya and Gillam 2019; Khan et al. 2020). As shown in Table 3, if the proposed NS-SPSO technique outperforms other algorithms, a plus sign '+' was inserted in front of the corresponding results. Similarly, if no significance differences were observed between NS-SPSO and other algorithm, an equal sign '=' was used. Furthermore, other algorithms outperform the proposed NS-SPSO algorithm, a minus sign '–' was used. At the bottom Table 3, the total number of '+', '=' and '-' is outlined. Moreover, best results for each function are shown in boldface (Cheng and Jin 2015).

The newly developed NS-SPSO has produced the best value (global optimum) in comparison with all other algorithms in terms of minimum values (lower curves). Further, comparatively NS-SPSO is faster in execution than the closest rivals. Therefore, the newly developed NS-SPSO has the promising performance for uni-modal problems. As shown in Fig. 5, the NS-SPSO is on top rank for functions like f_1, f_2, f_3, f_4, f_5 and f_6. Furthermore, the NS-MJPSO is on rank 2 for these functions—NS-SPSO outperforms the NS-MJPSO algorithm. Moreover, the newly developed NS-SPSO is the fastest one in execution time in comparison with all other algorithms. However, the newly developed NS-SPSO and NS-MJPSO algorithms have promising performance for such kind of optimization problems.

For multi-modal function f_7, as shown in Fig. 6, the CLPSO and DMS-PSO have the minimum values; and our algorithm diverge quickly. The main reason is that the pro-

