Neural Computing and Applications (2024) 36:7471–7489
https://doi.org/10.1007/s00521-024-09472-w

ORIGINAL ARTICLE

Hybrid particle swarm optimization algorithm for text feature selection problems

Mourad Nachaoui[2] · Issam Lakouam[1] · Imad Hafidi[1]

Received: 4 July 2023 / Accepted: 14 January 2024 / Published online: 19 February 2024

Abstract
Feature selection (FS) is a crucial preprocessing step that aims to eliminate irrelevant and redundant features, reduce the dimensionality of the feature space, and enhance clustering efficiency and effectiveness. FS is categorized as NP-Hard due to the high number of existing solutions. Various metaheuristic methods have been developed to address the FS problem, yielding promising results. Particularly, particle swarm optimization (PSO), an evolutionary computing (EC) approach guided by swarm intelligence, has gained widespread adoption owing to its implementation simplicity and potential for global search. This paper analyzes several variants of PSO algorithms and introduces a new FS method called HPSO. The proposed approach utilizes an asynchronously adaptive inertia weight and an improved constriction factor. Additionally, it incorporates a chaotic map and a MAD fitness function with a feature count penalty to tackle the clustering FS problem. The efficiency of the developed method is evaluated against the genetic algorithm (GA) and well-known variants of PSO algorithms, including PSOs with fixed inertia weights, PSOs with improved inertia weights, PSOs with fixed constriction factors, PSOs with improved constriction factors, PSOs with adaptive inertia weights, and PSO's includes advanced learning exemplars and sophisticated structure topologies. This paper assesses two different reference text data sets, Reuters-21578 and Webkb. In comparison with competitive methods, the proposed HPSO method achieves higher clustering precision and selects a more informative feature set.

Keywords Text feature selection · Particle swarm optimization algorithm · Genetic algorithm · Constriction factor · Chaotic map · *K*-mean text clustering algorithm

1 Introduction

The quantity of textual data generated in recent years on the internet has exploded exponentially due to the technology evolution which affects the process of grouping text documents [1]. Text clustering methods are unsupervised algorithms, which aim to process a large amount of documents text and group them into a predetermined number of clusters [2], so that each group contains similar documents (documents within the same cluster have a higher degree of similarity than documents in other clusters). Both relevant and non-informative features are included in text clustering data sets, with the redundant, useless, and noisy non-informative features having the potential to impair the accuracy and computing performance of the clustering technique [3]. Features selection technique can be used to address these by chosen an best subset of relevant attributes among a wide set of features. Therefore, FS is a primary and important task, which is the inevitable part of data mining that deals with the curse of dimensionality. Moreover, these methods are designed to enhance accuracy of clustering approach and minimize the number of non-informative features for each document. Many areas in text extraction are supported by the FS approach [4], including

✉ Mourad Nachaoui
m.nachaoui@usms.ma; mourad.nachaoui@univ-nantes.fr

Issam Lakouam
issam.lakouam.info@gmail.com

Imad Hafidi
imad.hafidi@gmail.com

[1] National School of Applied Science ENSA, 25000 Khouribga, Morocco

[2] Equipe de Mathématiques et Interactions, Faculté des Sciences et Techniques, Université Sultan Moulay slimane, Beni-Mellal, Marocco

text clustering, classification, text categorization, information retrieval, etc.

This paper introduces a hybrid PSO algorithm named HPSO to enhance the clustering accuracy. The developed method is used for reducing the number of non-informative features to enhance clustering technique and effectiveness based on an asynchronously adaptive inertia weight and improved constriction factor. Thus, to improve text clustering accuracy, a chaotic map and the mean absolute difference (MAD) fitness function with a feature count penalty are introduced. Chaotic maps play a major role in enhancing evolutionary algorithms to prevent the local optima and accelerate the convergence [5]. Concretely, the HPSO is used in each document to generate new subsets of useful text features, which are then combined to be the input for the *k*-means text clustering approach, which is a popular unsupervised procedure thanks to its speed convergence [2, 6]. The methodology of numerical validation of the HPSO is conducted on two different popular text data sets Reuters-21578 [7] and Webkb [8]. The performance of HPSO is examined through its comparison with genetic algorithm [9] and recent various PSO algorithms [1, 10]. The experiment results show that the proposed HPSO is better than the competitive approaches in terms of Precision, Recall, *F*-measures, and Accuracy measures.

The rest of this paper is organized as follows. Section 2 reviews literature on PSO and genetic algorithm. Section 3 is devoted to the text preprocessing steps, the FS using PSO method, the proposed approaches HPSO, and the *k*-means clustering technique. The numerical experiments results are illustrated in Sect. 4. Finally, the conclusion is given in Sect. 5.

2 State of art

Numerous methodologies have been proposed to make the problem of selecting characteristics more effective. These approaches can be classified into wrapper, filter, embedded, and hybrid methods [11]. The difference between these three techniques is whether a learning approach is applied and how it is used. Wrapper approaches are dependent on the used learning algorithm, making classification. In this process, a search strategy is applied to generate subsets of features and a learning algorithm to examine the accuracy of the selected features subsets [12]. This method is intended to grow accuracy by inserting or eliminating features from the subset consecutively. Since the clustering technique is used in each evaluation, wrapper methods outperform other approaches in terms of accuracy [13]. However, the action of reaching accuracy improvement typically suffers from overfitting, and they are computationally expensive. Moreover, these methods are not generic. Indeed, the elected subset of features is significantly reliant on the clustering algorithm employed to measure quality. Therefore, any modification in this algorithm leads to re-execution of the FS algorithm. Moreover, filter approaches are based on some statistics tools, like correlation and consistency, etc., to evaluate the set of features in order to generate a new subset of informational text features, without interacting with learning techniques [1, 14]. An optimal feature subset is derived by eliminating the features that perform poorly. While there is no reliance upon classifiers, these approaches tend to be faster, simpler, and less accurate than classifiers [15]. Hybrid approaches [16] combine filter methods and wrapper to select relevant features. Based on the learning technique's variable selection process, embedded approaches automatically generate and evaluate new feature subsets. As a result, the computation cost and clustering accuracy of embedded methods are situated between wrappers and filters [17].

Metaheuristic algorithms appear to be extensively employed techniques for enhancing feature selection (FS) methods. Notably, various well-known evolutionary algorithms are employed in these studies to overcome the challenge of prior works getting stuck in local optima. These include PSO [18–20], GA [21–23], ant colony optimization (ACO) [24, 25], differential evolution (DE) [26], among others.

The PSO approach was initially developed in [27]; it is an optimization method based on swarm intelligence, which mimics species social behavior, such as bird flocking. Thus, in order to find the optimum solution, PSO is positioned in the search space for a FS problem using a swarm of particles, where each particle's movement is determined by its own velocity and also the movement experiences of other particles. The authors introduced two variations of the PSO algorithm: (1) "GBEST" approach, where each particle keeps track of the best solution found on the swarm, and (2) "LBEST" approach, wherein each particle follows the best solution found upon its neighbors. In 1998, another study proposes a new parameter called inertia weight to boost the effectiveness of the original PSO [28], along with two models for the PSO algorithm. In the first model, different values of the inertia parameter were tested using the fixed inertia weight PSO algorithm, whereas the second model proposes a time-decreasing inertia weight. The experiments results demonstrated that the developed method brings a good improvement in the PSO performance. PSO has been successfully used to resolve many optimization problems such as medical care problems [29], image processing [30], cloud computing [31], and wireless networks [32]. PSO is also used to improve classification accuracy in support vector machine (SVM) by determining parameters and FS of the SVM [3]. In another study, a multi-swarm PSO algorithm is

introduced to diminish the number of informational features and improve classification performance, where a multi-swam strategy is applied to PSO for parameter determination and FS in SVM [33].

There are several variants of the PSO algorithm that have been proposed to improve the convergence and learning abilities of this algorithm, to overcome some of its drawbacks, such as premature convergence and getting stuck in local optima [34]. In this case, the authors [35] have investigated four PSO variants and introduced two PSO models. These variants and models include: (1) the first variant used a fixed inertia weight [35, 36], (2) the second variant used a functional inertia weight [35, 37], (3) the third variant used a fixed constriction factor [38], (4) the fourth variant used a functional constriction factor [38, 39], (5) the first proposed model used a synchronously inertia weight and constriction factor, and (6) the final model used an asynchronously inertia weight and constriction factor. Through some experiment results, the authors establish that the sixth PSO model overwhelms the other comparative approaches in terms of classification accuracy across different feature dimensions. Another study proposes integrating opposition-based initialization, chaotic strategy, fitness-based dynamic inertia weight, and mutation into binary PSO (BPSO), to enhance the global search capability of the PSO [2]. The presented experiments results show that the introduced approach outperforms BPSO, chaotic BPSO (CBPSO), simple GA (SGA), and adaptive inertia weight PSO (AIWPSO) in terms of convergence speed and the accuracy of clustering. On the other hand, the authors of the paper [1] have proposed (FSPSOTC), an improved inertia weight PSO algorithm with MAD as a fitness function to resolve the text FS problem. This allows improvement in the performance of the text clustering technique and reduces the computing time. The introduced method is compared with genetic algorithm (GA), harmony search approach (HS), and *k*-mean clustering without any FS method on six Benchmark data sets. The effectiveness of the FSPSOTC in terms of text clustering technique has been demonstrated by some numerical experiments.

