An Investigation of the Use of Innovative Biology-Based Computational Intelligence in Ubiquitous Robotics Systems: Data Mining Perspective

Bo Xing

Abstract Sensor technologies are crucial in ambient assisted living system because they can observe, measure, and detect users' daily activities, in the meanwhile issue warnings when parameters exceed particular thresholds. Despite their enormous supporting potential, the quantities of data generated from multiple sensor is huge (i.e., big data), because they involved everywhere, such as smart home, health-based wearable devices, and assistive robots. Generally speaking, big data can be defined as large pools of data which comes from digital pictures, videos, intelligent sensors, posts to social media sites, purchase transaction records, cell phone global positioning system signals, to name a few. During the past few years, there is a great interest both in the commercial and in the research communities around big data. Under these circumstances, data mining, whose main purpose is to extract value from mountains of datasets, is drawing a lot of people's attention. To follow this trend, in this article, we intend to take an algorithmic point of view, i.e., applying intelligent algorithms to data, with an emphasis on the biology-based innovative computational intelligence (CI) methods. This work makes several contributions. First, it investigates a set of biology-based innovative CI algorithms which can enable a high throughput under extract from the insights of data. Second, it summarizes the core working principles of these algorithms systematically and highlights their preliminary applications in different areas of data mining. This will allow us to clearly pinpoint the intrinsic strengths of these novel algorithms, and also to define the potential further research directions. The findings of this chapter should provide useful insights into the current big data literature and be a good source for anyone who is interested in the application of CI approaches to big data and its corresponding fields.

B. Xing (✉)
Computational Intelligence, Robotics, and Cybernetics for Leveraging
E-Future (CIRCLE), Department of Computer Science, School of Mathematical
and Computer Sciences, Faculty of Science and Agriculture, University of Limpopo,
Private Bag X1106, Sovenga, Limpopo 0727, South Africa
e-mail: bxing2009@gmail.com

K.K. Ravulakollu et al. (eds.), *Trends in Ambient Intelligent Systems*,
Studies in Computational Intelligence 633, DOI 10.1007/978-3-319-30184-6_6

Keywords Innovative computational intelligence · Ubiquitous robotics systems · Data mining · Artificial fish swarm algorithm · Dove swarm optimization · Firefly algorithm · Fireworks optimization algorithm · Flockbyleader · Flocking-based algorithm · Fruit fly optimization algorithm · Glowworm swarm optimization · Harmony search · Human group formation · Photosynthetic algorithm · Shark-Search algorithm · Stem cells optimization algorithm · Wasp optimization algorithm

1 Introduction

The amount of data in our world has been exploding, in particular, with the emergence of the advanced infrastructures, e.g., ambient intelligent system, Internet of things (IoT) and cloud computing. For instance, in ubiquitous robotics system domain, based on the IoT concept, the living (patients, medical staff, etc.) and non-living (medical sensors, smart devices, etc.) entities can be easily linked and thus open new opportunities for right care. Obviously, that is good which could bring great benefits to the medical staff and patients. But how we measure the value of those big data (i.e., achieving highly scalable data analysis) is a circuital challenge, as the overall applications' potential value is highly dependent on the data discovery.

The aim of this chapter is twofold: first, identifying a set of novel biology-based computational intelligence (CI) algorithms that have been applied to various data mining problems and summarizing their corresponding core working principles; second, presenting a detailed analysis regarding each of these algorithms' preliminary applications in terms of data mining. Having this in mind, the remainder of this chapter is organized as follows: in Sect. 2, the background information is briefly introduced; the core working principles of biology-based CI algorithms and their corresponding applications in data mining are detailed in Sects. 3, 4 discusses some findings obtained via this study; the limitations of the present work are outlined in Sect. 5; finally, the conclusion drawn in Sect. 6 close this study.

2 Background

2.1 What Is Big Data?

Big data refers to datasets whose size is beyond the ability of traditional or commonly used techniques to capture, store, manage, and analyze within a "tolerable elapsed time" [1]. Of course, this definition is relative and evolves in time as technology progresses. However, the question is what this phenomenon means. Is the proliferation of data simply create potential value for us? The answer is not. A shortage of the analytical and managerial method necessary to make the most of big data is a significant and pressing challenge. In addition, several authors (e.g., [2, 3]) pointed out that the focus not only on size but on three different dimensions of growth for data, i.e., volume, variety and velocity. In fact, it is not just the size of an individual data set, but rather the collection of data that is available to us through different formats (such as pictures, sound, movies, documents, and experimental measurements) and different sources (such as radio-frequency identification (RFID), GPS, and online). In a similar vein, the authors of [4] proposed HACE theorem (i.e., heterogeneous and autonomous sources, and complex and evolving relationships among data) in terms of big data characteristics. As a result, the big data forces us to use or create innovative methodologies.

2.2 Data Issue Associated with Ubiquitous Robotics Systems

Inevitably, ubiquitous robotics systems belong to data-intensive domain, due to the fact that they are often related to a large number of continuously changing data objects, such as data streams from sensors and tracking devices. As a result, the complexity of managing different sources of sensor-based information can easily make our head spin. In the light of this statement, the authors of [5] provided an insight into the use of sensor-based technology to support ambient assisted living. Additionally, in Ref. [6], a comprehensive framework for managing continuously changing data objects is presented. Indeed, sensor-based ambient intelligent system can function proactively if the information that hidden in these piles of data can be extracted.

2.3 Data Mining

But why are we interested in data? It is common belief that data without a model is just noise. In other words, data needs to be processed and condensed into more connected forms in order to be useful for decision making. That is the job of data mining. Let us now more precisely define data mining. Data mining is used to describe salient features in the data [7, 8]. For big data, it is the application of data mining techniques to discover patterns from the big amount of data.

According to the literature (e.g., [9–11]), data mining techniques can be categorized into several different types: generalization, characterization, classification, clustering, association, evolution, pattern matching, data visualization, and meta-rule guided mining. In addition, the authors of [12] argued that using a class of algorithms can effectively extract information from huge amounts of data in many data repositories or in dynamic data streams.

2.4 Innovative Computational Intelligence

Computational intelligence (CI) is a fairly new research discipline, and thus, there is little agreement regarding its accurate definition. However, most academicians and practitioners would include techniques such as artificial neural networks (ANN), fuzzy systems (FS), many versions of evolutionary algorithms (EA) (e.g., evolution strategies (ES), genetic algorithm (GA), genetic programming (GP), differential evolution (DE)), as well as ant colony optimization (ACO), artificial immune systems (AIS), multi-agent systems (MAS), particle swarm optimization (PSO), and the hybridization versions of these, under the umbrella of CI. According to [13], these algorithms is often referred to as the traditional CI.

In data mining field, ANN and GA are examples of some widely used data mining algorithms. For example, for extracting the rules form databases, the authors of [14] presented a new algorithm that via trained NN using a GA. More recently, other CI methods, such as ACO, PSO, AIS have been applied to the data mining domain. For example, the authors of [15] used ant colony decision rule algorithm for finding if-then rules for classification tasks. Also, based centrally on ACO algorithm, the authors of [16] developed a new algorithm for data clustering. Inspired by one theoretical model of the immune system, i.e., immune networks, the authors of [17] represented a new algorithm called aiNet for data clustering. In addition, [18] proposed a multi-objective chaotic PSO method as search strategy to deal with classification rule mining problem.

Although the applications of traditional CI approaches in the field of data mining is well documented by several excellent reviews (e.g., [7, 9, 19–21]), the author of this study still intends to look at data mining through CI but via another lens, i.e., innovative CI. For the sake of clarity, the algorithms covered by this work are listed as below:

- Artificial Fish Swarm Algorithm
- Dove Swarm Optimization
- Firefly Algorithm
- Fireworks Optimization Algorithm
- FlockbyLeader
- Flocking-based Algorithm
- Fruit Fly Optimization Algorithm
- Glowworm Swarm Optimization
- Harmony Search
- Human Group Formation
- Photosynthetic Algorithm
- Shark-Search Algorithm
- Stem Cells Optimization Algorithm
- Wasp Optimization Algorithm.

3 Biology-Based Innovative CI Algorithms

Briefly, biology can be defined as a comprehensive science concerning all functions of living systems [22]. From an evolutionary process point of view, biological systems possess many appealing characteristics such as sophistication, robustness, and adaptability [23]. These features represent a strong motivation for imitating the mechanisms of natural evolution in an attempt to create CI algorithms with merits comparable to those of biological systems.

3.1 Artificial Fish School Algorithm (AFSA)

Artificial fish school algorithm (AFSA), which was proposed in [24], is a stochastic search optimization algorithm inspired by the natural social behaviour of fish schooling. In principle, AFSA is started first in a set of random generated potential solutions, and then performs the search for the optimum one interactively [25]. The main steps of AFSA are outlined as follows [24, 26, 27]:

- Assuming in an n–dimensional searching space, there is a group composed of K articles of artificial fish (AF).
- Situation of each individual AF can be expressed as vector $X = (x_1, x_2, \ldots, x_k)$ is denoted the current state of AF, where $x_k (k = 1, 2, \ldots, k)$ is control variable.
- $Y = f(X)$ is the fitness or objective function of X, which can represent food concentration (FC) of AF in the current position.
- $d_{ij} = \left\| X_i - X_j \right\|$ is denoted the Euclidean distance between fishes.
- *Visual* and *Step* are denoted respectively the visual distance of AF and the distance that AF can move for each step.
- $X_v = \left(x_1^v, x_2^v, \ldots, x_k^v\right)$ is the visual position at some moment. If the state at the visual position is better than the current state, it goes forward ad step in this direction, and arrives the X_{next} state, otherwise, continues an inspecting tour in the vision.
- *try-number* is attempt times in the behaviour of prey.
- δ is the condition of jamming $(0 < \delta < 1)$.