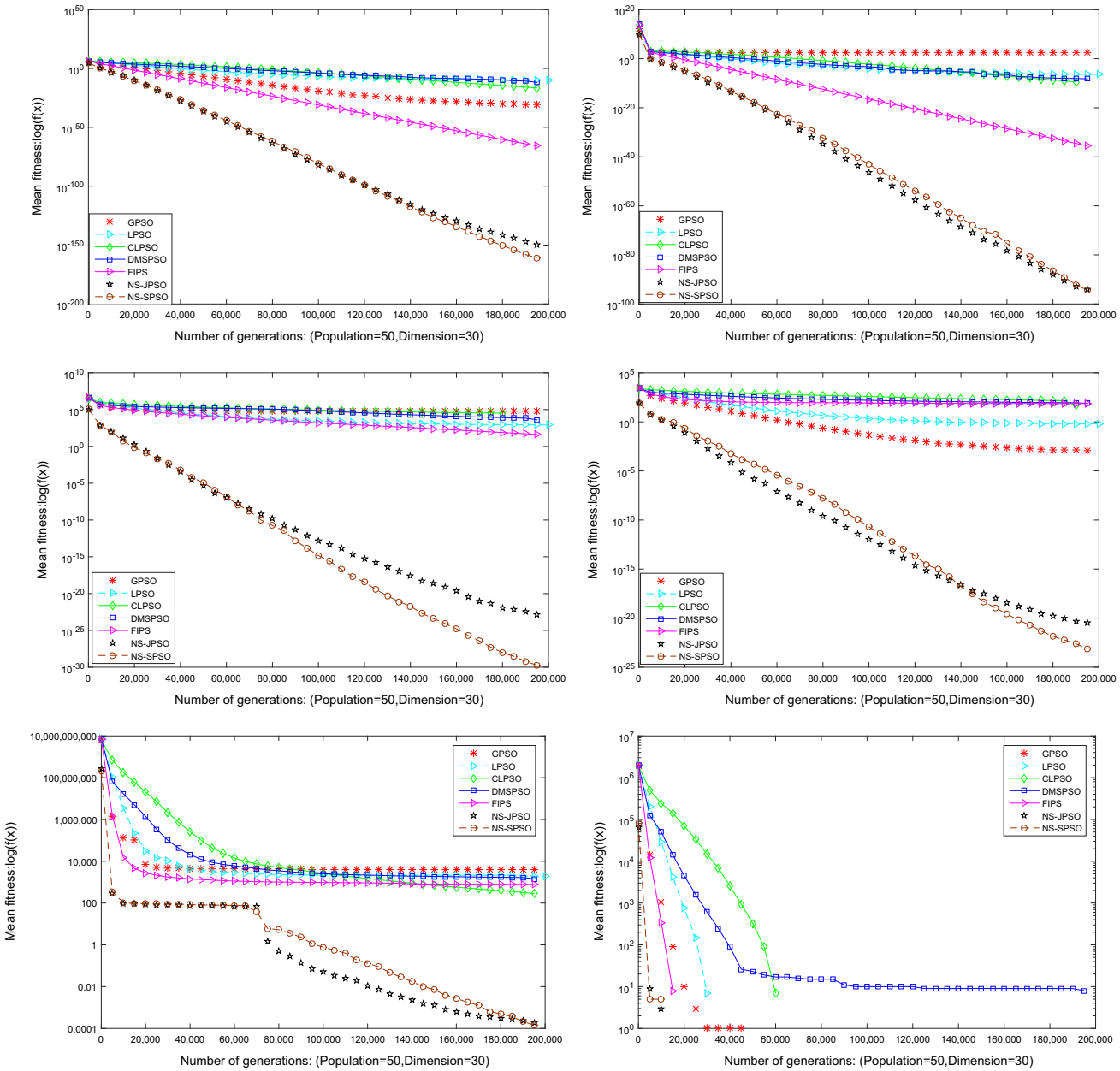

Fig. 5 Various test functions and their convergence profiles (the smallest values denote the best approaches); f_1 to f_6—from leftward to right-side and uppermost to bottommost. The proposed NS-SPSO algorithm has revealed the finest performance on ten out of twelve test functions (f_1 to f_3; f_5; f_6 and f_8 to f_{12}), together with 4 uni-modal and 6 multi-modal functions

posed algorithm is a gbest-based approach, which leads the population towards a single position. However, in multi-modal problems the lbest-based approaches would be more suitable (Chowdhury et al. 2014; Ghosh et al. 2012). We are working to consider the lbest-based version of the proposed NS-SPSO algorithm, in the near future. In the same setting, the performance of the NS-MJPSO is not good for this particular function. Similarly, for multi-modal function f_8, algorithms like NS-MJPSO, CLPSO, DMS-PSO and GPSO have the best performance. Based on our investigation, we found that the newly developed NS-SPSO needs some further parameters' adjustment in order to produce good and relatively optimal results, for these multi-modal functions. For other functions like f_9, f_{10}, f_{11} and f_{12}, algorithms like NS-SPSO, NS-MJPSO, CLPSO, DMS-PSO and GPSO have similar and almost comparable performance in terms of optimal evaluation values. Furthermore, NS-SPSO and NS-MJPSO are relatively faster than the other algorithms. In short, due to the gbest nature of the proposed NS-SPSO algorithm, the algorithm has performed well rather than other