The GA and the PSO are two algorithms that are frequently applied to solve difficult optimization problems [40–42]. Crossover and mutation are powerful elements of the GA. Exploration and exploitation are handled by these elements in the algorithm. In some research, crossover and mutation are incorporated into the PSO algorithm to enhance its search capabilities. The FS problem is solved in [43] by integrating GA with PSO. The bare bone PSO for the FS problem is used in [44]. The GA for FS for credit card fraud detection is used in [45]. In [46] a fast genetic algorithm for feature selection-A qualitative approximation approach is proposed.

For this method, particles' local leaders are updated using reinforced memory. Moreover, it is proposed that a uniform combination of crossovers and mutations should be applied to average out exploration and exploitation of the algorithm.

Recently, PSO has witnessed numerous advancements. Surprisingly Popular Algorithm-based Adaptive Euclidean Distance-based Topology Learning Particle Swarm Optimization (SpadePSO) is a more recent addition to the PSO family and incorporates innovative learning exemplars and structural topologies, distinguishing itself from conventional PSO approaches [10]. Its design aims to address challenges faced by traditional PSO algorithms, providing enhanced exploration and exploitation capabilities. As we delve into the landscape of feature selection for text clustering, this study extends beyond established PSO variants, including a comparative analysis with SpadePSO to elucidate the distinctive contributions of the proposed Hybrid PSO (HPSO) and Improved Inertia Weight PSO (IIWPSO). This comparative exploration seeks to shed light on the evolving dynamics within the realm of PSO-based optimization, offering a comprehensive perspective on the state-of-the-art algorithms.

3 Methodology

The digital format has led to a gradual increase in the volume of text documents, making text clustering a crucial approach for organizing them. The main objective of a text clustering algorithm is to group these documents based on their inherent features. To achieve this, certain common preprocessing steps are applied to the documents before clustering. These standard preprocessing steps include tokenization, eliminating stop words, stemming, and term weighting, which are used to transform the documents into a suitable format. In this section, we provide a concise overview of these preprocessing steps. Then, we present the PSO method for the features selection problem.

Next, we introduce the developed HPSO. The goal is to improve clustering efficiency and effectiveness by reducing the number of irrelevant and redundant text features for each document. Documents are represented using a standard model called the vector space model (VSM), and the TF-IDF is the metric used to evaluate the importance of terms in the clustering process. Finally, we give the clustering technique based on the *k*-means approach used to examine the performance of the FS methods.

3.1 Preliminaries

The text documents are transformed into numerical representations during the preprocessing processes [2].

Tokenization, stop-word elimination, stemming, term weighting, and VSM representation are the five steps. The subsequent subsections provide an overview of various preprocessing stages.

3.1.1 Tokenization

A text document is broken into tokens, each of which represents a single term or a group of related terms, in the tokenization process. A single term is applied for vector space model representation in this work.

3.1.2 Stop words elimination

Stop words build a family of frequent words such as "the," "is," "an," "by" and other recognizable terms that appear frequently in text texts but contain little information for the clustering process. As a result, it is vital to get rid of them because they provide more features and make the text clustering approach perform worse.

3.1.3 Stemming

It is possible to get a word's grammatical root form by stemming it from its inflectional or derivative forms. For instance, the words "information," "informations," and "informative" all share the same root, "inform." The list of potential stemming techniques can be found in [47]. In this work, stemming tasks are carried out using the porter stemmer.

3.1.4 Term weighting

Using a procedure called term weighting, words in documents are converted into a numerical vector representation. This task has been accomplished using a variety of term weighting procedures, but the most well-liked method is term frequency inverse document frequency (TF-IDF). The TF-IDF is a metric that assesses how crucial a word is in separating the contents of the documents [1]. When a term is used frequently and just sometimes in a few documents, this value rises. The use formula to compute the word-weighting is defined as follows:

$$w_{k,l} = tf_{k,l} \times idf_{k,l} = tf_{k,l} \times \log\left(\frac{n}{df_l}\right). \tag{1}$$

where $tf_{k,l}$ represents the number of times l^{th} term appears in the kth document, n represents the total number of documents in the data set, and df_l represents the number of documents that include the l^{th} term.

3.1.5 VSM

The VSM is a common model to represent documents as vectors of weights, where each term weight represents the weight that a word should have in the clustering process. The following expression illustrates how documents are represented in this paper using the VSM:

$$\text{VSM} = \begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,j} & \cdots & \phi_{1,t} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{i,1} & \cdots & \phi_{i,j} & \cdots & \phi_{i,t} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{n,1} & \cdots & \phi_{n,j} & \cdots & \phi_{n,t} \end{bmatrix} \tag{2}$$

3.1.6 Standard PSO

PSO is a sophisticated optimization approach that falls under category of population-based meta-heuristic methods. It is modeled after how a flock of birds or a school of fish would exploit and explore a problem space to find food. Swarm is the name given to the PSO population, and each member of the swarm is abstracted as a particle. The positioning of the particle i is designed as an $\mathcal{N}$-dimensional vector $\zeta_i = (\zeta_1, \zeta_2, \ldots, \zeta_{\mathcal{N}})$, and its velocity is the vector denoted by $\upsilon_i = (\upsilon_1, \ldots, \upsilon_{\mathcal{N}})$. The position ζ and velocity vector υ are initialized randomly and adjusted during each iteration using the position's own well-established best position ($\mathcal{P}optim_j$). The current global best position is represented by the swarm's best position, which is denoted by ($\mathcal{G}optim_j$). The original PSO concept makes use of both the individual and the present global best. The position that has lowest cost, denoted by $\mathcal{F}$, is what we called the best position. The updating strategy is written as follows:

$$\upsilon_{i+1} = \phi\upsilon_i + \gamma_1\xi_1(\mathcal{P}optim_i - \zeta_i) + \gamma_2\xi_2(\mathcal{G}optim_i - \zeta_i) \tag{3}$$

$$\zeta_{i+1} = \zeta_i + \upsilon_{i+1}. \tag{4}$$

where ξ_1 and ξ_2 are two uniformly distributed random numbers in [0, 1]. The inertia weight denoted by ϕ, regulates how much the prior velocity will have an impact. This value is crucial for achieving a balance between the algorithm's capacity for exploration and exploitation. Additionally, the parameters γ_1 and γ_2 stand for the acceleration coefficients that regulate, respectively, self-awareness and social impact. Algorithm 1 summarizes the common PSO procedures.

Algorithm 1 PSO algorithm

Require: Given the PSO parameters γ_1, γ_2, w, $\mathcal{N}p$ the size of population.
Initialization

(a) $k \leftarrow 0$: for the particle $j = 1, \ldots, \mathcal{N}p$, **do**
(b) The particle's position x_j^k and the velocity υ_j^k are randomly initialized.
(c) Use the fitness function $\mathcal{F}$ to evaluate particle ζ_j^k.
(d) Initialize $\mathcal{P}optim_j^k = \zeta_j^k$.
(e) Initialize $\mathcal{G}optim_j^k = \operatorname{argmin} \mathcal{F}(\zeta_j^k)$ corresponds to the minimal value of the swarm k.

while Termination criteria **do**
For the particle $j = 1, \ldots, \mathcal{N}p$, **do**

(a) Select ξ_1, ξ_2.
(b) Update the velocity υ_j^k with (3).
(c) Update ζ_j^k with (4).
(d) Use the fitness function $\mathcal{F}$ to evaluate particle ζ_j^k.
(e) If $\mathcal{F}(\mathcal{P}optim_j^k) < \mathcal{F}(\zeta_j^k)$ thus $\mathcal{P}optim_j^k \leftarrow \zeta_j^k$
(f) If $\mathcal{F}(\mathcal{G}optim_j^k) < \mathcal{F}(\zeta_j^k)$ thus $\mathcal{G}optim_j^k \leftarrow \zeta_j^k$

$k \leftarrow k + 1$:
end while
return $\mathcal{G}optim_j^k$ which is the best solution founded.

3.2 Proposed HPSO for the FS problem

This work proposes a new approach for solving FS named HPSO, which uses an asynchronously adaptive inertia weight and improved constriction factor, and also uses a chaotic map and a MAD fitness function with a feature count penalty to improve text clustering accuracy. To accomplish this goal, HPSO is used in each document to generate new subsets of useful text features, which are then combined to form the input for the *k*-mean clustering approach.

Before starting to expose our methodology of HPSO, we present the mathematical model for considered FS problem.