The basic behaviours of AF inside water are defined as follows [24, 26, 27]:

- Chasing trail behaviour (AF_Follow): When a fish finds the food dangling quickly after a fish, or a group of fishes, in the swarm that discovered food. If $Y_j > Y_i$ and $\frac{n_f}{n} < \delta$, then the AF_Follow behaviour is defined as follows [24, 26–28]:

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_j - X_i^{(t)}}{\left\| X_j - X_i^{(t)} \right\|} \cdot Step \cdot rand(\ ). \tag{1}$$

- Gathering behaviour (AF_Swarm): In order to survive and avoid hazards, the fish will naturally assemble in groups. There are three rules while fish gathering: firstly, a fish will try to keep a certain distance with each other to avoid crowding (i.e., Compartmentation Rule); secondly, a fish will try to move in a similar direction with its surrounding partners (i.e., Unification Rule); finally, a fish will try to move to the centre of its surrounding partners (i.e., Cohesion Rule). If $Y_c > Y_i$ and $\frac{n_f}{n} < \delta$, then the AF_Swarm behaviour is defined as follows [24, 26–28]:

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_c - X_i^{(t)}}{\left\| X_c - X_i^{(t)} \right\|} \cdot Step \cdot rand(\ ). \tag{2}$$

where X_c denotes the centre position of AF, X_i be the AF current state, n_f be the number of its companions in the current neighbourhood ($d_{ij} < Visual$), and n is the total fish number.
- Random searching behaviour (AF_Random): This is a basic biological behaviour that tents to the food. Generally the fish perceives the concentration of food in water to determine the movement by vision or sense and then chooses the tendency. The effect of this behaviour is similar to that of mutation operator in GA (i.e., genetic algorithm). It is defined as follows [28]:

$$X_i^{(t+1)} = X_i^{(t)} + Visual \cdot rand(\ ). \tag{3}$$

- Leaping behaviour (AF_Leap): When a fish 'stagnates' in a region, it looks for food in other regions defining the leaping behaviour. It is can be defined as follows [28]:

$$\begin{cases} \text{if } \left(FC_{best}(m) - FC_{best}(n)\right) < eps \\ \text{then } X_{some}^{(t+1)} = X_{some}^{(t)} + \beta \cdot Visual \cdot rand(\ ) \end{cases}. \tag{4}$$

- Foraging behaviour (AF_Prey): As feeding the fish, they will gradually move to the place where food is increasing. It is defined as follows, respectively [24, 26–28]:

$$X_j = X_i + Visual \cdot rand(\ ). \tag{5}$$

where X_i be the AF current state and select a state X_j randomly in its visual distance, $rand(\ )$ is a random function in the range, and $Visual$ represents the visual distance.

$$\begin{cases} \text{if } Y_i < Y_j, & \text{then } X_i^{(t+1)} = X_i^{(t)} + \frac{X_j - X_i^{(t)}}{\left\| X_j - X_i^{(t)} \right\|} \cdot Step \cdot rand(\ ) \\ \text{if } Y_i > Y_j, & \text{then} \begin{cases} X_j = X_i + Visual \cdot rand(\ ) \\ X_i^{(t+1)} = X_i^{(t)} + Visual \cdot rand(\ ) \end{cases} \end{cases}. \tag{6}$$

where Y is the food concentration (objective function value), $X_i^{(t+1)}$ represents the AF's next state, and *Step* denotes the distance that AF can move for each step.

Taking into account the above mentioned behaviours, the steps of implementing AFSA can be summarized as follows [28]:

- Step 1: Generate the initial fish swarm randomly in the search space.
- Step 2: Initialize the parameters.
- Step 3: Evaluate the fitness value of each AF.
- Step 4: Selecting behaviour. Each AF simulate the swarming and following behaviour, respectively, and select the best behaviour to perform by comparing the function values, the default is searching food behaviour.
- Step 5: Update the function value of the AF again.
- Step 6: Check the termination condition.

Data Mining using AFSA. Research, such as [29], employed AFSA as new tool to discover classification rules from data. Other than being a classifier (i.e., for higher classification accuracy), the AFSA is able to automatically mine a set of smaller IF-THEN rule. The proposed method has been tested on publicly available databases (i.e., Iris Plants Database), compared with other algorithms, the computational results showed that AFSA is very effective and robust. In addition, several authors, such as [30–32], proposed a hybrid clustering method, based on AFSA and K-means. Also, the fuzzy clustering method (i.e., fuzzy C-means) is combined within AFSA, e.g., [33–35]. Simulation results showed that AFSA could achieve good clustering effects.

3.2 *Dove Swarm Optimization (DSO) Algorithm*

Dove swarm optimization (DSO) algorithm was recently proposed in [36]. The basic working principles of DSO are listed as follows [36]:

- Step 1: Initializing the number of doves and deploying the doves on the 2-dimensional artificial ground.
- Step 2: Setting the number of epochs ($e = 0$), and the degree of satiety, $f_j^e = 0$ for $j = 1, \ldots, M \times N$. Initializing the multi-dimensional sense organ vector, $\vec{w}_j$ for $j = 1, \ldots, M \times N$.
- Step 3: Computing the total amount of the satiety degrees in the flock, $T(e) = \sum_{j=1}^{M \times N} f_j^e$.
- Step 4: Presenting an input pattern (i.e., piece of artificial crumb) $\vec{x}_k$ to the $M \times N$ doves.

- Step 5: Locating the dove b_f closest to the crumb $\vec{x}_k$ according to the minimum-distance criterion shown below [36]:

$$b_f = \arg\min_j \left\| \vec{x}_k - \vec{w}_j(k) \right\|, \text{ for } j = 1, \ldots, M \times N. \tag{7}$$

  The dove with the artificial sense organ vector which is the most similar to the artificial crumb, $\vec{x}_k$, is claimed to be the winner.
- Step 6: Updating each dove's satiety degree as follows [36]:

$$f_j^e(new) = \frac{\left\| \vec{x}_k - \vec{w}_{b_f}(k) \right\|}{\left\| \vec{x}_k - \vec{w}_j(k) \right\|} + \lambda f_j^e(old), \text{ for } j = 1, \ldots, M \times N. \tag{8}$$

- Step 7: Selecting the dove, b_f, with the highest satiety degree based on the following criterion expressed as follows [36]:

$$b_s = \arg \max_{1 \le j \le M \times N} f_j^e. \tag{9}$$

- Step 8: Updating the sense organ vectors and the position vectors via the following equations, respectively [36]:

$$\vec{w}_j(k+1) = \begin{cases} \vec{w}_{b_f}(k) + \eta_w\left(\vec{x}_k - \vec{w}_{b_f}(k)\right) & \text{for } j = b_f \\ \vec{w}_j(k) & \text{for } j \neq b_f \end{cases}. \tag{10}$$

$$\vec{p}_j(k+1) = \vec{p}_j(k) + \eta_p \beta\left(\vec{p}_{b_s}(k) - \vec{p}_j(k)\right), \text{ for } j = 1, \ldots, M \times N. \tag{11}$$

- Step 9: Returning to Step 4 until all patterns are processes.
- Step 10: Stopping the whole training procedure if the following criterion is met [36]:

$$\left| \sum_{j=1}^{M \times N} f_j^e - T(e) \right| \le \varepsilon. \tag{12}$$

  Otherwise, increasing the number of epochs by one $(e = e + 1)$, and go back to Step 3 until the pre-defined limit for the number of epochs is met. The satisfaction of the criterion given above means that the total amount of satiety degree has converged to some extent.

Data Mining using DSO. In general there are two main obstacles encountered in data clustering: the geometric shapes of the clusters are full of variability, and the cluster numbers are not often known a priori. In order to determine the optimal number of clusters, the authors of [36] employed DSO to perform data projection task, i.e., projecting high-dimensional data onto a low-dimensional space to facilitate visual inspection of the data. This process allows us to visualize high-dimensional data as a 2-dimensional scatter plot.

The basic idea in their work can be described as follows [36]: In a data set, each data pattern, $\vec{x}$, is regarded as a piece of artificial crumb and these artificial crumbs (i.e., data patterns) will be sequentially tossed to a flock of doves on a two-dimensional artificial ground. The flock of doves adjusts its physical movements to seek these artificial crumbs. Individual members of the flock can profit from discoveries of all of the other members of the flock during the foraging procedure because an individual is usually influenced by the success of the best individual of the flock and thus has a desire to imitate the behaviour of the best individual. Gradually, the flock of the doves will be divided into several groups based on the distributions of the artificial crumbs. Those formed groups will naturally correspond to the hidden data structure in the data set. By viewing the distributions of the doves on the 2-dimensional artificial ground, one may quickly find out the number of clusters inherent in the data set.

However, many practical data sets have high-dimensional data points. Therefore, the aforementioned idea has to be generalized so that it can process high-dimensional data. In the real world, each dove has a pair of eyes to find out where crumbs are, but in the artificial world, a virtual dove does not have the capability to perceive a piece of multi-dimensional artificial crumb that is located around it. In order to cope with issue, the authors of [36] equipped each dove with functionalities, i.e., a multi-dimensional artificial sense organ represented as a sense organ vector, $\vec{w}$, which has the same dimensionality as a data pattern, $\vec{x}$, and a 2-dimensional position vector, $\vec{p}$, which represents its position on the 2-dimensional artificial ground. In addition to these two vectors, $\vec{w}$ and $\vec{p}$, a parameter called the satiety parameter is also attached to each dove. This special parameter endows a dove with the ability of expressing its present satiety status with respect to the food, that is, a dove with a low degree of satiety will have a strong desire to change its present foraging policy and be more willing to imitate the behaviour of the dove which performs the best among the flock.

To test the performance of DSO, five (two artificial and three real) data sets were selected in the study. These data sets include Two-Ellipse, Chromosomes, Iris, Breast Cancer, and 20-Dimensional Non-Overlapping. The projection capability of DSO was compared with the other popular projection algorithms, e.g., Sammon's algorithm. For DSO, the maximum number of epochs for every data set (excluding Iris and 20-Dimensional data sets) were set to be 5, while for the Iris and 20-Dimensional data sets, were set to be 10 and 20, respectively. The case studies showed that DSO can fulfil the projection task. Meanwhile, the performance of DSO is not so sensitive to the size of dove swarm.

3.3 *Firefly Algorithm (FA)*

Firefly algorithm (FA) is a nature-inspired, optimization algorithm which is based on the social (flashing) behaviour of fireflies, or lighting bugs, in the summer sky in the tropical temperature regions [37–39]. In FA, physical entities (fireflies) are randomly distributed in the search space. They carry a bio-luminescence quality, called luciferin, as a signal to communicate with other fireflies, especially to prey attractions [40].

In detail, each firefly is attracted by the brighter glow of other neighbouring fireflies. The attractiveness decreases as their distance increases. If there is no brighter one than a particular firefly, it will move randomly. Its main merit is the fact that the FA uses mainly real random numbers and is based on the global communication among the swarming particles (i.e., the fireflies), and as a result, it seems more effective in multi-objective optimization.