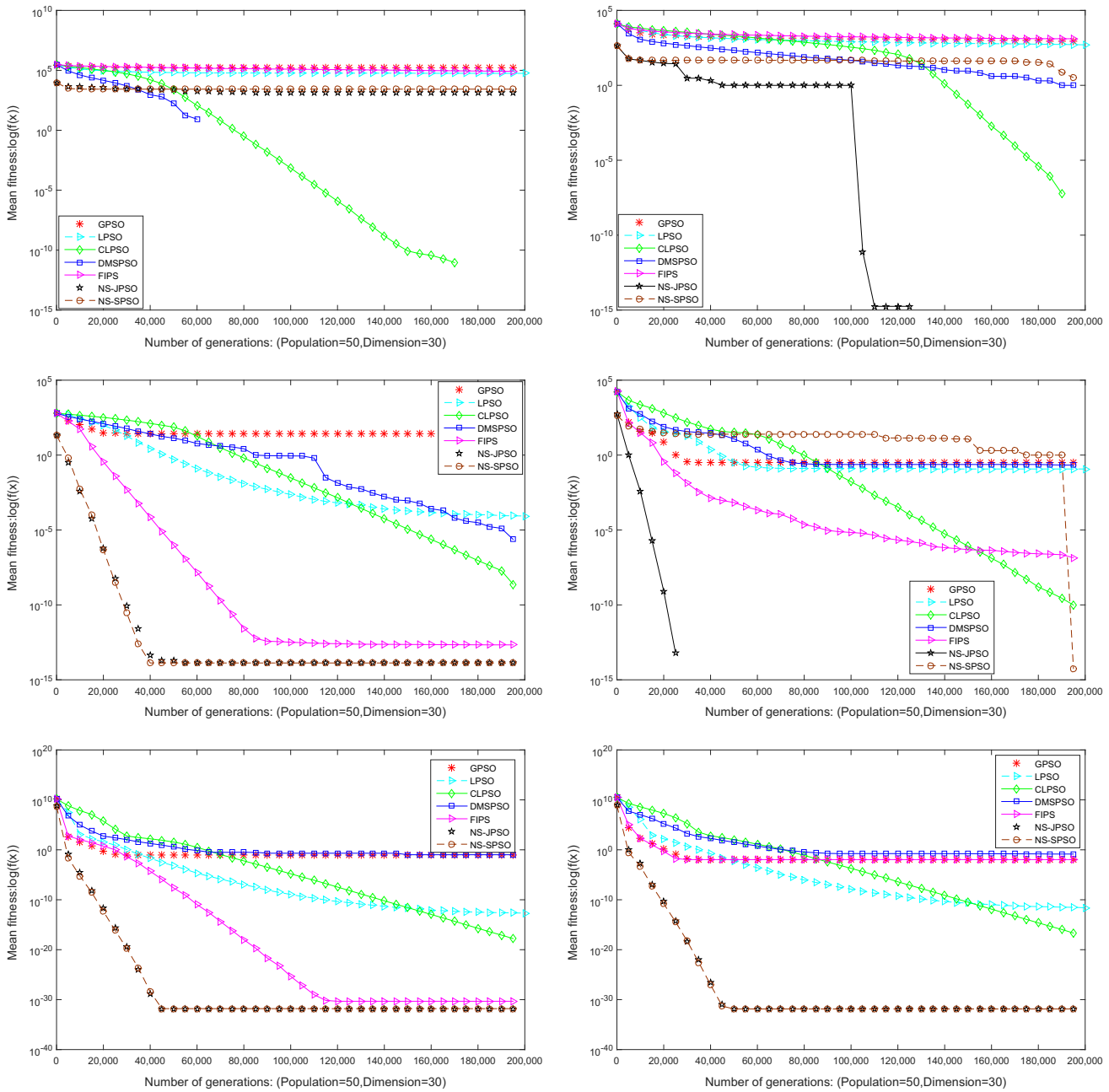

Fig. 6 Various test functions and their convergence profiles (the smallest values denote the best approaches); f_7 to f_{12}—from leftmost to rightmost and topmost to bottommost. The proposed NS-SPSO algorithm has revealed the finest performance on ten out of twelve test functions (f_1 to f_3; f_5; f_6 and f_8 to f_{12}), together with 4 uni-modal and 6 multi-modal functions

rivals for uni-modal functions. Unfortunately, we observed its worst performance for multi-modal functions. For scalable optimization, we consider high-dimensional optimization functions and several other PSO variants, as described in Sect. 5.4. The evaluation is being carried out using different setting for dimensionality and different values for N, i.e. the number of states.

5.2 Computation time of the proposed NS-SPSO

In this section, we analyse the average computational time of the proposed NS-SPSO algorithm and others. The computational time represents the CPU clock time, i.e. the absolute CPU time spent in evaluation—the lower its is, the faster is the approach. Further, all algorithms were experimentally evaluated with the same number of iterations, i.e. 2×10^5

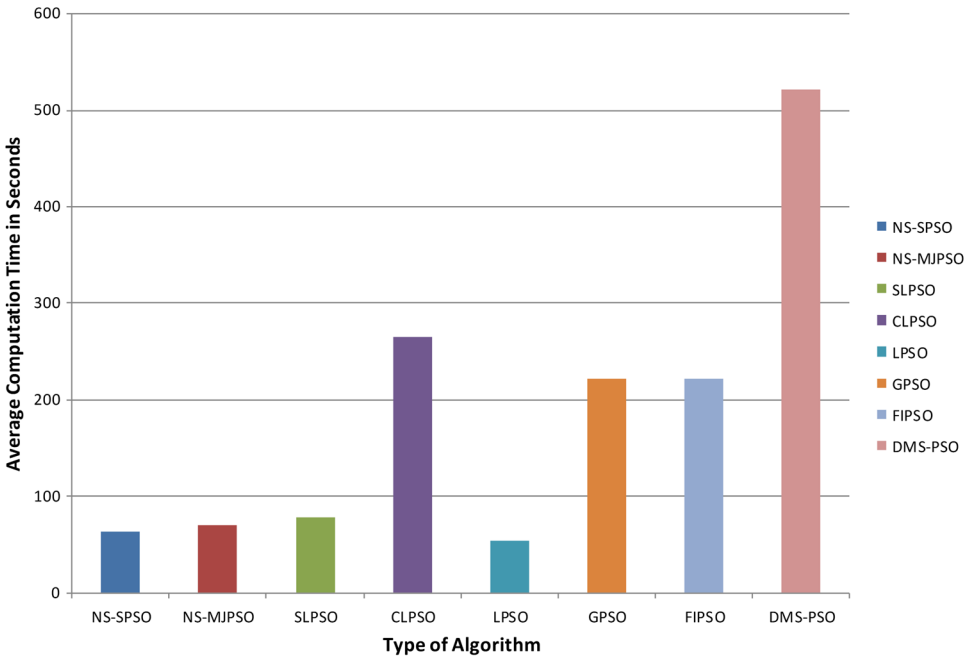

Fig. 7 Average computation of the proposed NS-SPSO in comparison

Table 4 Optimization errors using NS-SPSO for various number of states N on 10 basic test functions (f_1–f_{10}), the first value represents the mean value (μ) and the second value denotes the standard deviation (σ)