3.2.1 Mathematical model

Given VSM as vectors of weights of text features in each document, *VSM* is represented as a vector

$$\mathrm{VSM}_i = \phi_{i,1}, \phi_{i,2}, \ldots, \phi_{i,j}, \ldots, \phi_{i,t-1}, \phi_{i,t},$$

where i is the document number and t is the number of all unique terms. The FS algorithm generates a new subset of text features S, represented as a vector

$$S_i = s_{i,1}, \ldots, s_{i,j}, \ldots, s_{i,t}, \quad s_{i,j} \in \{0, 1\}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, t,$$

where n represents the total number of documents. If $s_{i,j} = 1$, the feature number j in the ith document has been chosen as informative feature. If $s_{i,j} = 0$, it implies that the feature number j in the ith document is an non-informative text feature. It is possible to formulate the text FS problem as an optimization problem to identify the ideal subset of practical features as follows:

$$\begin{aligned} &Max && \mathcal{F}_i := \mathcal{MAD}_i \\ &s.t. && s_{i,j} \in \{0, 1\} \\ &&& \forall\ i = 1, \ldots, n, \quad \forall\ j = 1, \ldots, t \end{aligned} \tag{5}$$

where ($\mathcal{MAD}$) is the mean absolute difference, which is used as an objective function for the text FS problem [1]. It involves applying the common weighting formula to analyze the measure that the PSO algorithm uses to assess each solution it offers in each generation (TF-IDF). The solutions with the highest $\mathcal{MAD}$ value provided by the PSO in each document are considered to be the optimal solutions to the FS problem. Calculating mean value first, followed by the absolute value of the difference between value and mean value of the chosen feature weights $\phi_{i,j}$, is how $\mathcal{MAD}$ assigns a score (fitness value) to each candidate solution, as illustrated below:

$$\mathcal{MAD}_i = \frac{1}{a_i}\sum_{j=1}^{t} s_{i,j}\left|\phi_j - \bar{\zeta_i}\right|$$

where (6)

$$\bar{\zeta_i} = \frac{1}{a_i}\sum_{j=1}^{t} s_{i,j}\phi_j$$

$\mathcal{MAD}_i$ represents the value of the fitness for the ith particle. If the jth term is selected in the ith solution, $s_{i,j} = 1$; otherwise, $s_{i,j} = 0$. The ϕ_j stands for the weight of the jth feature in the current document, a_i for the total number of selected features in the ith particle for the current document, t for the total number of terms, and $\bar{\zeta_i}$ for the average value of the selected weights in the VSM for the current document.

To resolve the FS problem, we applied PSO approach. The main goal is to develop a new subset of informative text features that will serve as the best possible replacement for the existing document. The process starts with random vector of features and then enhances the population until the stop criterion is reached. The PSO swarm is made up of particles (solutions), each of which is represented by a binary vector of positions (features) [2]. Each placement denotes the state of a single feature in the document. The solution (particle) representation of the PSO algorithm is given by

$$X = [0, 1, 1, -1, 0, 1, 1, -1, 0, 1], \quad (7)$$

where each unique term in the t search area for the FS problem has the option of being chosen or ignored. When position j is 1, it indicates the jth feature is chosen as a valuable text feature; when position j is 0, it indicates the jth feature is not chosen; and when position j is -1, it indicates the jth feature is absent from the real document.

3.2.2 Penalty fitness evaluation

Denote by d the size of a given feature subset, which is used as a constraint to force the generated subset of text features that satisfy the given size requirement [2], and thus particles that violate this constraint are penalized. The particle fitness is represented in the following equation.

$$\mathcal{F}_i = \mathcal{MAD}_i - \delta \times |a_i - d| \quad (8)$$

where δ is a penalty coefficient, d is the required size value, and a_i means the total amount of features selected in the particle ith.

3.2.3 Chaotic map

Chaos refers to some random irregular motions appearing in deterministic systems. It is a nonlinear dynamic system that is highly sensitive to its initial circumstances and parameters. It possesses the traits of determinism, ergodicity, stochasticity, and regularity. Several researchers have employed chaotic maps to improve PSO's search and global convergence capabilities [48]. In this study, we use a logistic map to generate chaotic sequences [2]. Mathematically, the logistic map is defined as

$$ch_{I+1} = 4 \times ch_I \times (1 - ch_I) \quad (9)$$

ch generates a chaotic value between 0.0 and 1.0 at each iteration I. ch_0 is generated randomly, with ch_0 not equal to 0, 0.25, 0.5, 0.75, or 1. Figure 1 represents the changing curve of the ch value.

3.2.4 Constriction factor

The original PSO has been improved and extensively used in many applications. Nevertheless, no rigorous mathematical justification for the convergence of the PSO algorithm has been established. Thus, Clerc [49] has explained, through some mathematical tools, how a simplified PSO model behaves in its seek for an optimal solution with the conclusion it may not converge in some situations. Indeed, the system equations of PSO (3–4) can be seen as system dynamics. In order to assure convergence and prevent premature convergence, constriction factors have been developed as a result of the analysis of the system trajectory [50]. Due to the large dimensions of the text feature set, the classical PSO will stick in a local optimum where the global optimum has not yet been found [35]. Thus, this work incorporated the constriction factor K into PSO to achieve the best convergence. The velocity and constriction factor based on the cosine function is shown in Eq. (10). In the early iterations, a convex function with a large K value is chosen, and a concave function with a

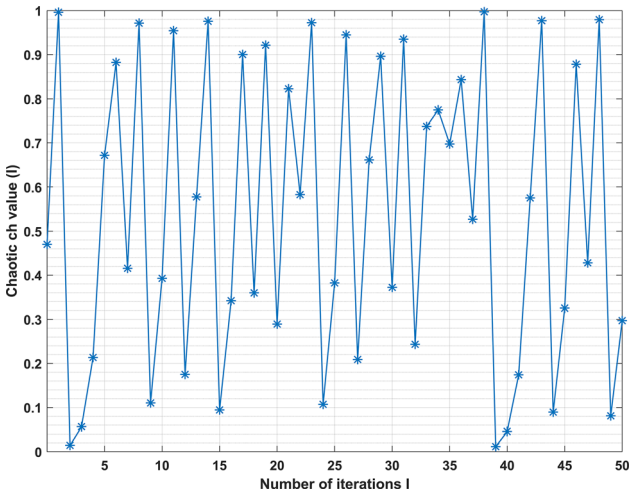

Fig. 1 The changing curve of value ch

smaller K value is chosen in the late period so that a particle in PSO will search over vast area to find the placement of the best solution. Then, in a small range, it will converge to determine the best solution.

$$\upsilon_{i,j} = K[\upsilon_{i,j} + \gamma_1 \times \xi_1 \times (\mathcal{P}optim_I - \zeta_{i,j}) + \gamma_2 \times \xi_2 \times (\mathcal{G}optim_I - \zeta_{i,j})] \quad (10)$$

where

$$K = 0.25\cos((\pi/I_{max}) \times I) + \frac{5}{8}$$

I is the number of iterations. Figure 2 represents the changing curve of the K value. The curve of value K in Fig. 2 begins as a convex function and eventually becomes a concave function.

3.2.5 Adaptive inertia weight

As can be seen from Eq. (10), there are three main components that keep track of the velocity update. The first component of the information refers to the particle's previous velocity, the second to the information the particle itself possesses, and the third to the information stored in the swarm. The inertia weight controls the current velocity, while γ_1, ξ_1 and γ_2, ξ_2 control the second and the third ones, respectively. These parameters are crucial for enhancing the PSO algorithm's ability to search. In this study, the PSO algorithm uses a fitness-based dynamic inertia weight to dynamically update velocity and change the inertia weight's value based on the particle's current fitness [2]. In order to search throughout a large portion of the search space (exploration), it allocates lower inertia weights to high fitness particles and larger inertia weights to low fitness particles (exploitation). The improved velocity equation and the fitness-based dynamic inertia weight are described as follows:

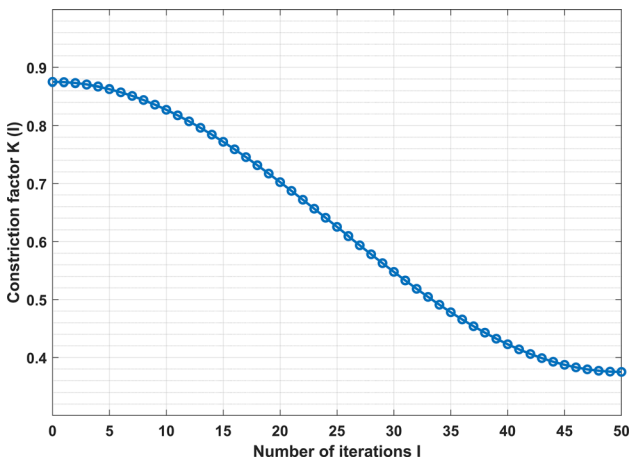

Fig. 2 The changing curve of value K

$$\upsilon_{i,j} = fw_i \times \upsilon_{i,j} + \gamma_1 \times \xi_1 \times (\mathcal{P}optim_I - \zeta_{i,j}) + \gamma_2 \times \xi_2 \times (\mathcal{G}optim_I - \zeta_{i,j}) \quad (11)$$

where

$$fw_i = 1.1 - \frac{0.9 \times fw_i}{fw_{\text{best}} + 0.1}$$

fw_i and fw_{best} represent the fitness of the ith and global best particles, respectively.