Normally, FA uses the following three idealized rules [37] to simplify its search process to achieve an optimal solution:

- Fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex, that means no mutation operation will be done to alter the attractiveness fireflies have for each other;
- The sharing of information or food between the fireflies is proportional to the attractiveness that increases with a decreasing Cartesian or Euclidean distance between them due to the fact that the air absorbs light. Thus for any two flashing fireflies, the less bright one will move towards the brighter one. If there is no brighter one than a particular firefly, it will move randomly; and
- The brightness of a firefly is determined by the landscape of the objective function. For the maximization problems, the light intensity is proportional to the value of the objective function.

Furthermore, there are two important issues in the FA that are the variation of light intensity or brightness and formulation of attractiveness. Yang [38] simplifies a firefly's attractiveness β (determined by its brightness I) which in turn is associated with the encoded objective function. As light intensity and thus attractiveness decreases as their distance from the source increases, the variations of light intensity and attractiveness should be monotonically decreasing functions.

- Variation of light intensity: Suppose that there exists a swarm of n fireflies, and x_i, $i = 1, 2, \ldots, n$ represents a solution for a firefly i initially positioned randomly in the space, whereas $f(x_i)$ denotes its fitness value. In the simplest form, the light intensity $I(r)$ varies with the distance r monotonically and exponentially. That is determined as follows [37–39]:

$$I = I_0 e^{-\gamma r_{ij}}. \tag{13}$$

  where I_0 is the original light intensity, γ is the light absorption coefficient, and r is the distance between firefly i and firefly j at x_i and x_j as Cartesian distance $r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2}$ or the ℓ_2-norm, where $x_{i,k}$ is the kth component of the spatial coordinate x_i of the ith firefly and d is the number of dimensions we have, for $d = 2$, we have $r_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$.
- Movement toward attractive firefly: A firefly attractiveness is proportional to the light intensity seen by adjacent fireflies [38]. Each firefly has its distinctive

attractiveness β which implies how strong it attracts other members of the swarm. However, the attractiveness is relative; it will vary with the distance between two fireflies. The attractiveness function $\beta(r)$ of the firefly is determined via the following equation [37–39]:

$$\beta = \beta_0 e^{-\gamma r_{ij}^2}. \tag{14}$$

where β_0 is the attractiveness at $r = 0$, and γ is the light absorption coefficient which controls the decrease of the light intensity.
The movement of a firefly $\boldsymbol{i}$ at location x_i attracted to another more attractive (brighter) firefly j at location x_j is determined as follows [37–39]:

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j - x_i\right) + \alpha\varepsilon_i. \tag{15}$$

where the first term is the current position of a firefly, the second term is used for considering a firefly's attractiveness to light intensity seen by adjacent fireflies, and the third term is randomization with the vector of random variables ε_i being drawn from a Gaussian distribution, in case there are not any brighter ones. In addition, the coefficient α is a randomization parameter determined by the problem of interest.

- Special cases: From the equation above, it is easy to see that there exit two limit cases when γ is small or large, respectively [37–39]. When γ tends to zero, the attractiveness and brightness are constant $\beta = \beta_0$ which means the light intensity does not decrease as the distance r between two fireflies increases. Therefore, a firefly can be seen by all other fireflies, a single local or global optimum can be easily reached. This limiting case corresponds to the standard particle swarm optimization algorithm. On the other hand, when γ is very large, then the attractiveness (and thus brightness) decreases dramatically, and all fireflies are short-sighted or equivalently fly in a deep foggy sky. This means that all fireflies move almost randomly, which corresponds to a random search technique.

In general, the FA corresponds to the situation between these two limit cases, and it is thus possible to fine-tune these parameters, so that FA can find the global optima as well as all the local optima simultaneously in a very effective manner. A further advantage of FA is that different fireflies will work almost independently, it is thus particular suitable for parallel implementation. It is even better than genetic algorithm and particle swarm optimization because fireflies aggregate more closely around each optimum. It can be anticipated that the interactions between different sub-regions are minimal in parallel implementation.

Overall, taking into account the basic information described above, the steps of implementing FA can be summarized as follows [39, 41]:

- Step 1: Generate initial the population of fireflies placed at random positions within the n-dimensional search space.
- Step 2: Initialize the parameters, such as the light absorption coefficient (γ).

- Step 3: Define the light intensity (I_i) of each firefly (x_i) as the value of the cost function, $f(x_i)$.
- Step 4: For each firefly (x_i), compare its light intensity with the light intensity of every other firefly (e.g. x_j).
- Step 5: If ($I_j > I_i$), then move firefly x_i towards x_j in n-dimensions.
- Step 6: Calculate the new values of the cost function for each firefly and update the light intensity.
- Step 7: Rank the fireflies and determine the current best.
- Step 8: Repeat Steps 3–7 until the termination criteria is satisfied.

Data Mining using FA. The authors of [42] used FA to deal with clustering problem. A set of well-known and well-used benchmark data set have been used to test the performance. Compared with other algorithms (i.e., artificial bee colony and PSO), the simulation results showed that FA can be efficiently used for clustering.

3.4 Fireworks Optimization Algorithm (FOA)

In this section, we will introduce an emerging CI algorithm that is derived from the explosion process of fireworks, an explosive devices invented by our clever ancestor, which can produce striking display of light and sound [43].

Fireworks optimization algorithm (FOA) was recently proposed in [44]. The basic idea was when we need to find a point x_j satisfying $f(x_i) = y$, a set of fireworks will be continuously fired in the potential search space until an agent (i.e., a spark in fireworks context) gets to or reasonably close to the candidate point x_j. Based on this understanding, to implement FOA, the following steps need to be performed [44–46]:

- Step 1: Fireworks explosion process designing. Since the number of sparks and their coverage in the sky determines whether an explosion is good or not, Tan and Zhu [44] first defined the number of sparks created by each firework x_j as follows:

$$s_i = m \cdot \frac{y_{\max} - f(x_i) + \xi}{\sum_{i=1}^{n} [y_{\max} - f(x_i)] + \xi}. \tag{16}$$

  where m is a parameter used to control the total number of sparks created by the n fireworks, $y_{\max} = \max(f(x_i))$ (for $i = 1, 2, \ldots, n$) stands for the maximum value of the objective function among the $y_{\max}$ fireworks, and ξ represents a small constant which is used to avoid zero-division-error.

  Meanwhile, in order to get rid of the overwhelming effects of the splendid fireworks, bounds s_i are also defined as follows [44]:

$$\hat{s}_i = \begin{cases} round(a \cdot m) & \text{if } s_i < am \\ round(b \cdot m) & \text{if } s_i > bm,\ a < b < 1\,. \\ round(s_i) & \text{otherwise} \end{cases} \tag{17}$$

  where a and b are constant parameters.

Next, Tan and Zhu [44] also designed the explosion amplitude as follows:

$$A_i = \hat{A} \cdot \frac{f(x_i) - y_{\min} + \xi}{\sum_{i=1}^{n} [f(x_i) - y_{\min}] + \xi}. \tag{18}$$

where $\hat{A}$ represents the maximum amplitude of an explosion, and $y_{\min} = \min(f(x_i))$ (for $i = 1, 2, \ldots, n$) denotes the minimum value of the objective function among the n fireworks.
Finally, the directions of the generated sparks are computed as follows [44]:

$$z = round(d \cdot rand(0, 1)). \tag{19}$$

where d denotes the dimensionality of the location x, and $rand(0, 1)$ represents an uniformly distributed number within [0, 1].

- Step 2: In order to obtain a good implementation of FOA, the locations of where we want the fireworks to be fired need to be chosen properly. According to [44], the general distance between a location x and other locations can be expressed as follows:

$$R(x_i) = \sum_{j \in K} d(x_i, x_j) = \sum_{j \in K} \|x_i - x_j\|. \tag{20}$$

where K denotes a group of current locations of all fireworks and sparks.

The selection probability of a location x_i is then defined as follows [44]:

$$p(x_i) = \frac{R(x_i)}{\sum_{j \in K} R(x_j)}. \tag{21}$$

Data Mining using FOA. To validate the performance of the proposed FOA, 9 benchmark test functions were chosen by Tan and Zhu [44] and the comparisons were conducted among the standard PSO, and the clonal PSO. The experiment results indicated that the FOA clearly outperforms the other algorithms in both optimization accuracy and convergence speed. In addition, the authors of [46] used three sampling data methods to approximate fitness landscape in order to accelerating FOA.

3.5 Shark-Search Algorithm (SSA)

Shark-search algorithm (SSA) was originally proposed in [47] for enhancing the Web browsing and search engine performance [48–50]. There are several variants and applications of SSA can be found in the literature [51–53]. The SSA is built on its predecessor called "fish-search" algorithm and the key principles underlying them are the following:

The algorithm first takes an input as a seed URL (standing for uniform resource locator) and search query. Then it dynamically sets up a priority sequence for the next URLs to be explored. At each step, the first node is popped from the list and attended. As each file's text becomes available, it will be analyzed by a scoring component and then evaluated for its relevance or irrelevance to the search query.

Putting it simply, in SSA, when relevant information (standing for food) is discovered, searching agents (i.e., fish) reproduce and keep looking for food. They will die when the food is in absent condition or encountering polluted water (poor bandwidth situation). The original fish-search algorithm suffers from the following limitations [47]:

- First, the relevance score is assigned in a discrete way.
- Second, the differentiation degree of the priority of the pages in the list is very low.
- Third, the number of addressed children is reduced by arbitrary using the width parameter.

Bearing this in mind, there are several improvements provided in SSA [47]:

- Improvement 1: A similarity engine is introduced to rank the document relevance degree.
- Improvement 2: Refining the computation of the potential score of the children by taking the following two factors into account. First, propagating ancestral relevance scores deeper down the hierarchical structure. Second, making use of the meta-information carried by the links to the files.

Data Mining using SSA. To evaluate the efficacy of SSA, a measure called "getting as many relevant documents with the shorted delays" was proposed in [47]. By testing SSA on four case studies, the significant improvements were experimentally verified which offer SSA an advantage to replace original fish-search algorithm in dealing with dynamic and precise searches within the limited time range.