$N =$	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
4	6.86E−142	2.26E−100	1.54E−25	2.52E−20	1.36E−05	0.00E+00	3.25E−03	2.41E−03	1.45E−14	5.33E−15
	1.71E−161	3.46E−95	1.87E−30	7.63E−24	1.47E−04	0.00E+00	5.62E−04	3.00E−04	1.33E−14	4.55E−16
8	1.23E−150	1.89E−110	2.65E−30	3.32E−24	8.96E−06	0.00E+00	2.55E−04	5.60E−05	2.56E−15	4.11E−16
	2.51E−161	2.25E−98	3.77E−35	**2.55E−28**	**3.27E−05**	0.00E+00	6.88E−04	2.45E−05	**5.63E−16**	**2.22E−16**
12	3.54E−145	1.59E−120	2.58E−31	1.31E−24	3.66E−05	0.00E+00	2.16E−05	3.64E−04	6.55E−15	3.33E−15
	2.36E−160	**2.96E−100**	6.69E−30	5.77E−24	9.11E−04	0.00E+00	4.77E−04	**1.10E−05**	5.88E−14	4.22E−16
16	1.58E−142	6.70E−110	5.23E−28	1.65E−22	6.44E−04	0.00E+00	8.11E−03	5.66E−04	1.23E−14	7.99E−14
	2.69E−160	2.62E−95	**2.67E−35**	4.21E−26	3.55E−05	0.00E+00	**3.44E−04**	2.99E−04	3.55E−14	1.58E−15
20	1.76E−138	3.53E−98	6.41E−24	1.97E−15	3.56E−04	0.00E+00	1.10E−03	3.88E−03	4.65E−12	1.42E−13
	2.81E−158	5.31E−99	2.61E−25	4.83E−24	1.25E−04	0.00E+00	6.60E−03	4.77E−03	7.81E−12	1.48E−14
	4	**12**	**16**	**8**	**8**	**N/A**	**16**	**12**	**8**	**8**

Minimum μ values are best and shown in boldface; further, the last line shows the optimal number of states for each benchmark function

over the same 12 benchmark functions. The average computational times for all algorithms are shown in Fig. 7. The Local PSO (LPSO) has the smallest average computational time and stands at rank 1. The proposed NS-SPSO algorithm has the second shortest computational time and stands at rank 2. Note that, NS-SPSO also scores the second best, in terms of minimum values for most of the uni-modal and multi-modal problems, algorithm with respect to accuracy.

5.3 Impact of number of states on results

As described earlier, the contribution of our proposed algorithm is twofold: (i) the utilization of N states rather than four states used in the SPSO algorithm; and (ii) the adaptation of linearly decreasing inertia weight (LDIW) (Shi and Eberhart 1998b). In respect of (i), the number of acceleration coefficient also varies. If $N = 8$, then there will be total 8 values (pairs) for c_1 and c_2, where each pair of (c_1, c_2) relates to a particular state among the standard 4 states, i.e. jumping out, exploration, exploitation and convergence. In the same way, for $N = 12$ there will be twelve values (pairs) each one for c_1 and c_2. As described earlier in Sect. 4, the values of c_1, c_2 are pre-determined and tuned accordingly because they help particles in escaping from one state to others. Consequently, a variety of values will produce variations in outcomes. In order to automatically compute c_1, c_2, we have to use a simple approach where in every pair, c_1 value is decreased or increased with 0.05 and the related c_2 value is increased or

Table 5 Seven high-dimensional test functions and their optimization errors—for each algorithm and test function, the primary row characterizes the statistical average or mean value (μ) while the subsequent row designates the statistical standard deviation (σ)