3.2.6 Asynchronously adaptive inertia weight and improved constriction factor

The constriction factor affects PSO particle convergence, while the inertia weight affects how much the initial velocity is maintained. In this study, inspired by the idea of [35] that tries to improve asynchronously inertia weight and constriction factor, we apply the constriction factor and the inertia weight adaptively throughout distinct PSO periods depending on their diverse features. During the initial phases, when the particle exhibits a high fitness value, the inertia weight is set to a smaller value. This allows the particle to retain a limited portion of its previous velocity, focusing on local development. As the particle moves farther away from the optimal solution, the inertia weight is increased, enabling the particle to maintain a larger portion of its previous velocity and explore the search space globally. Moreover, we introduce the chaotic strategy to enhance the global search capability of PSO. Indeed, the properties of a chaotic system provide a good exploration of the search space and refine the selected feature subspace. This strategy avoids the entrapment of an individual at an undesirable local minimum solution. Equation (12) shows the new formulas for inertia weight, constriction factor, and velocity.

$$\begin{cases} \upsilon_{i,j} = fw_i \times \upsilon_{i,j} + \gamma_1 \times ch \times (\mathcal{P}optim_I - \zeta_{i,j}) + \gamma_2 \times (1 - ch) \times (\mathcal{G}optim_I - zeta_{i,j}) & \text{if } I < \frac{I_{max}}{2} \\ \upsilon_{i,j} = K[0.7 \times \upsilon_{i,j} + \gamma_1 \times ch \times (\mathcal{P}optim_I - \zeta_{i,j}) + \gamma_2 \times (1 - ch) \times (\mathcal{G}optim_I - zeta_{i,j})] & \text{if } I \geq \frac{I_{max}}{2} \end{cases}$$

where

$$fw_i = 1.1 - \frac{0.9 \times fw_i}{fw_{\text{best}} + 0.1}$$

$$K = \frac{\cos\left((2\pi/I_{\max}) \times (I - \frac{I_{\max}}{2})\right) + 2.428571}{4} \quad (12)$$

ch value between 0.0 and 1.0 generated by Eq. (9) at each iteration I.

3.2.7 Hybrid particle swarm optimization algorithm

The FS problem is addressed by developing a new hybrid PSO method that identifies the best subset of text features. Sequence provides a detailed description of the created process, while Algorithm provides an algorithmic flow. 2.

Algorithm 2 HPSO

1: Initialize the particle swarm optimization parameters γ_1, γ_2, and etc.
2: Initialization
   (a) Initiate the swarm with a set of randomly generated particles.
   (b) Use the fitness function to evaluate all particles as shown in Eq. (6 and 8).
3: **while** Maximum iterations I_{max} not attained **do**
4: For the particle $j = 1, \ldots, \mathcal{N}p$, **do**
   (a) Update ch using Eq. (9).
   (b) Update constriction factor K and dynamic inertia weight fw as shown in Eq. (12).
   (c) Update the velocity using Eq. (12).
   (d) Update the positions of each particle.
   (e) Use the fitness function to examine all particles.
   (f) Update $\mathcal{P}optim$ and $\mathcal{G}optim$.
5: **end while**
6: **return** The best solution is founded (represented by a new subset of useful text features).

With a set of solutions that were produced at random, the PSO algorithm starts the swarm of particles. Each of them is assessed by the MAD fitness function using Eqs. (6 and 8) defined feature count penalty. The PSO swarm consists of several particles; each of them contains a number of positions (features) that move around with their own velocity using Eq. (12), causing the PSO algorithm to be positioned in the search space of the text FS problem. The fitness value of particles is evaluated in each iteration. In addition, parameters such as the chaotic map ch, constriction factor K, and fitness-based dynamic inertia weight fw_i are calculated in each iteration, and the current and best fitness is saved to affect particle movement in subsequent iterations. Finally, the best solution discovered by the PSO approach is chosen as the best solution, representing a new subset of revealing features.

3.3 Clustering

The accuracy of the FS techniques is assessed using the k-mean clustering approach after creating a fresh subset of informational text characteristics. The following subsections provide an explanation of the k-mean clustering method.

3.3.1 Mathematical model

Given a large set of text documents, $D = d_1, d_2, \ldots, d_i, \ldots, d_n$, where $d_i = w_{i1}, w_{i2}, \ldots, w_{ij}, \ldots,$ w_{it}, represents the document number i, w_{ij} represents the weight of the feature number j in the ith document, n is the number of documents in the given document collection, and t represents the total number of terms. The cost function $\mathcal{C}os(d_i, c_l)$ that evaluates the cosine similarity measure between the document number i and the cluster centroid number l, where $c_l = c_{l1}, c_{l2}, \ldots, c_{lj}, \ldots, c_{lt}$ must be updated in every iteration using Eq. (13). An extensive collection of text documents is clustered into k clusters using the considered objective function. By assigning documents to the cluster with the highest degree of similarity based on their resemblance to the cluster centroid, documents in these clusters are more similar to one another than documents in other groups [1].

$$c_l = \frac{\sum_{i=1}^{n}(a_{li})d_i}{\sum_{i=1}^{n} a_{li}} \tag{13}$$

where d_i represents the ith document. a_{li} is equal to 1 if the document number i is assigned to the lth cluster, and 0 otherwise. The cosine measure is employed in this study to compute the similarity value between the document vector

and the cluster centroid vector. The similarity value is calculated using the following formula:

$$Cos(d_i, c_l) = \frac{\sum_{j=1}^{t} w_{ij} \times c_{lj}}{\sqrt{\sum_{j=1}^{t} w_{ij}^2}\sqrt{\sum_{j=1}^{t} c_{lj}^2}} \quad (14)$$

where w_{ij} represents the magnitude of feature j in the ith document, c_{lj} is the value of the jth term in cluster centroid number l, $\sum_{j=1}^{t} w_{ij}^2$ is the square of the norm of the score vector for the document number i, and $\sum_{j=1}^{t} c_{lj}^2$ is the square of score vector for the cluster centroid number l.

3.3.2 *k*-means algorithm

The k-means method is an unsupervised one that attempts to process a large amount of text and organize it into a specified number of clusters, with each group containing documents that are similar to one another. k-means solves the clustering problem by using Eq. (14) to iteratively reassign text documents to clusters based on their similarity to the cluster centroid. Each iteration of the reassignment method will result in the recalculation of the cluster centroids using Eq. (13). In this method, $\Lambda(n \times k)$ represents the number of documents and clusters, where n represents the number of documents. This procedure is described by Algorithm 3. The k-mean clustering method seeks the best clustering solution $(n \times k)$.

Algorithm 3 k-means text clustering algorithm

1: Choose randomly k documents as the centroids of the initial clusters.
2: **while** Maximum of iterations not attained **do**
3: Initialize matrix Λ as zero.
4: **for** All d_i in D **do**
5: Compute
(a) Let $j = argmax_{l \in \{1..k\}} Cos(d_i, c_l)$.
(b) Assign d_i to the cluster j (i.e. $\Lambda[i][j] = 1$).
6: **end for**
7: Update the clusters by the Eq. (13).
8: **end while**
9: **return** $\Lambda(n \times k)$ matrix of n documents assigned to k cluster.

3.4 Complexity analysis

In this subsection, we present a comprehensive computational complexity analysis of the proposed HPSO used for feature selection in text data. Evaluating the efficiency and scalability of optimization algorithms is crucial for understanding their performance characteristics. The computational complexity of our HPSO is assessed by considering key operations involved in the optimization process. Specifically, we analyze the complexities associated with objective function evaluations, the generation of chaotic sequences using logistic maps, velocity updates, and the adaptive adjustment of inertia weights and constriction factors. The overall computational complexity is derived by aggregating these individual complexities over the course of HPSO iterations. Let us break down the complexities associated with each key algorithmic component:

3.4.1 Fitness function evaluation complexity

The objective function, representing the MAD fitness, involves the calculation of fitness values for each particle. Considering $\mathcal{N}p$ particles and t features, the complexity (C_{MAD}) can be expressed as:

$$C_{\text{MAD}} = O(\mathcal{N}p \cdot t)$$

3.4.2 Chaotic sequence generation complexity

Logistic map equation (9) for generating chaotic sequences involves iterative calculations. With $I_{\max}$ iterations, the complexity (C_{chaotic}) is given by:

$$C_{\text{chaotic}} = O(I_{\max})$$

3.4.3 Velocity updates complexity

Velocity update Equation (10) includes arithmetic operations and trigonometric functions. Considering $I_{\max}$ iterations and $\mathcal{N}p \cdot t$ features, the complexity (C_{velocity}) is expressed as:

$$C_{\text{velocity}} = O(I_{\max} \cdot \mathcal{N}p \cdot t)$$

3.4.4 Inertia weight and constriction factor adaptation complexity

The adaptation of inertia weight and constriction factor involves conditional statements and arithmetic operations. With $I_{\max}$ iterations and $\mathcal{N}p$ particles, the complexity ($C_{\text{adaptation}}$) is given by:

$$C_{\text{adaptation}} = O(I_{\max} \cdot \mathcal{N}p)$$

By summing up these complexities, the overall computational complexity (C_{total}) of the proposed PSO variant can be expressed as:

$$C_{\text{total}} = C_{\text{MAD}} + C_{\text{chaotic}} + C_{\text{velocity}} + C_{\text{adaptation}}$$

This analysis provides valuable insights into the algorithm's resource requirements and scalability, offering a foundation for discussions on optimization efficiency in the context of large-scale text feature sets.

The classical PSO complexity is given by $O(\mathcal{N}p \cdot t + I_{\max} \cdot \mathcal{N}p \cdot t + I_{\max} \cdot \mathcal{N}p)$. Comparing both complexities, the proposed HPSO introduces an additional complexity term associated with chaotic sequence generation. However, it is important to note that the chaotic sequence generation complexity ($O(I_{\max})$) is generally lower than the velocity updates complexity ($O(I_{\max} \cdot \mathcal{N}p \cdot t)$) in both algorithms.

In summary, while the proposed HPSO introduces some additional computational load due to chaotic sequence generation, the overall impact on complexity is moderate.