3.6 *FlockbyLeader Algorithm*

The FlockbyLeader algorithm was proposed by Bellaachia and Bari [54] in which the recently discovered leadership dynamic mechanisms in pigeon flocks are incorporated in the normal flocking model (i.e., Craig Reynolds' Model [55]). In every iteration, the algorithm starts by finding flock leaders. The main steps are illustrated as follows [54]:

- Calculating fitness value of each flock leader (L_i) according to the objective function (i.e., $d_{\max}^{L_i}$). It will be defined as follows [54]:

$$d_{\max}^{L_i} = \max_{o \in kNB_t(x_i)} \{\rho(x_i, o)\}. \tag{22}$$

  where $kNB_t(x_i)$ is the k-neighbourhood of x_i at iteration t, $d_{\max}^{L_i}$ as radius associated with leader L_i at iteration t, x_i is a node in the feature graph, and $\rho(x_i, o)$ is the given distance function between objects x_i and o.

- Ranking the LeaderAgent (A_i). This procedure is defined as follows, respectively [54]:

$$Rank_t(A_i) = \mathrm{Log}\left(\frac{|N_{i,t}|}{|N_t|} \cdot 10\right) \cdot ARF_t(A_i). \tag{23}$$

$$ARF_t(A_i) = \frac{|DR\_kNB_t(x_i)|}{|DR\_kNB_t(x_i)| + |D\_kNB_t(x_i)|}. \tag{24}$$

$$\begin{cases} \text{if } ARF_t(A_i) \geq 0.5, & \text{then } x_i \text{ is a flockleader} \\ \text{if } ARF_t(A_i) < 0.5, & \text{then } x_i \text{ is a follower} \end{cases}. \tag{25}$$

  where $DR\_kNB_t(x_i)$ represents the dynamic reverse k-neighbourhood of x_i at iteration t, $ARF_t(A_i)$ is the dynamic agent role factor of the agent A_i at iteration t, $|N_{i,t}|$ is the number of the neighbours A_i at iteration t, and $|N_t|$ is the number of unvisited nodes at iteration t.
- Performing the flocking behaviour.
- Updating the FindFlockLeaders (G_f).

Data Mining using FlockbyLeader Algorithm. To test the efficiency of the FlockbyLeader algorithm, two large datasets that one is consists of 100 news articles collected from cyberspace, and the other one is the Iris Plant Dataset were adopted by Bellaachia and Bari [54]. Compared with other CI algorithms, the proposed algorithm is significant in improving the results.

3.7 Flocking-Based Algorithm (FBA)

Flocking-based algorithm (FBA) was originally proposed in [56–58]. The basic flocking model is composed of three simple steering rules (see below) that need to be executed at each instance over time, for each individual agent.

- Rule 1: Separation. Steering to avoid collision with other boids nearby.
- Rule 2: Alignment. Steering toward the average heading and speed of the neighboring flock mates.
- Rule 3: Cohesion. Steering to the average position of the neighboring flock mates.

In the proposed algorithm, a fourth rule is added as below:

- Rule 4: Feature similarity and dissimilarity rule. Steering the motion of the boids with the similarity among targeted objects.

All these four rules can be formally express by the following equations [56]:

- The function of separation rule is to act as an active boid trying to pull away before crashing into each other. The mathematical implementation of this rule is thus can be described as follows [56]:

$$d(P_x, P_b) \leq d_2 \Rightarrow \vec{v}_{sr} = \sum_{x}^{n} \frac{\overline{\vec{v}_x + \vec{v}_b}}{d(P_x, P_b)}. \tag{26}$$

  where v_{sr} is velocity driven by Rule 1, d_2 is the distance pre-defined, v_b and v_x are the velocities of boids B and X.
- The function of alignment rule is to act as the active boid trying to align its velocity vector with the average velocity vector of the flock in its local neighbourhood. The degree of locality of this rule is determined by the sensor range of the active flock boid. This rule can be presented in a mathematical way through the following equation [56]:

$$d(P_x, P_b) \leq d_1 \cap d(P_x, P_b) \geq d_2 \Rightarrow \vec{v}_{ar} = \frac{1}{n} \sum_{x}^{n} \vec{v}_x. \tag{27}$$

  where v_{cr} is velocity driven by Rule 3, d_1 and d_2 are pre-defined distance, and $(P_x - P_b)$ calculates a directional vector point.
- The flock boid tries to stay with the other boids that share the similar features with it. The strength of the attracting force is proportional to the distance (between the boids) and the similarity (between the boids' feature values) which can be expressed as follows [56]:

$$v_{ds} = \sum_{x}^{n} (S(B, X) \times d(P_x, P_b)). \tag{28}$$

  where v_{ds} is the velocity driven by feature similarity, $S(B, X)$ is the similarity value between the features of boids B and X.
- The flock boid attempts to stay away from other boids with dissimilar features. The strength of the repulsion force is inversely proportional to the distance (between the boids) and the similarity value (between the boids' features) which are defined as follows [56]:

$$v_{dd} = \sum_{x}^{n} \frac{1}{S(B, X) \times d(P_x, P_b)}. \tag{29}$$

  where v_{dd} is the velocity driven by feature dissimilarity. To get comprehensive flocking behavior, the actions of all the rules are weighted and summed to obtain a net velocity vector required for the active flock boid using the following equation [56]:

$$v = w_{sr}v_{sr} + w_{ar}v_{ar} + w_{cr}v_{cr} + w_{ds}v_{ds} + w_{dd}v_{dd}. \tag{30}$$

where v is the boid's velocity in the virtual space, and w_{sr}, w_{ar}, w_{cr}, w_{ds}, w_{dd} are pre-defined weight values.

Data Mining using FBA. Document clustering is an essential operation used in unsupervised document organization, automatic topic extraction, and information retrieval. It provides a structure for organizing large bodies of data (in text form) for efficient browsing and searching. The authors of [56] utilized FBA for document clustering analysis. A synthetic data set and a real document collection (including 100 news articles collected from the Internet) were used in their study. In the synthesis data set, four data types were included with each containing 200 2-dimensional (x, y) data objects. Parameters x and y are distributed according to Normal distribution $N(\mu, \sigma)$; while for the real document collection data set, 100 news articles collected from the Internet at different time stages were categorized by human experts and manually clustered into 12 categories such as Airline safety, Iran Nuclear, Storm Irene, Volcano, and Amphetamine. In order to reduce the impact of the length variations of different documents, the authors of [56] further normalized each file vector to make it in unit length. Each term stands one dimension in the document vector space. The total number of terms in the 100 stripped test files is thus 4790 (i.e., 4790 dimensions). The experimental studies were carried out on the synthetic and the real document collection data sets, respectively, among FBA and other popular clustering algorithms such as ant clustering algorithm and K-means algorithm. The final testing results illustrated that the FBA can have better performance with fewer iterations in comparison with the K-means and ant clustering algorithm. In the meantime, the clustering results generated by FBA were easy to be visualized and recognized even by an untrained human user.

3.8 Fruit Fly Optimization Algorithm (FFOA)

Fruit fly optimization algorithm (FFOA) was originally proposed in [59, 60] that is based on the food foraging behaviour of fruit fly. Generally, the procedures of FFOA are described as follows [59]:

- Initialization phase: The fruit flies are randomly distributed in the search space (*InitX_axis* and *InitY_axis*) via the following equations, respectively [59]:

$$X_i = X\_axis + RandomValue. \tag{31}$$

$$Y_i = Y\_axis + RandomValue. \tag{32}$$

where the term "*RandomValue*" is a random vector that were sampled from a uniform distribution.

- Path construction phase: The distance and smell concentration value of each fruit fly can be defined as follows, respectively [59]:

$$Dist_i = \sqrt{X_i^2 + Y_i^2}. \tag{33}$$

$$S_i = \frac{1}{Dist_i}. \tag{34}$$

  where $Dist_i$ is the distance between the ith individual and the food location, and S_i is the smell concentration judgment value which is the reciprocal of distance.
- Fitness function calculation phase. It can be defined as follows, respectively [59]:

$$Smell_i = Function(S_i). \tag{35}$$

$$[bestSmell, bestIndex] = \max(Smell_i). \tag{36}$$

  where $Smell_i$ is the smell concentration of the individual fruit fly, *bestSmell* and *bestIndex* represent the largest elements and its indices along different dimensions of smell vectors, and $\max(Smell_i)$ is the maximal smell concentration among the fruit flies.
- Movement phase: The fruit fly keeps the best smell concentration value and will use vision to fly towards that location via the following equations, respectively [59]:

$$Smellbest = bestSmell. \tag{37}$$

$$X\_axis = X(bestIndex). \tag{38}$$

$$Y\_axis = Y(bestIndex). \tag{39}$$

Overall, taking into account the key phases described above, the steps of implementing FFOA can be summarized as follows [59]:

- Step 1: Initialize the optimization problem and algorithm parameters.
- Step 2: Repeat till stopping criteria met. First, randomly select a location via distance and smell concentration judgment value. Second, calculate its fitness function $Function(S_i)$. Third, find out the fruit fly with maximal smell concentration among the fruit fly swarm. Fourth, rank the solutions and move to the best solution.
- Step 3: Post process and visualize results.

Data Mining using FFOA. In order to show how the FFOA performs, two functions (i.e., one minimum and one maximum) are tested in [59]. In addition, the authors used FFOA to deal with the financial distress data of Taiwan's enterprise, computational results showed that FFOA has a very good classification and prediction capability.

3.9 Glowworm Swarm Optimization (GSO) Algorithm

Also inspired by luminous insect, the glowworm swarm optimization (GSO) algorithm was originally proposed by Krishnanand and Ghose [61] to deal with multimodal problems. Just like ants, elephants, mice, and snakes, glowworms also use some chemical substances, called luciferin, as signals for indirect communication. By sensing luciferin, glowworms can be attracted by strongest luciferin concentrations. In this way, the final optimization results can be found.