	NS-SPSO		NS-MJPSO		SL-PSO		CCPSO2		DMS-L-PSO
100-D (Number of states = 12)									
f_{13}	1.33E−31		2.99E−31		1.09E−27		7.73E−14		0.00E+00
	3.68E−30	+	1.74E−29	+	3.50E−28	+	3.23E−14	–	0.00E+00
f_{14}	1.28E−07		3.52E−07		9.45E−06		6.08E+00		3.65E+00
	1.90E−07	+	1.89E−06	+	4.97E−06	+	7.83E+00	+	7.30E−01
f_{15}	3.96E+00		4.25E+00		5.74E+02		4.23E+02		2.83E+02
	1.43E+01	+	3.69E+01	+	1.67E+02	+	8.65E+02	+	9.40E+02
f_{16}	1.30E−06		2.75E−06		7.46E+01		3.98E−02		1.83E+02
	1.16E−06	+	1.23E−06	+	1.21E+01	+	1.99E−01	+	2.16E+01
f_{17}	0.00E+00		0.00E+00		0.00E+00		3.45E−03		0.00E+00
	0.00E+00	=	0.00E+00	=	0.00E+00	+	4.88E−03	=	0.00E+00
f_{18}	1.07E−20		1.68E−20		2.10E−14		1.44E−13		0.00E+00
	1.99E−18	–	2.47E−19	+	5.22E−15	+	3.06E−14	–	**0.00E+00**
f_{19}	−1.11E+03		−1.36E+03		−1.48E+03		−1.50E+03		−1.14E+03
	1.35E+00	+	2.99E+00	+	1.90E+01	+	1.04E+01	+	8.48E+00
+/=/–		**5/1/1**		**6/1/0**		**7/0/0**		**4/1/2**	
500-D (Number of states = 16)									
f_{13}	1.65E−30		2.76E−28		7.24E−24		7.73E−14		0.00E+00
	1.66E−29	+	1.39E−28	+	2.05E−25	+	3.23E−14	–	0.00E+00
f_{14}	9.55E−02		8.76E+00		3.47E+01		5.79E+01		6.89E+01
	3.65E−01	+	7.95E+00	+	1.03E+00	+	4.21E+01	+	2.01E+00
f_{15}	5.45E+03		2.21E+03		6.10E+02		7.24E+02		4.67E+07
	1.56E+02	+	1.99E+02	+	1.87E+02	–	**1.54E+02**	+	5.87E+06
f_{16}	2.46E−02		4.35E−01		2.72E+03		3.98E−02		1.61E+03
	1.93E−02	+	3.33E−01	+	3.25E+02	+	1.99E−01	+	1.04E+02
f_{17}	5.88E−20		1.22E−15		3.33E−16		1.18E−03		0.00E+00
	1.53E−20	+	3.22E−14	–	0.00E+00	+	4.61E−03	–	**0.00E+00**
f_{18}	2.52E−15		1.87E−14		1.46E−13		5.34E−13		2.00E+00
	1.99E−15	+	3.44E−14	+	2.95E−15	+	8.61E−14	+	9.66E−02
f_{19}	−1.36E+02		−2.77E+03		−5.94E+03		−7.23E+03		−4.20E+03
	4.26E+01	+	5.52E+02	+	1.72E+02	–	4.16E+01	–	**1.29E+01**
+/=/–		**7/0/0**		**6/0/1**		**5/0/2**		**4/0/3**	

decreased with the same value. For example, beneath we are describing eight values for each pair of c_1, and c_2 when $N = 8$.

$$c_1, c_2 = \begin{cases} 2, 2.05, 2.1, 2.15, 2.2, 2.25, 1.8, 1.85 \\ 2, 1.95, 1.9, 1.85, 1.8, 1.75, 2.2, 2.15 \end{cases}$$

We have found that increasing the N value for high-dimensional problems even increases the convergence speed and accuracy, but at a small increase in computational time. Note that, accuracy reflects the total number of positive hits for the least fitness evaluation test in 30 experimental hits. In addition, it is noted that no further improvement is made when increasing the value of N, in particular, for low-dimensional problems. For instance, the least function evaluation value for f_0, with 30 dimensions, was achieved at $N = 12$; thus, fixing $N = 20$ just increases the algorithm's computational time with no benefits and performance gains. Likewise, as the number of N states grows, the effort involved in choosing sufficient acceleration coefficients is also increasing. Table 4 depicts fitness values for ten test functions (from f_1 to f_{10}) when considered for optimization using the suggested PSO variant (NS-SPSO) and 5 different values for states, i.e. N. Because, for both functions, i.e. f_{11} and f_{12}, we did not overlooked any considerable improvements (minimization or maximization), thus they are not listed in Table 4.

Table 5 continued

	NS-SPSO		NS-MJPSO		SL-PSO		CCPSO2		DMS-L-PSO
1000-D (Number of states = 20)									
f_{13}	5.47E−28		2.33E−25		7.10E−23		5.18E−13		0.00E+00
	4.36E−28	+	1.66E−24	+	1.40E−24	+	9.61E−14	–	0.00E+00
f_{14}	2.55E+00		4.10E+01		8.87E+01		7.81E+01		4.25E+01
	1.66E+00	+	3.22E+01	+	5.25E+00	+	2.35E+01	–	**2.35E−01**
f_{15}	2.20E+01		3.22E+02		1.04E+03		1.22E+03		7.33E+09
	1.89E+01	–	**1.57E+01**	+	5.14E+01	+	1.63E+02	+	3.28E+08
f_{16}	1.77E−02		2.67E−01		5.89E+02		1.88E−01		3.44E+03
	1.64E−02	+	3.65E−01	+	9.26E+00	+	3.04E−01	+	1.61E+02
f_{17}	1.91E−20		3.87E−18		4.44E−16		1.08E−03		0.00E+00
	2.34E−20	+	2.40E−18	–	0.00E+00	+	2.37E−03	–	**0.00E+00**
f_{18}	1.38E−17		2.48E−16		3.44E−13		1.22E−12		2.76E+00
	2.58E−16	+	1.36E−15	+	5.32E−15	+	1.56E−13	+	7.88E−02
f_{19}	−2.33E+01		−4.63E+02		−1.30E+04		−1.53E+04		−5.50E+03
	1.51E+01	+	3.46E+02	+	1.04E+02	+	2.58E+01	+	2.66E+01
+/=/–		**6/0/1**		**6/0/1**		**7/0/0**		**4/0/3**	