4 Experiments results

We implement a Java software that uses the HPSO algorithm to choose a fresh set of practical text features before using the *k*-means text clustering method. This section includes the data sets, parameter settings, evaluation criteria, and results. All tests are performed on a Laptop with a core i7 processor and 16GB of RAM in a Windows 10 environment (Table 1).

4.1 Data sets

The experiment is carried out on two different reference text data sets, Reuters-21578 and Webkb. Tables 2, 3, and 4 represent a summary of the data sets used to compare HPSO with other competitive methods (GA [9], and other well-known PSO variants such as PSO with fixed inertia weight, PSO with improved inertia weight, PSO with fixed constriction factor, PSO with improved constriction factor, and PSO with adaptive inertia weight). One of the most popular document clusters for text categorization research, the Reuters-21578 data set includes a variety of historical data from the Reuter news agency. Reuters, Ltd. and Carnegie Group, Inc. collected and initialized the data. The R8 set has 7674 documents overall and extracts from Reuters-21578 all papers with the eight most common subjects, which are acq, crude, earn, grain, interest, money-fx, ship, and trade. A thorough distribution of these documents is presented in Table 2.

Table 3 shows the detailed distribution of the R52 set, which extracts a large number of documents from Reuters-21578 with 52 subjects and a total of 9100 documents. The R8 and R52 data sets are created by applying the following transformations to the original Reuters-21578:

- SPACE is used in place of the characters TAB, NEWLINE, and RETURN. Only letters should be kept (i.e., turn punctuation, numbers, etc. to SPACES). Lowercase all letters. Multiple SPACES should be replaced with a single SPACE. Titles and subjects are simply added to documents.
- Words with fewer than three characters are removed. For instance, remove 'he' but keep 'him.'
- 524 stop words should be removed. As a result of being shorter than three characters, many of them have already been eliminated.
- Applying Stemmer Porter to the remaining words.

Webkb extracts documents from the web based on four popular topics: Project, Course, Faculty, and Students, with a total of 4199 documents, as shown in Table 4. The data sets documents are described as follows: they are all text files, one document for each row, each document is represented by a "word" representing the document's class, a TAB character, and a series of "words" separated by spaces, which represent the terms contained in the document. Each document is constituted of its group and its terms.

4.2 Parameter settings

In this paper, seven meta-heuristic algorithms are compared: PSO with fixed inertia weight (FIWPSO) [35], PSO with improved inertia weight (IIWPSO) [35], PSO with fixed constriction factor (FCFPSO) [35], PSO with improved constriction factor (ICFPSO) [35], PSO with adaptive inertia weight (AIWPSO) [2], genetic algorithm (GA) [9], and the proposed HPSO method. The algorithms under consideration make use of a different changeable parameters. The parameters for the competing approaches were derived from relevant papers that the researchers

Table 1 Parameters setting used in this paper

Parameter	FIWPSO	IIWPSO	FCFPSO	ICFPSO	AIWPSO	GA	HPSO
Population size	10	10	10	10	10	10	10
c1 and c2	2.05	2.05	2.05	2.05	2.05	NA	2.05
w	0.58	Improved	NA	NA	Dynamic	NA	Dynamic
k	NA	NA	0.7298	Improved	NA	NA	Improved
# of clusters	Number of subjects in the data set						
Imax	50	50	50	50	50	50	50
Imax *k*-means	20	20	20	20	20	20	20

Table 2 Detailed distribution of the R8 data set

R8			
Class	# of Train docs	# of Test docs	Total # of docs
acq	1596	696	2292
crude	253	121	374
earn	2840	1083	3923
grain	41	10	51
interest	190	81	271
money-fx	206	87	293
ship	108	36	144
trade	251	75	326
Total	**5485**	**2189**	**7674**

The bold highlights the high metric

Table 3 Detailed distribution of the R52 data set

R52			
Class	# of Train docs	# of Test docs	Total # of docs
⋮	⋮	⋮	⋮
tin	17	10	27
trade	251	75	326
veg-oil	19	11	30
wpi	14	9	23
zinc	8	5	13
Total	**6532**	**2568**	**9100**

The bold highlights the high metric

suggested based on their experimental study. The values of the parameters used in the paper are shown in Table 1.

4.3 Evaluation criteria

Comparative evaluations were performed using one internal evaluation measure (the similarity measure) and four external evaluation measures (the precision (P), recall (R), F-measure, and average accuracy). The variables taken into consideration are generally accepted evaluation standards for assessing cluster correctness in the context of text clustering. We must count the number of documents with the same topic that are in the same cluster as well as the number of documents with different topics that are in distinct clusters to examine the performance of clustering. One of the aforementioned conditions could be true for every set of documents:

Table 4 Detailed distribution of the Webkb data set

Webkb			
Class	# of Train docs	# of Test docs	Total # of docs
project	336	168	504
course	620	310	930
faculty	750	374	1124
students	1097	544	1641
Total	**2803**	**1396**	**4199**

The bold highlights the high metric

- *SS* Both documents are grouped together in both our clusters and the corpus.
- *SD* Although the two documents are in separate clusters in the corpus, they are in the same cluster in our clusters.
- *DS*In our clusters, documents are divided into distinct groups; yet, in the corpus, they are put together in the same groups.
- *DD* Both documents were categorized in separate clusters in both the corpus and our clusters.

The accuracy rate is given by Eq. (15), where α, β, η, and ρ are the amount of document couplings in the SS state, SD, DS, and DD, respectively.

$$\text{Average Accuracy} = \frac{1}{2}\left(\frac{\alpha}{\alpha+\eta}+\frac{\rho}{\beta+\rho}\right) \tag{15}$$

Utilizing the F-measure, which has the following formula, is an additional strategy for evaluating clustering

$$\text{F-measure} = \frac{2 \times P \times R}{P + R}$$

where

$$P = \frac{\alpha}{\alpha + \beta} \qquad (16)$$

$$R = \frac{\alpha}{\alpha + \eta}$$

Average accuracy, precision, recall, and *F*-measure can all go as high as 1. This value may appear if all documents are successfully grouped.

4.4 Results and discussion

Three experiments are realized to show the efficiency of the HPSO and to discuss the performance of each PSO variant. In the first experiment, the inertia weight variants of PSO named FIWPSO, IIWPSO, and AIWPSO are compared (Tables 5, 6). The second compares fixed inertia weight (FIWPSO) to the constriction variants of PSO known as FCFPSO and ICFPSO in order to illustrate the effectiveness of the constriction factor in the particle swarm optimization algorithm. Finally, we compare HPSO results to all PSO variants and GA results. All algorithms are executed independently twenty times on all of the six data sets. These algorithms are compared in terms of accuracy, precision, recall, *F*-measure, and convergence rate. The convergence rate assesses how quickly the algorithm approaches the optimal solution over iterations. The comparison of convergence rates is presented in Fig. 3.

Based on the *k*-means text clustering algorithm, Table 7 displays the Algorithms effectiveness (Accuracy, Fmeasure, Precision, and Recall). The FS technique using the AIWPSO algorithm and the IIWPSO outperformed the other comparable inertia weight variant of PSO FIWPSO in terms of outcomes. According to two of the evaluation metrics (Accuracy and Precision), the AIWPSO performed best in three of the six data sets, followed by the IIWPSO in two of the six data sets. IIWPSO performed best in three of the six data sets and five of the six data sets, respectively, as measured by Fmeasure and Recall. In the second experiment, the FCFPSO method clearly outperformed the other comparative Constriction-based method ICFPSO and the conventional Fixed Inertia Weight FIWPSO algorithm for almost all data sets in terms of clustering evaluation criteria. Finally, for the third experiment, based on the evaluation criteria, the developed HPSO recorded the best effectiveness and outperformed the other comparative methods (FIWPSO, IIWPSO, AIWPSO, FCFPSO, ICFPSO, and GA), followed by the IIWPSO. According to Table 7, HPSO achieved the outstanding results in three out of the six data sets (i.e., WebkbTest, WebkbTrain, and R8Train), based on the Accuracy measure, followed by IIWPSO in two out of six data sets (i.e., R52Test and R52Train). Based on Fmeasure, HPSO and IIWPSO achieved comparable results, with HPSO get the distinguished results in three of six data sets (i.e., WebkbTest, WebkbTrain, and R8Train) and IIWPSO get the most good results in three of six data sets (i.e., R8Test, R52Test, and R52Train). Based on Precision metric, HPSO outperformed the other algorithms in four out of six data sets (i.e., WebkbTest, WebkbTrain, R8Train, and R52Train) followed by both IIWPSO and AIWPSO in one out of six data sets. Based on Recall measure, IIWPSO performed best in three of the six data sets (i.e., R8Test, R52Test, and R52Train) followed by HPSO, FCFPSO, and FIWPSO, who each had the most good results in one of the six data sets.

Table 8 displays the algorithm's mean accuracy, Fmeasure, precision, and recall in six data sets. In terms of performance, HPSO-based FS approaches clearly outperform more traditional FS methods. To begin, HPSO has one of the best Recall results and the highest mean in the Accuracy, Fmeasure, and Precision measures. Meanwhile, IIWPSO and AIWPSO have the best results after HPSO in most of the measures. According to Table 8, AIWPSO had the second best mean based on the Accuracy and Precision measures, with IIWPSO coming in third. According to the Fmeasure metric, IIWPSO obtained the second best mean followed by AIWPSO in the third place. Finally, in terms of recall, IIWPSO outperformed the other algorithms and achieved the highest mean.