Typically, each iteration of the GSO algorithm consists of two phases, namely, a luciferin-update phase and a movement phase. In addition, for GSO, there is a dynamic decision range update rule that is used to adjust the glowworms' adaptive neighbourhoods. The details are listed as below [62]:

- Luciferin-update phase: It is the process by which the luciferin quantities are modified. The quantities value can either increase, as glowworms deposit luciferin on the current position, or decrease, due to luciferin decay. The luciferin update rule is given as follows [62]:

$$l_i(t+1) = (1-\rho)\cdot l_i(t) + \gamma \cdot J \cdot [x_i \cdot (t+1)]. \tag{40}$$

  where $l_i(t)$ denotes the luciferin level associated with the glowworm i at time t, ρ is the luciferin decay constant $(0 < \rho < 1)$, γ is the luciferin enhancement constant, and $J(x_i(t))$ stands for the value of the objective function at glowworm i's location at time t.
- Movement phase: During this phase, glowworm i chooses the next position j to move to using a bias (i.e., probabilistic decision rule) toward good-quality individual which has higher luciferin value than its own. In addition, based on their relative luciferin levels and availability of local information, the swarm of glowworms can be partitioned into subgroups that converge on multiple optima of a given multimodal function. The probability of moving toward a neighbour is given as follows [62]:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} [l_k(t) - l_i(t)]}. \tag{41}$$

  where $j \in N_i(t)$, $N_i(t) = \left\{ j : d_{ij}(t) < r_d^i(t);\ l_i(t) < l_{js}(t) \right\}$ is the set of neighbours of glowworm i at time t, $d_{ij}(t)$ denotes the Euclidean distance between glowworms i and j at time t, and $r_d^i(t)$ stands for the variable neighbourhood range associated with glowworm i at time t.
  Based on the above equation, the discrete-time model of the glowworm movements can be stated as follows [62]:

$$x_i(t+1) = x_i(t) + s\left[\frac{x_j(t) - x_i(t)}{\left\|x_j(t) - x_i(t)\right\|}\right]. \tag{42}$$

where $x_i(t) \in \mathbf{R}^m$ is the location of glowworm i at time t in the m–dimensional real space, $\|\cdot\|$ denotes the Euclidean norm operator, and $s(>0)$ is the step size.

- Neighbourhood range update rule: In addition to the luciferin value update rule that is illustrated in the movement phase, in GSO the glowworms use a radial range (i.e., $\left(0 < r_d^i \le r_s\right)$) update rule to explore an adaptive neighbourhood (i.e., to detect the presence of multiple peaks in a multimodal function landscape). Let r_0 be the initial neighbourhood range of each glowworm (i.e., $r_d^i(0) = r_0\ \forall i$), then the updating rule is given as follows [62]:

$$r_d^i(t+1) = \min\left\{r_s,\ \max\left\{0,\ r_d^i(t) + \beta(n_t - |N_i(t)|)\right\}\right\}. \tag{43}$$

where β is a constant parameter, and $n_t \in N$ is a parameter used to control the number of neighbours.

Furthermore, in order to escape the dead-lock situation (i.e., all the glowworms converge to suboptimal solutions), Krishnanand and Ghose [62] employed a local search mechanism. The working principle is described as follows: during the movement phase, each glowworm moves a distance of step size (s) toward a neighbour. Hence, when $d_{ij}(t) < s$, glowworm i leapfrogs over the position of a neighbour j and becomes a leader to j. In the next iteration, glowworm i remains stationary and j overtakes the position of glowworm i, thus regaining its leadership. In this way, the GSO algorithm converges to a state in which all the glowworms construct the optimal solution over and over again.

Typically, by taking into account the basic rules described above, the steps of implementing the GSO algorithm can be summarized as follows [62]:

- Step 1: Initialize the parameters.
- Step 2: Initiation population of N candidate solution is randomly generated all over the search space.
- Step 3: The fitness function value corresponding to each candidate solution is calculated.
- Step 4: Perform the iteration procedures that include luciferin update phase, movement phase, and decision range update phase.
- Step 5: Check if maximum iteration is reached, go to step 3 for new beginning. If a specified termination criteria is satisfied, stop and return the best solution.

Data Mining using GSO. Based on GSO, the authors of [63] proposed two cluster analysis methods deal with data mining. Experimental results showed that the proposed algorithms have much potential.

3.10 Glowworm Swarm Optimization (GSO) Algorithm

Harmony search (HS) algorithm was originally proposed in Geem, Kim [64]. With the underlying fundamental of natural musical performance processes in which the musicians improvise their instruments' pitch by searching for the pleasing harmony (a perfect state), HS find the solutions through the determination of an objective function (i.e., the audience's aesthetics) in which a set of values (i.e., the musicians) assigned to each decision variable (i.e., the musical instrument's pitch). In general, the HS algorithm has three main operations: harmony memory (HM) consideration, pitch adjustment, and randomization [64]. The HS algorithm is performed in several steps, outlined below [64]:

- Preparation of harmony memory: The main building block of HS is the usage of HM, because multiple randomized solution vectors are stored in HM as follows [65]:

$$\mathrm{HM} = \begin{bmatrix} D_1^1 & D_2^1 & \cdots & D_n^1 & f\left(\mathbf{D}^1\right) \\ D_1^2 & D_2^2 & \cdots & D_n^2 & f\left(\mathbf{D}^2\right) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ D_1^{HMS} & D_2^{HMS} & \cdots & D_n^{HMS} & f\left(\mathbf{D}^{HMS}\right) \end{bmatrix}. \tag{44}$$

  where D_i^j is the ith decision variable in the jth solution vector that has one discrete value out of a candidate set $\{D_i(1), D_i(2), \ldots, D_i(k), \ldots, D_i(K_i)\}$, $f\left(\mathbf{D}^j\right)$ is the objective function value for the jth solution vector, and HMS is the harmony memory size (i.e., the number of multiple vectors stored in the HM).
- Improvisation of new harmony: A new harmony vector $D_i^{new} = \left(D_1^{new}, D_2^{new}, \ldots, D_n^{new}\right)$ is improvised by the following three rules [65]:

  1. Random selection: Based on this rule, one value is chosen out of the candidate set as follows [65]:

  $$D_i^{new} \leftarrow D_i(k),\ D_i(k) \in \{D_i(1), D_i(2), \ldots, D_i(K_i)\}. \tag{45}$$

  2. HM consideration: In memory consideration, one value is chosen out of the HM set with a probability of harmony memory consideration rate (HMCR) as follows [65]:

  $$D_i^{new} \leftarrow D_i(l),\ D_i(l) \in \left\{D_i^1, D_i^2, \ldots, D_i^{HMS}\right\}. \tag{46}$$

  3. Pitch adjustment: According to this rule, the obtained value as in Eq. 46 is further changed into neighbouring values, with a probability of pitch adjusting rate (PAR) as follows [65]:

$$D_i^{new} \leftarrow D_i(l \pm 1),\ D_i(l) \in \left\{D_i^1, D_i^2, \cdots, D_i^{HMS}\right\}. \tag{47}$$

Overall, these three rules are the core terms of the stochastic derivative of HS and can be summarized as follows [65]:

$$\begin{aligned}\left.\frac{\partial f}{\partial D_i}\right|_{D_i = D_i(l)} = & \frac{1}{K_i} \cdot (1 - HMCR) \\ & + \frac{n(D_i(l))}{HMS} \cdot HMCR \cdot (1 - PAR) \\ & + \frac{n(D_i(l \pm 1))}{HMS} \cdot HMCR \cdot PAR\end{aligned} \tag{48}$$

where $\frac{1}{K_i} \cdot (1 - HMCR)$ denotes for the rate to choose a value $D_i(l)$ for the decision variable D_i by random selection, $\frac{n(D_i(l))}{HMS} \cdot HMCR \cdot (1 - PAR)$ chooses the rate by HM consideration, and $\frac{n(D_i(l \pm 1))}{HMS} \cdot HMCR \cdot PAR$ chooses the rate by pitch adjustment.

- Update of HM: Once the new vector $D_i^{new} = \left(D_1^{new}, D_2^{new}, \cdots, D_n^{new}\right)$ is completely generated, it will be compared with the other vectors that stored in HM. If it is better than the worst vector in HM with respect to the objective function, it will be updated (i.e., the new harmony is included in the HM and the existing worst harmony is excluded from the HM).

In summary, the general optimization procedure of the HS algorithm can be given as follows [64, 66]:

- Step 1: Initialize the optimization problem and algorithm parameters.
- Step 2: Initialization of HM.
- Step 3: Improvise a new harmony from the HM.
- Step 4: Update the HM.
- Step 5: Repeat Steps 3 and 4 until the termination criterion is satisfied.

Data Mining using HS. In terms of data mining, the authors of [67] proposed a novel harmony K-means algorithm which based on HS for document clustering. Compared with other optimization methods, the proposed algorithm is capable of convergence to the best known optimum faster than others. In a similar vein, the authors of [68] and [69] hybridized K-means and HS for clustering web documents. Experimental results revealed that the proposed algorithm can find better clusters when compared to similar methods. To deal with classification accuracy, studies, such as [70] and [71], made a preliminary attempt. Also, HS is used to solving the feature selection problem, such as [72] and [73].

3.11 Human Group Formation (HGF) Algorithm

Human group formation (HGF) algorithm was recently proposed in [74]. The key concept of this algorithm is about the behaviour of in-group members that try to unite with their own group as much as possible, and at the same time maintain social distance from the out-group members. To implement the HGF algorithm, the following steps need to be performed [74]:

- Step 1: Cluster centres representation refers to the number of classes, number of available input patterns, and number, type, and scale of the features available to the clustering algorithm. At first, there are a total of Q clusters, which is equal to the number of target output classes.
- Step 2: Accuracy selection is usually measured by a distance function defined on pairs of patterns as follows [74]:

$$\begin{aligned}&\text{Accuracy} = \frac{\sum_{i=1}^{p} A_i}{P}\\ &A_i = \begin{cases} 1, & \text{if } J \in Y_i \\ 0, & \text{otherwise} \end{cases} \\ &J = \arg_j \min(d_j(X_i)), d_j(X_i) = \left\| X_i - z_j \right\|\end{aligned} \quad . \tag{49}$$

  where P denotes the total number of patterns in the training data set; J represents the index of a cluster whose reference pattern is the closest match to the incoming input pattern X_i; Y_i stands for the target output of the ith input pattern; z_j refers to the centre of the jth cluster; and $d_j(X_i)$ states the Euclidean distance between the input pattern X_i and the centre of the jth cluster.
- Step 3: The grouping/formation step can be performed in a way that in-group member try to unite with their own group and maintain social distance from the non-members as much as possible, update the centre value of each cluster (Z_j) as follows [74]:

$$\begin{aligned}&Z_{jk}^{new} = Z_{jk}^{old} + \Delta Z_{jk}\\ &\Delta Z_{jk} = \sum_{m \in q} \eta_{jm} \beta_j \delta_{jm} (Z_{mk} - Z_{jk}) - \sum_{n \notin q} \eta_{jn} \beta_j \delta_{jn} (Z_{nk} - Z_{jk})\end{aligned} \quad \cdot \tag{50}$$

  where k ($k = 1, 2, 3, \ldots, k$) is the number of features in the input pattern; q is the class to which the jth cluster belongs; $\eta_{jm} = e^{-\left[\left(Z_{jk} - Z_{mk}\right)/\sigma\right]^2}$ and $\eta_{jn} = e^{-\left[\left(Z_{jk} - Z_{nk}\right)/\sigma\right]^2}$ have values between 0 and 1which determine the influence of mth and nth clusters on the jth cluster. In general, the further apart mth and nth clusters are from the jth cluster, the lower the values of η_{jm} and η_{jn}; β_j is the velocity of the jth cluster with respect to its own ability to move in the search space; and δ_{jm} is the parameter to prevent clusters of the same class from being too close to one another and normally with respect to two factors: (1) the distance between the *j*th cluster and the *m*th cluster, and (2) the territorial boundary of the clusters (T). If the distance between the *j*th cluster and the *m*th cluster is less than T, the value of δ_{jm} will be decreased by a predefined amount. After each centre is updated, if the accuracy is higher, save this new center value and then continue updating the next cluster centre; if it is lower, discard the new center value and return to the previous centre; and if it does not change, save the new center value and decrease the value of β_j by a predefined amount.