The smallest value for μ denote the best values and is shown in boldface; furthermore, the ranking formula was used to rank the NS-SPSO algorithm against other variants

5.4 Results for high-dimensional problems

Besides the above twelve benchmarks functions, NS-SPSO is further tested on seven more functions by setting the search dimensionality to 500-D and 1000-D, respectively (Tang et al. 2007). For low-dimensional (20-D) optimization problems, NS-SPSO has shown robust performance on 12 benchmark functions in comparison with five representative PSO variants. However, in order to verify the scalability of the proposed NS-SPSO, we are keen to further investigate its performance on large-scale (high-dimensional) optimization problems, whose search dimensionality is normally larger than 100. For this purpose, SL-PSO is tested on a large-scale optimization test set (denoted as f_{13} to f_{19}), which was proposed in the CEC'08 special session on large-scale optimization. Afterwards, SL-PSO is further tested on f_{13} to f_{19} by setting the search dimensionality to 100-D, 500-D, and 1000-D, respectively.

Four different algorithms, based on their evaluation for large-scale problems, were considered for this evaluation and comparison. Amongst the four, CCPSO2 (Li and Yao 2011) is the most widely used state-of-the-art PSO variant for large-scale optimization. Similarly, the DMS-L-PSO is the enhanced version of the DMS-PSO with a local search operator. DMS-PSO is described in Sect. 2. Moreover, NS-MJPSO (Rahman et al. 2020) is our own developed version of the PSO. The last one is the SL-PSO (Cheng and Jin 2015) that have outperformed for high-dimensional problems. We varied the number of states (N) for the proposed NS-SPSO algorithm in various runs, and the best results were summarized in Table 5.

The results obtained in Table 5 show that, on average, NS-SPSO performs better than the NS-MJPSO, DMS-PSO and CL-PSO. For example, for 100-dimensional functions, it outperfomed all the closest rivals. Similarly, for 500-dimensional functions, its performance is guaranteed but comparable to the NS-MJPSO algorithm. Moreover, for 1000-dimensional functions, largely, the proposed algorithm outperformed the other ones. However, the DMS-L-PSO is always able to find the real global optimum, regardless of the dimension, although it performs not so well on the other five test functions. Further, the proposed algorithm has comparable performance with SL-PSO and DMS-L-PSO. On the other hand, NS-SPSO outperforms NS-MJPSO and CCPSO2. The reasons for such out performance include: (i) the increasing number of states produces chances for state switching; (ii) adequate acceleration coefficients for particular states; and (iii) the large number of parameters set controls sensitivity of the algorithm.

In order to demonstrate that there are statistical significant differences amongst the results produced by the proposed algorithms and the closest ones, particularly, SL-PSO and DMS-L-PSO, a t test was performed over the data gathered in various runs. The confidence interval is set to 95%. The t test result (P value) shows that there are no significant differences amongst these; hence, NS-SPSO is comparable to these two algorithms.

6 Conclusions and future work

In this paper, we have proposed a novel PSO variant called *N* state switching particle swarm optimization (NS-SPSO) algorithm. The algorithm has combined the evolutionary method for population distribution with the traditional PSO algorithm. The performance of the newly developed NS-SPSO algorithm is demonstrated through various metrics over evaluating 12 various benchmark functions. These benchmark functions include several unimodal, multi-modal and nonlinear type problems. Furthermore, several variants of the classical PSO were coded, re-implemented, which are widely known for their best performance and capabilities to solve large-scale optimization problems. These include GPSO, LPSO, FIPS, DMS-PSO and CL-PSO. The newly developed NS-SPSO algorithm is also used in the tournament for the same objective functions. The significance of the proposed algorithm was demonstrated though analysing statistical results obtained on various benchmark functions. Our empirical evaluation suggests that NS-SPSO algorithm has the shortest computational time, fast convergence ratio and scores for the second best method in terms of accuracy.