The new developed FS method HPSO is used to generate new subsets of useful text features in order to improve the accuracy of the *k*-mean text clustering algorithm. To summarize, the proposed HPSO-based FS approaches can achieve the best performance of text clustering according to the majority of evaluation measures compared to similar variants of PSO algorithms and the GA. In addition, the HPSO has the best mean across almost all measures and is ranked first. The degree to which particles maintain their original velocity is measured by inertia weight. Thus, the inertia weight adjustment is critical for adjusting particle velocity so that they can escape from the local optimal solution and reach a more good solution via an effective search strategy. Tables 7 and 8 clearly illustrate this analysis. The results show that IIWPSO and AIWPSO achieve better mean results in almost all measures than the competitive methods and are ranked 2 and 3, respectively. The FCFPSO was ranked fourth, outperforming the FIWPSO and ICFPSO. The *k*-mean clustering with genetic-based FS algorithm was the worst, ranking seventh.

In order to demonstrate the convergence characteristics of the competing approaches, we recorded the convergence

Table 5 Algorithms effectiveness (Accuracy, Fmeasure, Precision, and Recall) in six data sets over ten runs based on *k*-means text clustering algorithm

Data set	Number of unique features	Method	FIWPSO	IIWPSO	AIWPSO	FCFPSO	ICFPSO	GA	HPSO
WebkbTest	4798	Accuracy	0.587	0.582	0.589	0.582	0.582	0.588	**0.610**
		Fmeasure	0.428	0.427	0.430	0.432	0.425	0.430	**0.460**
		Precision	0.387	0.375	0.389	0.370	0.376	0.387	**0.427**
		Recall	0.483	0.498	0.485	**0.523**	0.494	0.489	0.508
		Mean rank	4.87	5.25	3.37	4.00	5.50	3.75	1.25
		Rank	5	6	2	4	7	3	**1**
WebkbTrain	7287	Accuracy	0.616	0.598	0.611	0.616	0.621	0.628	**0.653**
		Fmeasure	0.478	0.464	0.472	0.476	0.484	0.492	**0.516**
		Precision	0.400	0.373	0.393	0.399	0.403	0.411	**0.447**
		Recall	0.602	0.624	0.597	0.596	0.618	0.615	**0.633**
		Mean rank	4.37	5.75	6.00	5.37	3.00	2.50	1.00
		Rank	4	6	7	5	3	2	**1**
R8Test	8575	Accuracy	0.575	0.594	**0.595**	0.575	0.581	0.567	0.590
		Fmeasure	0.362	**0.399**	0.397	0.368	0.367	0.344	0.386
		Precision	0.552	0.587	**0.594**	0.540	0.579	0.542	0.586
		Recall	0.272	**0.305**	0.300	0.281	0.269	0.253	0.289
		Mean rank	5.37	1.50	1.50	5.12	4.75	6.75	3.00
		Rank	6	**1**	**1**	5	4	7	3
R52Test	9730	Accuracy	0.538	**0.545**	0.539	0.541	0.535	0.524	0.531
		Fmeasure	0.167	**0.193**	0.170	0.179	0.161	0.120	0.144
		Precision	0.609	**0.618**	0.616	0.617	0.585	0.544	0.591
		Recall	0.097	**0.114**	0.099	0.104	0.093	0.067	0.082
		Mean rank	4.00	1.00	3.00	2.00	5.25	7.00	5.75
		Rank	4	**1**	3	2	5	7	6
R8Train	14575	Accuracy	0.574	0.569	0.574	0.572	0.568	0.569	**0.597**
		Fmeasure	0.373	0.362	0.365	0.369	0.360	0.344	**0.391**
		Precision	0.531	0.527	0.548	0.544	0.526	0.550	**0.644**
		Recall	**0.292**	0.279	0.275	0.285	0.275	0.251	0.281
		Mean rank	2.62	5.12	3.75	3.25	6.37	5.37	1.50
		Rank	2	5	4	3	7	6	**1**
R52Train	16145	Accuracy	0.536	**0.540**	0.536	0.539	0.533	0.522	0.530
		Fmeasure	0.160	**0.173**	0.161	0.169	0.152	0.112	0.133
		Precision	0.610	0.618	0.611	0.615	0.596	0.533	**0.656**
		Recall	0.092	**0.101**	0.093	0.098	0.087	0.062	0.074
		Mean rank	4.12	1.25	3.37	2.25	5.25	7.00	4.75
		Rank	4	**1**	3	2	6	7	5
Mean rank			4.23	3.31	3.50	3.67	5.02	5.40	2.87
Final rank			5	2	3	4	6	7	**1**

The bold highlights the high metric

values for each program obtained through ten runs, each consisting of 50 iterations. The fitness function MAD values versus the number of iterations are used in the comparative analysis of the techniques. Figure 3 illustrates the changing convergence curves of FIWPSO, IIWPSO, AIWPSO, FCFPSO, ICFPSO, GA, and HPSO on WebkbTest, WebkbTrain, R8Test, R52Test, R8Train, and R52Train data sets.

As observed in Fig. 3, HPSO exhibits a slower convergence when contrasted with other methods that rely

Table 6 Algorithm's mean Accuracy, Fmeasure, Precision, and Recall in six data sets

Method	FIWPSO	IIWPSO	AIWPSO	FCFPSO	ICFPSO	GA	HPSO
Accuracy	0.571	0.571	0.574	0.571	0.570	0.566	**0.585**
Fmeasure	0.328	0.336	0.332	0.332	0.325	0.307	**0.338**
Precision	0.515	0.516	0.525	0.514	0.511	0.495	**0.559**
Recall	0.306	**0.320**	0.308	0.315	0.306	0.290	0.311

The bold highlights the high metric

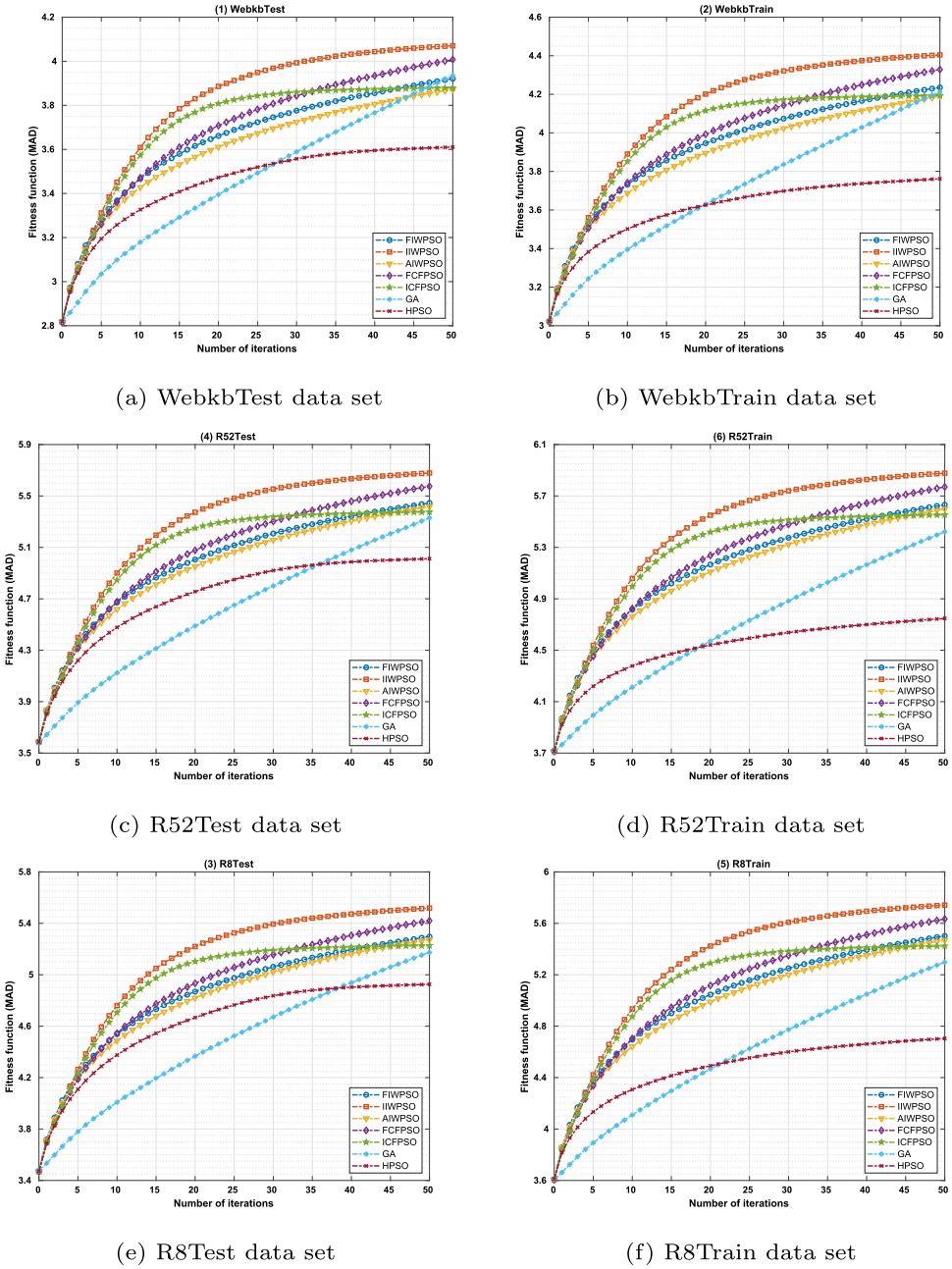

Fig. 3 The changing convergence curves of FIWPSO, IIWPSO, AIWPSO, FCFPSO, ICFPSO, GA, and HPSO on six data sets

(a) WebkbTest data set (b) WebkbTrain data set

(c) R52Test data set (d) R52Train data set

(e) R8Test data set (f) R8Train data set

solely on the MAD fitness function without incorporating a feature count penalty. This characteristic is attributed to the multi-objective nature introduced in the proposed HPSO. In this multi-objective approach, HPSO is designed to simultaneously optimize two critical objectives: the MAD fitness function and a feature count penalty.