- Step 4: Cluster validity analysis is the assessment of clustering procedure's output. The cluster which satisfies the following equation will be deleted [74]:

$$-\frac{1}{2\log_2\left(\frac{n_j}{p}\right)}\left(\frac{n_j^q}{n_j}\right)\left(\frac{\sum_{\forall X_i^j\in q}\left\|X_i^j - z_j\right\|}{n_j}\right) < \rho. \tag{51}$$

where n_j is the number of input patterns in the *j*th cluster; n_j^q is the number of input patterns in the *j*th cluster whose target outputs (Y) are q; X_i^j is the *i*th input pattern in the *j*th cluster; and ρ is the vigilance parameter.

- Step 5: Recalculating the accuracy of the model according to Eq. 49 [74].
- Step 6: For each remaining cluster, if the distance between the new centre updated in step 3 and the previous centre is less than 0.0001 ($\left\|Z_{jk}^{new} - Z_{jk}^{old}\right\| < 0.0001$), randomly pick k small numbers between −0.1 and 0.1, and then add them to the centre value of the cluster. The purpose of this step is to prevent the premature convergence of the proposed algorithm to sub-optimal solutions.
- Step 7: Terminating process is to check the end condition, if it is satisfied, stop the loop; if not, examine the following conditions: (1) if the accuracy of the model improves over the previous iteration, randomly select one input pattern from the training data set of each target output class that still has error. Then go to step 2; and (2) if the accuracy does not improve, randomly select the input patterns, a number equal to the number of clusters deleted in step 4, from the training data set of each target output class. Then go to step 2.

Data Mining using HGF. One instinctive ability of HGF algorithm is classification. The authors of [74] used HGF to deal with four artificial and twelve real-life data sets, such as iris data, wine recognition data, and Haberman's survival data. Compared with other well-known classification methods, experimental results showed that HGF performs very well in all problems in terms of the classification accuracy and the size of the model.

3.12 *Photosynthetic Algorithm (PA)*

Motivated by the principle of Benson-Calvin cycle Phase-1 and the reaction that happens in the chloroplast subcellular compartment for photorespiration, photosynthetic algorithm (PA) was originally proposed in [75]. To perform the PA, the following calculation processes need to be followed [75]:

- First, randomly generating the intensity of light.
- Second, evaluating the fixation rate of CO_2 via the following equation (also refer to as the stimulation function in the PA algorithm) based on the light intensity [75]. This is a unique characteristic of the PA algorithm. Such stimulation often happens as a result of randomly changed light intensity which in turn adjusts the influential degree on the elements of RuBP (i.e., ribulose-1, 5-bishosphate [76]) by photorespiration.

$$C = \frac{V_{\max}}{1 + A/L}. \tag{52}$$

  where the CO_2 fixation rate is denoted by C, $V_{\max}$ represents the maximum CO_2 fixation rate, A stands for the affinity of CO_2, and L is used to express the light intensity.
- Third, based on the fixation rate obtained from the stage above, one of two cycles, either Benson-Calvin or photorespiration will be selected at this stage. For both cycles, Murase [75] utilized 16-bit strings which shuffles based on carbon molecules recombination rule in photosynthetic pathways.
- Then after certain rounds of iterations, an amount of GAPs, i.e., glyceraldehyde-3-phosphate [77], are generated for representing intermediate knowledge strings in the PA algorithm. Each GAP is composed of 16 bits. The fitness of these GAPs will be evaluated at this stage. The best fit GAP will remain as a DHAP (i.e., di-hydroxyacetone phosphate [76]) which is referred to as the current estimated value.

Taking into account the fundamental process described above, the steps of implementing PA can be summarized as follows [75, 78, 79]:

- Step 1: Initializing the following problem parameters such as $f(x)$ (the object function), x_i (the decision variable), N (the number of decision variables), and the boundary of constraints.
- Step 2: Initializing the following problem parameters such as DHAPs, and CO_2 fixation parameters (e.g., affinity A, maximum fixation rate $V_{\max}$, and light intensity L).
- Step 3: Calculating CO_2 concentration, determining O_2/CO_2 concentration ration, and setting Benson-Calvin/photorespiration frequency ratio.
- Step 4: Evaluating if the stopping criteria are met. If yes, the algorithm stops; otherwise, go to the next step.
- Step 5: Depending the fixation rate of CO_2, the 16-bit strings are shuffled in either Benson-Calvin or photorespiration cycle.
- Step 6: Comparing the fitness value where the poor results will be removed and the desired DHAP strings and results will be remained.
- Step 7: Updating the light intensity and the next round of iteration of the PA algorithm starts.

Data Mining using PA. In order to verify the proposed PA, the finite element inverse analysis problem was employed in [75]. The prediction of the elastic moduli of the finite element model via PA was quite satisfied. The overall performance demonstrated by this preliminary application make PA a very attractive optimization algorithm. In data mining domain, the author of [78] used PA to solve multiple sequence alignment and association rules mining problem within biomedical data. Computational results showed that PA is capable of finding an effective global optima.

3.13 Stem Cells Optimization Algorithm (SCOA)

Stem cells optimization algorithm (SCOA) was originally proposed in [80, 81]. To perform the SCOA algorithm, the following procedure needs to be followed [80]:

- First, dividing the problem space into sections. The process can be accomplished totally in a random manner.
- Second, generating the initial population randomly and uniformly distributed in the whole search space of the target problem. At this stage, similar to most optimization algorithms, a variable matrix needs to be established for the purpose of obtaining a feedback with respect to problem variables. In SCOA, the key stem cells features are used to form the initial variable matrix. Such features may include liver cells, intestinal cells, blood cells, neurons, heart muscle cells, pancreatic islets cells, etc. Basically, the initial matrix can be express as equation below [80]:

$$\text{Population} = \begin{bmatrix} X_1 \\ X_2 \\ \cdots \\ X_N \end{bmatrix}. \tag{53}$$

  where $X_i = \text{Stem Cells} = [SC_1, SC_2, \ldots, SC_N]$; $i = 1, 2, \ldots, N$.
  In SCOA, some initialized parameters are defined as follows: M represents the maximum of stem cells; P stands for population size $(10 < P \leq M)$; $C_{Optimum}$ indicates the best of stem cell in each iteration; χ denotes the penalty parameter which is used to stop the growth of stem cell; and sc^i is the ith stem cell in the population.
- Third, the cost of each stem cell is obtained a criterion function which is determined based on the nature of the target problem. In SCOA, two types of memory, namely, local- and global-memory, are defined for each cell in which the local-memory is used to store the cost of each stem cell, and the global-memory stores the best cost among all locally stored cost values.
- Then, a self-renewal process will be performed which involves only the best cells of each area. At this stage, the information of each area's best cells will be shared and the cell that possesses the best cost will thus be chosen. In SCOA,

such cell is designed to play a more important role than other cells. Briefly, the stem cells' self-renewal operation is computed through equation below [80]:

$$SC_{Optimum}(t+1) = \zeta SC_{Optimum}(t). \tag{54}$$

where the iteration number is denoted by t, $SC_{Optimum}$ represents the best stem cell found in each iteration, and ζ is a random number which falls within [0, 1].

- Next, the above mentioned procedure will continue until the SCOA arrives at the goal of getting the best cell while keeping the value of cost function as low as possible. This is acquired via equation below [80]:

$$x_{ij}(t+1) = \mu_{ij} + \varphi\left(\mu_{ij}(t) - \mu_{kj}(t)\right). \tag{55}$$

where the ith stem cell position for the solution dimension j is represented by x_{ij}, the iteration number is denoted by t, two randomly selected stem cells for the solution dimension j are denoted by μ_{ij} and μ_{kj}, respectively, and $\varphi(\tau)$ (if $\mu_{ij}(t) - \mu_{kj}(t) = \tau$) generates a random variable falls within $[-1, 1]$.

- Finally, the best stem cell is selected when it has the most power relative to other cells. The comparative power can be computed via equation below [80]:

$$\varsigma = \frac{SC_{Optimum}}{\sum_{i=1}^{N} SC_i}. \tag{56}$$

where ς stands for stem cells' comparative power, $SC_{Optimum}$ denotes the stem cells selected in terms of cost, and N represents the final population size, i.e., when the best solution is obtained and the SCOA algorithm terminates.

Data Mining using SCOA. The authors of [80] employed SCOA for data clustering. A set of well-known datasets have been used to test the performance. Compared with other methods such as artificial bee colony, PSO, and ACO, experimental results showed that SCOA demonstrates superior clustering performance in terms of accuracy and high speed.