It is notable that the presented NS-SPSO algorithm could be useful to a diversity of modern optimization problems; for example, power systems and healthcare informatics, in which the precision and accurateness is the key fear of these optimization problems. We intend to put on the proposed NS-SPSO algorithm to the well-known economic load dispatch (ELD) issue with the purpose to decline and minimize the total power generation costs (Rahman 2016). Furthermore, in order to select suitable values of the control parameters, we will also examine the steadiness among the number of evolutionary states *N* and the total computational cost. It is notable that as the total number of states *N* rises, the choices for the acceleration coefficients also rises consequently. In similar situations, the programmed and adaptive approximation of the control parameters would be the chief priority task in forthcoming research. Similarly, in future research we will consider NS-SPSO for: complex many-objectives, high-dimensional, optimization problems (Han et al. 2019); and further improve its computational time. Moreover, we observed through experimentation that the proposed algorithm does not perform well for multi-modal function due to the gbest version of PSO implementation. In the near future, we look forward to preparing the *lbest* version of the NS-SPSO algorithm for multi-modal functions.

Acknowledgements This work is supported by Abdul Wali Khan University, Mardan (AWKUM). The research was conducted as part of the Ph.D. program, at Brunel University London, UK, under the supervision of Prof. Zidong Wang (Fellow IEEE). The implementation code of the proposed NS-SPSO algorithm is available on the GitHub repository (https://github.com/izazhere/Research_Project).

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants nor any studies with animals, performed by any of the authors.

References

Abualigah LM, Khader AT (2017) Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J Supercomput 73(11):4773–4795

Abualigah LM, Khader AT, Hanandeh ES (2018) A new feature selection method to improve the document clustering using particle swarm optimization algorithm. J Comput Sci 25:456–466

Alswaitti M, Albughdadi M, Isa NAM (2018) Density-based particle swarm optimization algorithm for data clustering. Expert Syst Appl 91:170–186

Brits R, Engelbrecht AP, van den Bergh F (2007) Locating multiple optima using particle swarm optimization. Appl Math Comput 189(2):1859–1883

Cheng R, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291:43–60

Cheng R, Sun C, Jin Y (2013) A multi-swarm evolutionary framework based on a feedback mechanism. In: 2013 IEEE Congress on evolutionary computation. IEEE, pp 718–724

Chowdhury A, Zafar H, Panigrahi BK, Krishnanand KR, Mohapatra A, Cui Z (2014) Dynamic economic dispatch using Lbest-PSO with dynamically varying sub-swarms. Memet Comput 6(2):85–95

Ciuprina G, Ioan D, Munteanu I (2002) Use of intelligent-particle swarm optimization in electromagnetics. IEEE Trans Magn 38(2):1037–1040

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, vol 1. New York, NY, pp 39–43

Eberhart RC, Shi Y (2001) Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 congress on evolutionary computation, 2001, vol. 1. IEEE, pp 81–86

Eberhart RC, Shi Y (2004) Guest editorial special issue on particle swarm optimization. IEEE Trans Evol Comput 8(3):201–203

Elijah P (2012) Optimization: algorithms and consistent approximations, vol 124. Springer, Berlin

Ghosh A, Chowdhury A, Sinha S, Vasilakos AV, Das S (2012) A genetic Lbest particle swarm optimizer with dynamically varying subswarm topology. In: 2012 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–7

Han D, Wenli D, Wei D, Jin Y, Chunping W (2019) An adaptive decomposition-based evolutionary algorithm for many-objective optimization. Inf Sci 491:204–222

Higashi N, Iba H (2003) Particle swarm optimization with Gaussian mutation. In: Swarm intelligence symposium, 2003. SIS '03. Proceedings of the 2003. IEEE, pp 72–79

Ho S-Y, Lin H-S, Liauh W-H, Ho S-J (2008) OPSO: orthogonal particle swarm optimization and its application to task assignment problems. IEEE Trans Syst Man Cybern Part A Syst Hum 38(2):288–298

Hu L, Wang Z, Rahman I, Liu X (2015) A constrained optimization approach to dynamic state estimation for power systems includ-

