Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 9457821, 16 pages
https://doi.org/10.1155/2018/9457821

Research Article

Bioinspired Computational Approach to Missing Value Estimation

Israel Edem Agbehadji,[1] Richard C. Millham,[1] Simon James Fong,[2] and Hongji Yang[3]

[1]*ICT and Society Research Group, Department of Information Technology, Durban University of Technology, Durban, South Africa*
[2]*Department of Computer and Information Science, University of Macau, Taipa, Macau*
[3]*Department of Computer and Information Science, Bath Spa University, Bath, UK*

Correspondence should be addressed to Richard C. Millham; richardmillham@hotmail.com

Received 26 June 2017; Revised 16 November 2017; Accepted 13 December 2017; Published 2 January 2018

Academic Editor: Erik Cuevas

Missing data occurs when values of variables in a dataset are not stored. Estimating these missing values is a significant step during the data cleansing phase of a big data management approach. The reason of missing data may be due to nonresponse or omitted entries. If these missing data are not handled properly, this may create inaccurate results during data analysis. Although a traditional method such as maximum likelihood method extrapolates missing values, this paper proposes a bioinspired method based on the behavior of birds, specifically the Kestrel bird. This paper describes the behavior and characteristics of the Kestrel bird, a bioinspired approach, in modeling an algorithm to estimate missing values. The proposed algorithm (KSA) was compared with WSAMP, Firefly, and BAT algorithm. The results were evaluated using the mean of absolute error (MAE). A statistical test (Wilcoxon signed-rank test and Friedman test) was conducted to test the performance of the algorithms. The results of Wilcoxon test indicate that time does not have a significant effect on the performance, and the quality of estimation between the paired algorithms was significant; the results of Friedman test ranked KSA as the best evolutionary algorithm.

1. Introduction

The concept of big data is defined using several characteristics including velocity, volume, and value. The characteristics of velocity are related to how fast incoming data need to be processed and how quickly the receiver of information needs the results from the processing system [1]; the characteristics of volume are related to the amount of data that has to be processed; and the characteristics of value are what the user of big data management will gain from the data analysis. Other characteristics of big data include variety and veracity. The characteristics of variety focus on the different structures that data may take, such as text and images, while the characteristics of veracity focus on authenticity of the data source that is being used for decision-making. These characteristics of big data have resulted in the use of innovative methods for decision-making. These innovative methods may require the combination of different technological platforms for storage, such as Hadoop, NoSQL, and relational databases, to successfully manage this big data. It is possible to have datasets on these platforms with missing values at random, which are a result of mismatched attributes [2] or omitted entries [3]. Hence, missing data is independent of the type of platform on which this data is placed. There are three categories of missing data: data missing completely at random (MCAR), data missing at random (MAR), and data missing not at random (MNAR) [4, 5], which has a different method to handle the missing data.

The missing data category of missing completely at random (MCAR) occurs when the missing values are randomly distributed throughout a matrix such that a missing value in a row of a matrix is not dependent on any other row entry in a dataset [4]. In other words, neither the row entry which is missing nor any other row entry can predict whether a value is missing. When this happens, the chances of the data being detected as missing are not dependent on either the missing or the complete value in the same row entry of a matrix. The listwise method to handle MCAR is ideally used to remove

all data that has one or more missing cases; however, by this removal, a problem is created in that the missing values produce both biased parameters and incorrect estimates in analysis. While the pairwise method is also another method to handle MCAR, this method sought to address the missing value problem by computing the covariance estimates from all samples of cases observed on variables. The pairwise deletion method assumes that all data is completely missing at random; therefore, variables with missing data are then deleted during computation. This deletion could cause error in computation because each element in the covariance matrix may have a different group of attributes [6, 7].

Missing at random (MAR) category occurs when the missing value in a row of a matrix depends on another known row entry in a dataset. In other words, the missing value can be predicted from a previously known value in a dataset. Thus, the missing value is dependent on the previously known value. When this happens, it becomes easy to trace a pattern on a missing value in a row of a matrix. The traditional approach to handle MAR is pairwise deletion method as described previously.

Missing not at random (MNAR) (also known as non-ignorable nonresponse) category occurs when the missing value in a row of a matrix depends on the other missing values in the row entry. When this happens, the known data cannot be used to estimate the missing value. Thus, the chances that the current value is detected as missing are dependent on the detection of previously missing values.