Table 7 Algorithm effectiveness (Accuracy, Fmeasure, Precision, and Recall) in six data sets over ten runs based on *k*-means text clustering algorithm

Data set	Number of unique features	Method	IIWPSO	HPSO	SpadePSO
WebkbTest	4798	Accuracy	0.582	**0.610**	0.603
		Fmeasure	0.427	**0.460**	0.454
		Precision	0.375	**0.427**	0.397
		Recall	0.498	0.508	**0.529**
		Mean rank	3.00	1.25	1.75
		Rank	3	**1**	2
WebkbTrain	7287	Accuracy	0.598	**0.653**	0.622
		Fmeasure	0.464	**0.516**	0.479
		Precision	0.373	**0.447**	0.413
		Recall	0.624	**0.633**	0.578
		Mean rank	2.75	1.00	2.25
		Rank	3	**1**	2
R8Test	8575	Accuracy	**0.594**	0.590	0.573
		Fmeasure	**0.399**	0.386	0.349
		Precision	**0.587**	0.586	0.562
		Recall	**0.305**	0.289	0.254
		Mean rank	1.00	2.00	3.00
		Rank	**1**	2	3
R52Test	9730	Accuracy	**0.545**	0.531	0.531
		Fmeasure	**0.193**	0.144	0.142
		Precision	**0.618**	0.591	0.587
		Recall	**0.114**	0.082	0.081
		Mean rank	1.00	2.00	2.75
		Rank	**1**	2	3
R8Train	14575	Accuracy	0.569	**0.597**	0.572
		Fmeasure	0.362	**0.391**	0.373
		Precision	0.527	**0.644**	0.533
		Recall	0.279	0.281	**0.290**
		Mean rank	3.00	1.25	1.75
		Rank	3	**1**	2
R52Train	16145	Accuracy	**0.540**	0.530	0.529
		Fmeasure	**0.173**	0.133	0.135
		Precision	0.618	**0.656**	0.586
		Recall	**0.101**	0.074	0.077
		Mean rank	1.25	2.25	2.5
		Rank	**1**	2	2
Mean rank			2.00	1.625	2.33
Final rank			2	**1**	3

The bold highlights the high metric

The MAD fitness function serves to assess the quality of solutions by evaluating their capacity to represent pertinent text features. Conversely, the feature count penalty introduces a cost for larger feature subsets, aiming to strike a delicate balance between the necessity for informative features and the aspiration for a concise feature set. Consequently, the integration of the feature count penalty adds a layer of complexity to the optimization process, resulting in a slower convergence rate compared to methods exclusively optimizing the MAD fitness function. This

Table 8 Algorithm's mean Accuracy, Fmeasure, Precision, and Recall in six data sets

Method	IIWPSO	HPSO	SpadePSO
Accuracy	0.571	**0.585**	0.572
Fmeasure	0.336	**0.338**	0.322
Precision	0.516	**0.559**	0.513
Recall	**0.320**	0.311	0.302

The bold highlights the high metric

deceleration is attributed to the algorithm's additional consideration of penalizing larger feature subsets. While it may lead to a more gradual convergence, this nuanced approach proves beneficial in enhancing overall clustering accuracy, as the algorithm strategically balances the quality and quantity of the selected features.

Another observed result is that the IIWPSO converges faster than the competing methods and achieves the best results across nearly all six text data sets. AIWPSO has similar convergence curves to FIWPSO, but its final fitness is only slightly worse. ICFPSO has a lower fitness than FCFPSO but the fastest convergence speed, except for IIWPSO, requiring only 30 iterations to achieve the optimum. In comparison with all methods, FCFPSO has the second best fitness. The GA has the final worst fitness, when compared to all PSO variants, its fitness is slightly worse than ICFPSO, but its convergence speed is the slowest.

To assess the robustness of the proposed HPSO in feature selection clustering against more recent variants of PSO, especially those incorporating learning exemplars and structure topologies, we compare HPSO with SpadePSO [10]. SpadePSO, introduced as one of the latest variants of PSO, is distinguished by its innovative approach that includes advanced learning exemplars and sophisticated structure topologies. These features aim to enhance the algorithm's adaptability and exploration–exploitation balance. By choosing SpadePSO as a benchmark, we aim to provide a comprehensive evaluation, considering not only the historical PSO variants but also the cutting-edge developments in the field.

In this case, we have adopted the source code of SpadePSO [10], for solving the FS problem. In Tables 7 and 8 we present the performance comparison between SpadePSO, IIWPSO, and HPSO applied to the six data sets.

In the comparative evaluation of SpadePSO, HPSO, and IIWPSO across the six data sets, noteworthy patterns emerge from the performance metrics. HPSO consistently demonstrates robust results, outperforming both SpadePSO and IIWPSO in several key aspects. In terms of accuracy, precision, recall, and *F*-measure, HPSO excels, showcasing its superior ability to generate effective feature subsets for text clustering. IIWPSO, known for its fast convergence, remains competitive, demonstrating strengths in specific data sets. SpadePSO, while exhibiting respectable performance, generally falls behind HPSO, particularly in accuracy and precision. These results underscore the efficacy of HPSO in enhancing clustering outcomes, emphasizing the importance of its unique combination of asynchronously adaptive inertia weight, improved constriction factor, chaotic map, and MAD fitness function with a feature count penalty. The multi-objective nature of HPSO, though contributing to slower convergence, proves instrumental in achieving top-tier effectiveness, providing a valuable trade-off for practitioners seeking optimal clustering solutions.

As a summarized conclusion of the comparison results for the three experiments realized, we have:

1. Comparison of Inertia Weight Variants of PSO (FIWPSO, IIWPSO, AIWPSO):

   AIWPSO and IIWPSO outperformed FIWPSO in most data sets for accuracy, precision, recall, and *F*-measure.
2. Comparison of Constriction Factor Variants of PSO (FCFPSO, ICFPSO):

   FCFPSO consistently outperformed ICFPSO and the conventional FIWPSO in almost all data sets based on clustering evaluation criteria.
3. The comparison between HPSO and the most recent variant of PSO (SpadePSO):

   Comparison reveals intriguing insights into their performance across various data sets and evaluation metrics. In terms of accuracy, *F*-measure, precision, and recall, HPSO demonstrates a competitive edge over SpadePSO in the majority of data sets.
4. HPSO vs. All PSO Variants and GA:

   - HPSO demonstrated superior performance when compared to all PSO variants and GA.
   - HPSO achieved outstanding results in three of the six data sets based on accuracy (WebkbTest, WebkbTrain, R8Train), followed by IIWPSO in two data sets (R52Test and R52Train).
   - In terms of *F*-measure, HPSO and IIWPSO achieved comparable results, with HPSO excelling in three data sets (WebkbTest, WebkbTrain, R8Train), and IIWPSO performing best in three data sets (R8Test, R52Test, R52Train).
   - HPSO outperformed other algorithms in four data sets in terms of precision (WebkbTest, WebkbTrain, R8Train, R52Train).
   - IIWPSO performed best in three data sets in terms of recall (R8Test, R52Test, R52Train).

- In the evaluation across the six data sets (WebkbTest, WebkbTrain, R8Test, R52Test, R8Train, R52Train), HPSO consistently exhibits superior performance compared to SpadePSO, showcasing its efficacy in feature selection for text clustering across diverse data sets.

It follows that the mean performance across all measures, including accuracy, *F*-measure, precision, and recall, is consistently better for HPSO-based FS approaches compared to traditional methods. Moreover, HPSO ranks first in most measures, with IIWPSO and AIWPSO following, while the convergence analysis reveals that HPSO, although having slower convergence due to its multi-objective nature, achieves the best overall effectiveness. It turns out that IIWPSO stands out with fast convergence and top results in most data sets.

We concluded that the HPSO-based FS approach significantly enhances text clustering performance across multiple evaluation metrics when compared to various PSO variants and the GA. This improvement is attributed to the exploration and exploitation capability owing to asynchronously adaptive inertia weight, improved constriction factor, chaotic map, and MAD fitness function with a feature count penalty. These strategies helped the algorithm improve its search capability (exploration and exploitation); hence, the HPSO attains a better solution and avoids stagnation of the particles at a local optimal solution.