ing PMU and missing measurements. IEEE Trans Control Syst Technol PP(99):1–1
Juang C-F (2004) A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern Part B Cybern 34(2):997–1006
Kenndy J, Eberhart RC (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw 4:1942–1948
Kennedy J (1999) Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of the 1999 congress on evolutionary computation, vol 3, 1999. CEC 99, p 1938
Kennedy J, Mendes R (2002) Population structure and particle swarm performance. In: Proceedings of the 2002 congress on evolutionary computation, vol 2, 2002. CEC '02, pp 1671–1676
Kennedy J, Kennedy JF, Eberhart RC (2001) Swarm intelligence. Morgan Kaufmann, Burlington
Khan AA, Zakarya M, Khan R, Rahman IU, Khan M et al (2020) An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters. J Netw Comput Appl 150:102497
Krohling RA, dos Santos Coelho L (2006) Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. IEEE Trans Syst Man Cybern Part B Cybern 36(6):1407–1416
Li X, Yao X (2011) Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans Evol Comput 16(2):210–224
Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer. In: Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005. IEEE, pp 124–129
Liang JJ, Kai Qin A, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295
Ling H-L, Jian-Sheng W, Zhou Y, Zheng W-S (2016) How many clusters? A robust pso-based local density model. Neurocomputing 207:264–275
Liu B, Wang L, Jin Y-H (2007) An effective PSO-based memetic algorithm for flow shop scheduling. IEEE Trans Syst Man Cybern Part B Cybern 37(1):18–27
Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evol Comput 8(3):204–210
Ozcan E, Mohan CK (1999) Particle swarm optimization: surfing the waves. In: Proceedings of the 1999 congress on evolutionary computation, 1999. CEC 99, vol 3. IEEE
Qu B-Y, Suganthan P, Das S (2013) A distance-based locally informed particle swarm model for multimodal optimization. IEEE Trans Evol Comput 17(3):387–402
Rahman IU (2016) Novel particle swarm optimization algorithms with applications in power systems. Ph.D. thesis, Brunel University London
Rahman IU, Wang Z, Liu W, Ye B, Zakarya M, Liu X (2020) An n-state markovian jumping particle swarm optimization algorithm. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/TSMC.2019.2958550
Robinson J, Sinton S, Rahmat-Samii Y (2002) Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In: Antennas and propagation society international symposium, , vol 1, 2002. IEEE, pp 314–317
Shelokar PS, Siarry P, Jayaraman VK, Kulkarni BD (2007) Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Appl Math Comput 188(1):129–142
Shi Y, Eberhart R (1998a) A modified particle swarm optimizer. In: The 1998 IEEE international conference on evolutionary computation proceedings, 1998. IEEE World Congress on Computational Intelligence. IEEE, pp 69–73
Shi Y, Eberhart RC (1998b) Parameter selection in particle swarm optimization. In: Evolutionary programming VII. Springer, pp 591–600
Shi Y, Eberhart RC (1999) Empirical study of particle swarm optimization. In: Proceedings of the 1999 congress on evolutionary computation, vol 3, 1999. CEC 99. IEEE
Suganthan PN (1999) Particle swarm optimiser with neighbourhood operator. In: Proceedings of the 1999 congress on evolutionary computation, vol 3, 1999. CEC 99. IEEE
Suganthan PN, Hansen N, Liang JJ, Deb K, Y-Po C, Anne A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC. Special session on real-parameter optimization. KanGAL report 2005005:2005
Tang K, Yáo X, Suganthan PN, MacNish C, Chen Y-P, Chen C-M, Yang Z (2007) Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Nat Inspired Comput Appl Lab USTC China 24:1–18
Tang Y, Wang Z, Fang J (2011) Parameters identification of unknown delayed genetic regulatory networks by a switching particle swarm optimization algorithm. Expert Syst Appl 38(3):2523–2535
Törn A, Žilinskas A (1989) Global optimization, vol 350. Springer, Berlin
Valdez F, Melin P, Castillo O (2014) Modular neural networks architecture optimization with a new nature inspired method using a fuzzy combination of particle swarm optimization and genetic algorithms. Inf Sci 270:143–153
Van den Bergh F, Petrus Engelbrecht A (2006) A study of particle swarm optimization particle trajectories. Inf Sci 176(8):937–971
Wang Z, Hu L, Rahman I, Liu X (2013) A constrained optimization approach to dynamic state estimation for power systems including PMU measurements. In: 2013 19th international conference on automation and computing (ICAC). IEEE, pp 1–6
Weber TO, Van Noije Wilhelmus AM (2012) Design of analog integrated circuits using simulated annealing/quenching with crossovers and particle swarm optimization. In: Simulated Annealing Advances, Applications and Hybridizations. https://doi.org/10.5772/50384
Weibo L, Zidong W, Xiaohui L, Nianyin Z, David B (2018) A novel particle swarm optimization approach for patient clustering from emergency departments. IEEE Trans Evol Comput 23:632–644
Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
Zakarya M, Gillam L (2019) Modelling resource heterogeneities in cloud simulations and quantifying their accuracy. Simul Model Pract Theory 94:43–65
Zhan Z-H, Xiao J, Zhang J, Chen W (2007) Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis. In: IEEE congress on evolutionary computation, 2007. CEC 2007. IEEE, pp 3276–3282
Zhan Z-H, Zhang J, Li Y, Chung HS-H (2009) Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern Part B Cybern 39(6):1362–1381
Zhang J, Chung HS-H, Lo W-L (2007) Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE Trans Evol Comput 11(3):326–335