3.14 Wasp Swarm Optimization (WSO) Algorithm

Wasp swarm optimization (WSO) algorithm was originally proposed in [82] that is based on some behaviours found in wasp colony [83, 84]. The basic idea of WSO was to mimic a wasp colony behaviour, in particular according to the importance of individual wasp to the whole colony, assigning the resources to different wasp [82, 85]. Therefore in WSO algorithm, resources will be allocated to individual candidate solutions and such allocation is completed in a randomly manner where the strength of each option controls its chosen probability. In [86], a tournament process was utilized to implement this stochastic selection process: The weakest option (for example a) challenges the second weakest option (for instance b) and the winning probability of a over b is determined through $p_{ab} = s_a^2 / \left(s_a^2 + s_b^2 \right)$. The winner of this challenge (say a) will carry on to challenges the third weakest option (denoted by c), and wins with a probability of $p_{ac} = s_a^2 / \left(s_a^2 + s_c^2 \right)$. The challenge will continue until the final winner is selected. In some situations, it is more convenient to allocate costs instead of strengths to the individual wasps, i.e., the lower the cost, the higher the strength of a wasp. In this case, the winning probability of wasp i over j can be defined via equation below [86]:

$$p_{ij} = \frac{s_j^2}{s_i^2 + s_j^2}, \; i, j = 1, \ldots, c. \tag{57}$$

Data Mining using WSO. In [87], the author used WSO to optimize the C-means clustering model. Compared with other algorithms such as ACO, WSO outperforms ACO in terms of robust and high speed.

4 Discussions

Like other essential techniques of data qualify such as capture, communicate, aggregate and store, it is increasingly the case that much of data mining techniques simply could not take place without innovation. A conceptual view of big data challenges is provided by [4]. In the proposed framework, there are three challenges related to big data management operations:

- Data-stream processing and actual computing procedures.
- Different semantics and domain knowledge for big data applications.
- Algorithms for creating, clustering, and analyzing big data.

Over the past decade, data mining based upon CI approaches has been a widely studied research topic, being capable of meeting the ever-increasing demand of reliable and intelligent data mining techniques.

In our view, the methods introduced in this study are all developed with their inspiring source more or less coming from our Mother Nature. Through the intelligence introduced via the different metaphor mechanisms, these clever algorithms can speculate the intrinsic patterns from data sets without or with limited prior knowledge of regularities that might existing in the data. Such speculation is normally realized through either learning or searching process.

5 Future Work

The present work has some limitations. Firstly, a widespread literature review of the applications of innovative CI presents a difficult task, because of the extensive background knowledge that is required during the process of collection, study, and classification of these publications. Although acknowledging a limited background knowledge, this research makes a brief overview of literature concerned with using biology-based innovative CI algorithms to deal with data mining. Secondly, there are also some algorithms that falls under the other categories such as physics-, chemistry-, mathematics-based innovative CI. However, the present study does not take them into account. This would be an immediately research direction that need to be considered in future study.

6 Conclusion

Ubiquitous robotics systems has grown very fast in recent years. This is a good news from many perspectives, in particular considering the huge challenges confronted with the rapidly aging population. However, this remarkable growth also comes with some critical technical challenges. Among others, a lack of knowledge on how to manage the constant flow of data is probably a daunting disadvantages. As a result, data mining techniques are gradually being introduced into ambient intelligent system as the key to better understanding user behaviour and as a layer of intelligence for their own manipulation.

Nowadays, compared to the standard data management, big data, which is generated by multiple velocity sources and volumes, poses a considerable threat to a successful ambient intelligent system's implementation. To effectively and efficiently analyse the ever-increasing massive amount of data, on one hand, we can enhance the capability of the existing data mining techniques; while on the other hand, developing more powerful versions of data mining algorithms is also becoming extremely important, in particular, when many of the traditional approaches are reaching the limit of their full potential.

By dedicating to the latter trend, this study presents an up-to-date overview regarding the development of innovative CI algorithms and their corresponding applications in data mining field. Through our investigation, although the preliminary practice of these approaches cannot yet reveal that the traditional data mining techniques are overall outperformed by their newly introduced counterparts, these recently joined CI family members do show various attractive and promising advantages in dealing with data mining. Therefore it is the author's hope that this survey will inspire other far better qualified researchers to bring these algorithms to their full potential for embracing the forthcoming ambient intelligence era.

References

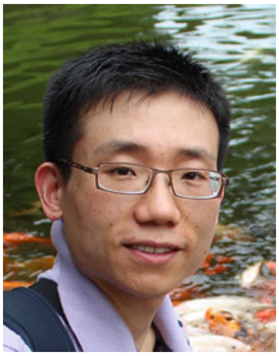
1. Manyika, J., et al.: Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute (2011)
2. Gobble, M.M.: Big data: the next big thing in innovation. Res. Technol. Manage. **56**(1), 64–66 (2013)
3. Katal, A., Wazid, M., Goudar, R.H.: Big data: issues, challenges, tools and good practices. In: 2013 Sixth International Conference on Contemporary Computing (IC3), pp. 404–409. IEEE (2013)
4. Wu, X., et al.: Data mining with big data. IEEE Trans. Knowl. Data Eng. **26**(1), 97–107 (2014)
5. Nugent, C.D., et al.: Managing sensor data in ambient assisted living. J. Comput. Sci. Eng. **5** (3), 237–245 (2011)
6. Yu, B., Sen, R., Jeong, D.H.: An integrated framework for managing sensor data uncertainty using cloud computing. Inf. Syst. **38**, 1252–1268 (2013)
7. Choudhary, A.K., Harding, J.A., Tiwari, M.K.: Data mining in manufacturing: a review based on the kind of knowledge. Int. J. Intell. Manuf. **20**, 501–521 (2009)
8. Ahmed, S.R.: Applications of data mining in retail business. Inf. Technol.: Coding Comput. **2**, 455–459 (2004)
9. Liao, S.-H., Chu, P.-H., Hsiao, P.-Y.: Data mining techniques and applications—a decade review from 2000 to 2011. Expert Syst. Appl. **39**, 11303–11311 (2012)
10. Liao, S.-H., Chen, Y.-J., Lin, Y.-T.: Mining customer knowledge to implement online shopping and home delivery for hypermarkets. Expert Syst. Appl. **38**, 3982–3991 (2011)
11. Ngai, E.W.T., Xiu, L., Chau, D.C.K.: Application of data mining techniques in customer relationship management: a literature review and classification. Expert Syst. Appl. **36**, 2592–2602 (2009)
12. Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques. 3rd ed. 2012, 225 Wyman Street, Waltham, MA 02451, Morgan Kaufmann, Elsevier Inc., USA. ISBN 978-0-12-381479-1 (2012)
13. Xing, B., Gao, W.-J.: Innovative computational intelligence: a rough guide to 134 clever algorithms. Springer International Publishing Switzerland, Cham, Heidelberg, New York, Dordrecht, London (2014). ISBN 978-3-319-03403-4
14. Elalfi, E., Haque, R., Elalami, M.E.: Extracting rules from trained neural network using GA for managing E-business. Appl. Soft. Comput. **4**, 65–77 (2004)
15. Xie, L., Mei, H.: The application of the ant colony decision rule algorithm on distributed data mining. Commun IIMA **7**(4), 85–94 (2007)
16. Sinha, A.N., Das, N., Sahoo, G.: Ant colony based hybrid optimization for data clustering. Kybernetes **36**(2), 175–191 (2007)
17. de Castro, L.N., Zuben, F.J.V.: aiNet: an artificial immune network for data analysis. In: Abbass, H.A., Sarker, R.A., Newton, C.S (ed.) Data mining: a heuristic approach, Idea Group Publishing, (2001)
18. Alatas, B., Akin, E.: Multi-objective rule mining using a chaotic particle swarm optimization algorithm. Knowl.-Based Syst. **22**, 455–460 (2009)
19. Mitra, S., Pal, S.K., Mitra, P.: Data mining in soft computing framework: a survey. IEEE Trans. Neural Netw. **13**(1), 3–14 (2002)
20. Romero, C. Ventura, S.: Educational data mining: a survey from 1995 to 2005. Expert Syst. Appl. **33**, 135–146 (2007)
21. Harding, J.A., et al.: Data mining in manufacturing: a review. J. Manuf. Sci. Eng. **128**, 969–976 (2006)
22. Glaser, R.: Biophysics: an introduction, 2nd edn. Springer, Berlin, Heidelberg (2012). ISBN 978-3-642-25211-2

23. Floreano, D., Mattiussi, C.: Bio-inspired artificial intelligence: theories, methods, and technologies. In: Arkin, R.C. (ed.) Intelligent Robotics and Autonomous Agents 2008, The MIT Press, Cambridge, Massachusetts. ISBN 978-0-262-06271-8 (2008)
24. Li, X-l: A new intelligent optimization method—artificial fish school algorithm (in Chinese with English abstract), in Institute of Systems Engineering. Zhejiang University, Hangzhou, P. R. China (2003)
25. Zhang, M., et al.: Evolving neural network classifiers and feature subset using artificial fish swarm. In: IEEE International Conference on Mechatronics and Automation 25–28 June, Luoyang, China, pp. 1598–1602. IEEE (2006)
26. Wang, C.-R., Zhou, C.-L., Ma J.-W.: An improved artificial fish-swarm algorithm and its application in feed-forward neural networks. In: Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, China, 18–21 Aug, pp. 2890–2894. (2005)
27. Luo, Y., Zhang, J., Li, X.: The optimization of PID controller parameters based on artificial fish swarm algorithm. In: IEEE International Conference on Automation and Logistics, 18–21 Aug, Jinan, China, pp. 1058–1062. (2007)
28. Neshat, M., et al.: A review of artificial fish swarm optimization methods and applications. Int. J. Smart Sens. Intell. Syst. **5**(1), 107–148 (2012)
29. Zhang, M., et al.: Mining classification rule with artificial fish swarm. In: 6th World Congress on Intelligent Control and Automation, 21–23 June, Dalian, China, pp. 5877–5881 (2006)
30. Wang, F., Xu, X., Zhang, J.: Application of artificial fish school and K-means clustering algorithms for stochastic GHP. In: Control and Decision Conference (CCDC), pp. 4280–4283. 2009
31. Neshat, M., et al.: A new hybrid algorithm based on artificial fishes swarm optimization and k-means for cluster analysis. Int. J. Comput. Sci. Issues **8**(4), 251–259 (2011)
32. Sun, S., Zhang, J., Liu, H.: Key frame extraction based on artificial fish swarm algorithm and k-means. In: International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), 16–18 Dec, Changchun, China, pp. 1650–1653. IEEE (2011)
33. He, S., et al.: Fuzzy clustering with improved artificial fish swarm algorithm. In: International Joint Conference on Computational Sciences and Optimization (CSO), pp. 317–321. IEEE (2009)
34. Cheng, Y., Jiang, M., Yuan, D.: Novel clustering algorithms based on improved artificial fish swarm algorithm. In: Sixth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 141–145. IEEE (2009)
35. Zhu, W., et al.: Clustering algorithm based on fuzzy C-means and artificial fish swarm. Procedia Eng. **29**, 3307–3311 (2012)
36. Su, M.-C., Su, S.-Y., Zhao, Y.-X.: A swarm-inspired projection algorithm. Pattern Recogn. **42**, 2764–2786 (2009)
37. Yang, X.-S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
38. Yang, X.-S.: Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, UK (2008). ISBN 978-1-905986-28-6
39. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: Watanabe, O., Zeugmann, T (ed.) SAGA 2009, LNCS 5792, pp. 169–178, Springer, Berlin, Heidelberg (2009)
40. Babu, B.G., Kannan, M.: Lightning bugs. Resonance **7**(9), 49–55 (2002)
41. Jones, K.O., Boizanté, G.: Comparison of firefly algorithm optimisation, particle swarm optimisation and differential evolution. In: International Conference on Computer Systems and Technologies (CompSysTech), 16–17 June, Vienna, Austria, pp. 191–197. (2011)
42. Senthilnath, J., Omkar, S.N., Mani, V.: Clustering using firefly algorithm: performance study. Swarm and Evol. Comput. **1**, 164–171 (2011)
43. Lancaster, R., et al.: Fireworks: principles and practice. Chemical Publishing Co., Inc., New York. ISBN 0-8206-0354-6 (1998)
44. Tan, Y. Zhu, Y.: Fireworks algorithm for optimization. In: Tan, Y., Shi, Y., Tan, K.C. (ed.) ICSI 2010, Part I, LNCS 6145, pp. 355–364, Springer, Berlin, Heidelberg (2010)