5 Conclusions and prospective directions

Text clustering offers an efficient means to automatically group digital documents based on their inherent characteristics. However, the high dimensionality of the feature space poses a significant challenge in text clustering. Various meta-heuristic techniques have been proposed in the literature to address the feature selection problem. In this paper, we analyze several variants of PSO algorithms and introduce a novel approach for feature selection, which we have named HPSO. Our aim is to overcome issues such as premature convergence and particle entrapment in local optima. To achieve this, we integrate different strategies into the PSO to enhance its search capabilities.

We work on four distinct stages of the PSO to improve its search efficiency. This includes the incorporation of an asynchronously adaptive inertia weight, an enhanced constriction factor, the use of chaotic maps, and the application of a MAD fitness function with a feature count penalty. Through numerical experiments, we have assessed the effectiveness of the developed method compared to other competitive methods. The results demonstrate that the proposed HPSO method not only achieves higher clustering precision but also selects a more informative feature set.

While the elementary steps introduced in HPSO incrementally increase the algorithm's computational complexity, they significantly enhance the PSO's search capabilities, improving both convergence behavior and accuracy. Statistical analysis validates that the HPSO outperforms competitive methods.

Looking ahead, we plan to address specific issues identified with HPSO in this study. We will explore other PSO variants known for their improved search capabilities and faster convergence to apply them to the feature selection task. We also aim to develop methods for the automatic adjustment of certain parameters, given their influence on PSO performance, while maintaining reasonable computational complexity.

Data availability The data that support the findings of this study are openly available in [7, 8].

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. 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
2. Bharti KK, Singh PK (2016) Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering. Appl Soft Comput 43:20–34
3. Lin S-W, Ying K-C, Chen S-C, Lee Z-J (2008) Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst Appl 35(4):1817–1824
4. Shamsinejadbabki P, Saraee M (2012) A new unsupervised feature selection method for text clustering based on genetic algorithms. J Intell Inf Syst 38(3):669–684
5. Lu H, Wang X, Fei Z, Qiu M (2014) The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms. Math Probl Eng
6. Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY (2002) An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans Pattern Anal Mach Intell 24(7):881–892
7. reuters21578, http://kdd.ics.uci.edu/databases/reuters21578
8. Webkb, http://www.cs.cmu.edu/~webkb/
9. Abualigah LM, Khader AT, Al-Betar MA (2016) Unsupervised feature selection technique based on genetic algorithm for improving the text clustering. In: 7th international conference on computer science and information technology (CSIT). IEEE, pp 1–6
10. Wu X, Han J, Wang D, Gao P, Cui Q, Chen L, Liang Y, Huang H, Lee HP, Miao C et al (2023) Incorporating surprisingly

popular algorithm and Euclidean distance-based adaptive topology into PSO. Swarm Evolut Comput 76:101222
11. Crone SF, Kourentzes N (2010) Feature selection for time series prediction-a combined filter and wrapper approach for neural networks. Neurocomputing 73(10–12):1923–1936
12. Khammassi C, Krichen S (2020) A nsga2-lr wrapper approach for feature selection in network intrusion detection. Comput Netw 172:107183. https://doi.org/10.1016/j.comnet.2020.107183
13. Hu G, Du B, Wang X, Wei G (2022) An enhanced black widow optimization algorithm for feature selection. Knowl Based Syst 235:107638
14. Jha K, Saha S (2021) Incorporation of multimodal multiobjective optimization in designing a filter based feature selection technique. Appl Soft Comput 98:106823. https://doi.org/10.1016/j.asoc.2020.106823
15. Xue B, Zhang M, Browne WN (2015) A comprehensive comparison on evolutionary feature selection approaches to classification. Int J Comput Intell Appl 14(02):1550008
16. Das S (2001) Filters, wrappers and a boosting-based hybrid for feature selection. In: Icml, Vol. 1, Citeseer, pp. 74–81
17. Liu X-Y, Liang Y, Wang S, Yang Z-Y, Ye H-S (2018) A hybrid genetic algorithm with wrapper-embedded approaches for feature selection. IEEE Access 6:22863–22874
18. Chhikara RR, Sharma P, Singh L (2016) A hybrid feature selection approach based on improved PSO and filter approaches for image steganalysis. Int J Mach Learn Cybern 7(6):1195–1206
19. Kılıç F, Kaya Y, Yildirim S (2021) A novel multi population based particle swarm optimization for feature selection. Knowl Based Syst 219:106894
20. Xue B, Zhang M, Browne WN (2012) Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans Cybern 43(6):1656–1671
21. Das AK, Das S, Ghosh A (2017) Ensemble feature selection using bi-objective genetic algorithm. Knowl Based Syst 123:116–127
22. Shastry KA, Sanjay H (2021) A modified genetic algorithm and weighted principal component analysis based feature selection and extraction strategy in agriculture. Knowl Based Syst 232:107460
23. Yang J, Honavar V (1998) Feature subset selection using a genetic algorithm. In: Feature extraction, construction and selection, Springer, pp 117–136
24. Kabir MM, Shahjahan M, Murase K (2012) A new hybrid ant colony optimization algorithm for feature selection. Expert Syst Appl 39(3):3747–3763
25. Wan Y, Wang M, Ye Z, Lai X (2016) A feature selection method based on modified binary coded ant colony optimization algorithm. Appl Soft Comput 49:248–258
26. Al-Ani A, Alsukker A, Khushaba RN (2013) Feature subset selection using differential evolution and a wheel based search strategy. Swarm Evol Comput 9:15–26
27. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the sixth international symposium on micro machine and human science, IEEE, pp 39–43
28. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). IEEE, pp 69–73
29. Sheikhpour R, Sarram MA, Sheikhpour R (2016) Particle swarm optimization for bandwidth determination and feature selection of kernel density estimation based classifiers in diagnosis of breast cancer. Appl Soft Comput 40:113–131
30. Naeini AA, Babadi M, Mirzadeh SMJ, Amini S (2018) Particle swarm optimization for object-based feature selection of vhsr satellite images. IEEE Geosci Remote Sens Lett 15(3):379–383
31. Sujana JAJ, Revathi T, Priya TS, Muneeswaran K (2019) Smart PSO-based secured scheduling approaches for scientific workflows in cloud computing. Soft Comput 23(5):1745–1765
32. Chaudhry R, Tapaswi S, Kumar N (2019) Fz enabled multi-objective PSO for multicasting in IoT based wireless sensor networks. Inf Sci 498:1–20
33. Liu Y, Wang G, Chen H, Dong H, Zhu X, Wang S (2011) An improved particle swarm optimization for feature selection. J Bionic Eng 8(2):191–200
34. Wang D, Tan D, Liu L (2018) Particle swarm optimization algorithm: an overview. Soft Comput 22(2):387–408
35. Lu Y, Liang M, Ye Z, Cao L (2015) Improved particle swarm optimization algorithm and its application in text feature selection. Appl Soft Comput 35:629–636
36. Wang D, Wang J, Wang H, Zhang R, Guo Z Intelligent optimization methods, China: Higher Education Press
37. Shi Y, Eberhart RC (1999) Empirical study of particle swarm optimization, in: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), Vol 3, IEEE, pp 1945–1950
38. Lindfield G, Penny J (2017) Chapter 3—particle swarm optimization algorithms. In: Lindfield G, Penny J (Eds.), Introduction to nature-inspired optimization, Academic Press, Boston, pp 49–68. https://doi.org/10.1016/B978-0-12-803636-5.00003-7
39. Wei J, Yuehong S, Xinning S (2010) A document clustering algorithm using particle swarm optimization. J Chin Soc Sci Tech Inf 29(3):428–432
40. Molaei S, Moazen H, Najjar-Ghabel S, Farzinvash L (2021) Particle swarm optimization with an enhanced learning strategy and crossover operator. Knowl Based Syst 215:106768
41. Nachaoui M, Afraites L, Laghrib A (2021) A regularization by denoising super-resolution method based on genetic algorithms. Signal Process Image Commun 99:116505. https://doi.org/10.1016/j.image.2021.116505
42. Nachaoui M, Chakib A, Nachaoui A (2020) An efficient evolutionary algorithm for a shape optimization problem. Appl Comput Math 19(2):220–244
43. Ghamisi P, Benediktsson JA (2014) Feature selection based on hybridization of genetic algorithm and particle swarm optimization. IEEE Geosci Remote Sens Lett 12(2):309–313
44. Zhang Y, Gong D, Hu Y, Zhang W (2015) Feature selection algorithm based on bare bones particle swarm optimization. Neurocomputing 148:150–157
45. Ileberi E, Sun Y, Wang Z (2022) A machine learning based credit card fraud detection using the GA algorithm for feature selection. J Big Data 9(1):1–17
46. Altarabichi MG, Nowaczyk S, Pashami S, Sheikholharam Mashhad P (2023) Fast genetic algorithm for feature selection-a qualitative approximation approach. In: Proceedings of the companion conference on genetic and evolutionary computation, pp 11–12
47. Hull DA (1996) Stemming algorithms: a case study for detailed evaluation. J Am Soc Inf Sci 47(1):70–84
48. Chuang L-Y, Yang C-H, Li J-C (2011) Chaotic maps based on binary particle swarm optimization for feature selection. Appl Soft Comput 11(1):239–248
49. Clerc M (1999) The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), Vol. 3, IEEE, pp 1951–1957

50. Naik BB, Raju CP, Rao RS (2018) A constriction factor based particle swarm optimization for congestion management in transmission systems. Int J Electr Eng Inf 10(2):232–241