45. Janecek, A., Tan, Y.: Swarm intelligence for non-negative matrix factorization. Int. J. Swarm Intell. Res. **2**(4), 12–34 (2011)
46. Pei, Y., et al.: An empirical study on influence of approximation approaches on enhancing fireworks algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2012), Seoul, Korea, 14–17 Oct, pp. 1322–1327. IEEE (2012)
47. Hersovici, M., et al.: The shark-search algorithm. An application: tailored Web site mapping. Comput. Netw. ISDN Syst. **30**, 317–326 (1998)
48. Hillis, K., Petit, M., Jarrett, K.: Google and the culture of search. Taylor & Francis, 711 Third Avenue, New York, NY: Routledge (2013). ISBN 978-0-415-88300-9
49. Cho, J., Garcia-Molina, H., Page, L.: Efficient crawling through URL ordering. Comput. Netw. ISDN Syst. **30**, 161–172 (1998)
50. Jarvis, J.: What whould Google do?. HarperCollins Publishers Ltd., 55 Avenue Road, Suite 2900, Toronto, ON, M5R, 3L2, Canada (2009). ISBN 978-0-06-176472-1
51. Luo, F.-f., Chen, G.-l., Guo W.-z.: An improved fish-search algorithm for information retrieval. In: IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE), pp. 523–528. IEEE (2005)
52. Chen, Z., et al.: An improved shark-search algorithm based on multi-information. In: Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 1–5. (2007)
53. Sun, T., et al.: Airplane route planning for plane-missile cooperation using improved fish-search algorithm. In: International Joint Conference on Artificial Intelligence (JCAI), pp. 853–856. (2009)
54. Bellaachia, A. Bari, A.: Flock by leader: a novel machine learning biologically inspired clustering algorithm. In: Tan, Y., Shi, Y., Ji, Z (ed.) ICSI 2012, Part I, LNCS 7332, pp. 117–126, Springer, Berlin, Heidelberg (2012)
55. Reynolds, C.W.: Flocks, herds, and schools: a distributed behavioral model. Comput. Graph. **21**(4), 25–34 (1987)
56. Cui, X., Gao, J., Potok, T.E.: A flocking based algorithm for document clustering analysis. J. Syst. Architect. **52**, 505–515 (2006)
57. Picarougne, F., et al.: A new approach of data clustering using a flock of agents. Evol. Comput. **15**(3), 345–367 (2007)
58. Luo, X., Li, S., Guan, X.: Flocking algorithm with multi-target tracking for multi-agent systems. Pattern Recogn. Lett. **31**, 800–805 (2010)
59. Pan, W.-T.: A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl.-Based Syst. **26**, 69–74 (2012)
60. Pan, W.-T.: Fruit fly optimization algorithm (in Chinese). Tsang Hai Book Publishing Co. ISBN 978-986-6184-70-3 (2011)
61. Krishnanand, K.N. Ghose, D:. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In: IEEE Swarm Intelligence Symposium (SIS), IEEE, pp. 84–91 (2005)
62. Krishnanand, K.N., Ghose, D.: Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intell. **3**, 87–124 (2009)
63. Huang, Z., Zhou, Y.: Using glowworm swarm optimization algorithm for clustering analysis. J. Convergence Inf. Technol. **6**(2), 78–85 (2011)
64. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)
65. Geem, Z.W.: Particle-swarm harmony search for water network design. Eng. Optim. **41**(4), 297–311 (2009)
66. Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput. Methods Appl. Mech. Eng. **194**, 3902–3933 (2005)
67. Mahdavi, M., Abolhassani, H.: Harmony K-means algorithm for document clusterin. Data Min. Knowl. Disc. **18**, 370–391 (2009)

68. Mahdavi, M., et al.: Novel meta-heuristic algorithms for clustering web documents. Appl. Math. Comput. **201**, 441–451 (2008)
69. Cobos, C., et al.: Web document clustering based on global-best harmony search, k-means, frequent term sets and Bayesian information crite. In: Proceedings of the IEEE World Congress on Computational lnteliigence (WCCI), 18–23 July 2010, CCIB, Barcelona, Spain, IEEE, pp. 4637–4644. 2010
70. Moeinzadeh, H., et al.: Combination of harmony search and linear discriminate analysis to improve classification. In: Proceedings of the Third Asia International Conference on Modelling & Simulation (AMS), IEEE, pp. 131–135 (2009)
71. Wang, X., Gao, X.-Z., Ovaska, S.J.: Fusion of clonal selection algorithm and harmony search method in optimization of fuzzy classification systems. Int. J. Bio-Inspired Comput. **1**, 80–88 (2009)
72. Venkatesh, S.K., Srikant, R., Madhu, R.M.: Feature selection & dominant feature selection for product reviews using meta-heuristic algorithms. In: Proceedings of the Compute'10, 22–23 Jan 2010, Bangalore, Karnataka, India, pp. 1–4. ACM (2010)
73. Ramos, C.C.O., et al.: A novel algorithm for feature selection using harmony search and its application for non-technical losses detection. Comput. Electr. Eng. **37**, 886–894 (2011)
74. Thammano, A., Moolwong, J.: A new computational intelligence technique based on human group formation. Expert Syst. Appl. **37**, 1628–1634 (2010)
75. Murase, H.: Finite element inverse analysis using a photosynthetic algorithm. Comput. Electron. Agric. **29**, 115–123 (2000)
76. Carpentier, R. (ed.) Photosynthesis research protocols. In: Walker, J.M., (ed.) 2nd edn. Methods in Molecular Biology, 2011, Springer Science+Business Media, LLC, ISBN 978-1-60671-924-6. Spring, New York, NY 10013, USA (2011)
77. Dubinsky, Z. (ed).: Photosynthesis. InTech, ISBN 978-953-51-1161-0: Janeza Trdine 9, 51000 Rijeka, Croatia (2013)
78. Alatas, B.: Photosynthetic algorithm approaches for bioinformatics. Expert Syst. Appl. **38**, 10541–10546 (2011)
79. Yang, X.-S.: Biology-derived algorithms in engineering optimization. In: Olarius, S., Zomaya, A. (ed.) Handbook of Bioinspired Algorithms and Applications, Chapter 32, pp. 585–596, CRC Press, LLC (2005)
80. Taherdangkoo, M., Yazdi, M., Bagheri, M.H.: A powerful and efficient evolutionary optimization algorithm based on stem cells algorithm for data clustering. Cent. Eur. J. Comput. Sci. **2**(1), 1–13 (2012)
81. Taherdangkoo, M., Yazdi, M., Bagheri, M.H: Stem cells optimization algorithm, in LNBI 6840, pp. 394–403. Springer, Berlin, Heidelberg (2011)
82. Theraulaz, G., et al.: Task differentiation in polistes wasps colonies: a model for self-organizing groups of robots. In: First International Conference on Simulation of Adaptive Behavior, MIT Press, pp. 346–355 (1991)
83. Karsai, I., Wenzel, J.W.: Organization and regulation of nest construction behavior in Metapolybia wasps. J. Insect Behav. **13**(1), 111–140 (2000)
84. Lucchetta, P., et al.: Foraging and associative learning of visual signals in a parasitic wasp. Anim. Cogn. **11**(3), 525–533 (2008)
85. Fan, H., Zhong, Y.: A rough set approach to feature selection based on wasp swarm optimization. J. Comput. Inf. Syst. **8**(3), 1037–1045 (2012)
86. Cicirello, V.A., Smith, S.F.: Wasp-like agents for distributed factory coordination. Auton. Agent. Multi-Agent Syst. **8**, 237–266 (2004)
87. Runkler, T.A.: Wasp swarm optimization of the c-means clustering model. Int. J. Intell. Syst. **23**, 269–285 (2008)

Author Biography



Bo Xing, DIng, is an Associate Professor at the Department of Computer Science, School of Mathematical and Computer Science, Faculty of Science and Agriculture, University of Limpopo, South Africa. He was a senior lecturer under the division of Center for Asset Integrity Management (C-AIM) at the Department of Mechanical and Aeronautic Engineering, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, South Africa. Dr. Xing earned his DIng degree (Doctorate in Engineering with a focus on soft computing and remanufacturing) in the early 2013 from the University of Johannesburg, South Africa. He also obtained his B.Sc. and M.Sc. degree both in Mechanical Engineering from the Tianjin University of Science and Technology, P.R. China, and the University of KwaZulu-Natal, South Africa, respectively. He was a scientific researcher at the Council for Scientific and Industrial Research (CSIR), South Africa. He has published more than 50 research papers in books, international journals, and international conference proceedings. His current research interests lie in applying various nature-inspired computational intelligence methodologies towards big data analysis, miniature robot design and analysis, advanced mechatronics system, and e-maintenance.