These traditional approaches to handle missing data such as listwise deletion or case deletion, pairwise deletion, and also sample mean substitution (i.e., k-NN and k-means clustering [4, 8]) are, however, not efficient at providing the best optimal estimates for missing values. For instance, the sample mean substitution method requires that each data point clustered around a centroid be computed so as to find the best estimates. Thus, the number of clusters, the number of data points, and the dimensions involved to compute missing values make it inefficient. With the pairwise deletion, since the method assumes all data is missing at random, it then uses the average sample size to estimate its standard error, which results in either underestimation or overestimation of the standard error in the analysis of missing values, and this makes it inefficient. In view of this, other efficient methods such as maximum likelihood [9] and multiple imputation method (for MAR), expectation-maximization (EM) algorithm [4, 10] and machine learning approach (such as autoencoder neural network) and meta-heuristic algorithms, such as genetic algorithms, [11] have been proposed to handle missing values at random.

The maximum likelihood method is a method for estimating missing values by selecting a set of parameters or values that maximizes a likelihood function. The advantage of the maximum likelihood method is its consistency and unbiased estimation of the parameter closest to the observed value [12]. The expectation-maximization algorithm uses the maximum likelihood method to impute all missing values in a dataset [4]. This procedure in finding missing values uses probability to iteratively impute values in its estimation of an approximate parameter that becomes closer to the missing value [10]. This iterative process generates a weighted value that is improved in each iteration until a termination condition is reached. Additionally, when there are many variables and multiple missing values, then the computational time increases at each iteration. On the other hand, the autoencoder neural network or the autoassociative multilayer perceptron method consists of an input and output layer where the number of inputs is equal to the number of outputs [13]. This network, when used to estimate approximate missing values, uses activation functions to map sparse input space through hidden units to output space. In other words, this activation function is used as a function to control the input data from a dataset. During the estimation process, weighted parameters are used in the hidden unit of the network. This weight parameter is solved iteratively by maximizing the probability of the weight parameter in the hidden unit to produce the best value that is close to the missing value [5]. The advantage of the autoencoder neural network is that it gives reliable estimates as missing values; however, its efficiency depends on the number of hidden layers chosen, and the higher the number of hidden layers, the more the computational time required to estimate the missing value(s).

Genetic algorithms are an evolutionary approach which is based on the survival of the fittest. This survival depends on the mechanism of "natural selection" (Darwin, 1868, as cited in [14]) where element is represented using a binary string. A genetic algorithm is an adaptive search procedure [15], as cited by [14], which involves the use of operators such as crossover, mutation, and selection methods to find a global optimal result/solution. The search procedure starts with an initial guess and attempts to improve the guess through evolution [14] by comparing the fitness of the initial generation of population with the fitness obtained after application of operators to the current population until the final optimal value is produced. This adaptive search procedure is an iterative process that allows the elimination of weak individuals of a population through a continuous update of the initial generation of population via multiple generations until the termination condition is reached. The adaptive search procedure helps to find approximate missing values [16] by optimizing an objective function/fitness function in any given search problem.

Particle swarm is a bioinspired method that is based on the swarm behavior of fish schools and bird flocks in nature [17]. The swarm behavior is expressed in terms of how particles adapt and make decisions on change of position within a space based on the position of other neighboring particles. The advantage of swarm behavior is that as an individual particle makes a decision, it leads to an emergent behavior [18]. This emergent behavior is the result of local interaction among particles in a problem space. Among the particle swarm algorithms for finding the best possible solutions in a problem space are the Firefly algorithm [19], bats [20], and cuckoo birds [21]. The successful characteristic of fireflies is the short and rhythmic flashes they produce [19]. This flashing light is used as a mechanism to attract mating partners and attract potential prey, and it serves as a warning to other fireflies. The signaling system of this flashing light

mechanism is controlled by simplified basic rules underlining the behavior of fireflies. Unlike a genetic algorithm which uses operators such as mutation, crossover, and selection, the firefly uses attractiveness and brightness to improve certain individuals in its population. The similarity between the genetic algorithm and the firefly is that both generate an initial population and continue to update the initial population using fitness functions. The brighter fireflies attract those closest around them and the fireflies whose flashes fall below a given threshold are removed from the population, while the brightest fireflies form the next generation, and the generations/iterations continue until a select criterion is reached or the maximum number of generations is reached. The behavior of fireflies where a bright firefly attracts a firefly with a weaker brightness has been applied in missing data imputation by finding estimates of values closest to known values and then replacing these missing values with these estimates.

Wolf Search Algorithm (WSA) is a bioinspired heuristic optimization algorithm which is based on wolf preying behavior [22]. The behavior of a wolf includes its ability to hunt independently by remembering its own trait (meaning wolves have memory); ability to only merge with its peer when the peer is in a better position (meaning there is trust among wolves to never prey on each other); ability to escape randomly upon appearance of a hunter [22]; and the use of scent marks as a way of demarcating its territory and communicating with other wolves [23]. This behavior expressed by wolves enables them to randomly adapt to their environment when hunting. If a wolf finds a new better position, the incentive is stronger to assume this new position provided that the position is already inhabited by a companion wolf. The wolf search algorithm is an iterative search process that starts with the setting of the initial parameter, random initialization of population, evaluation and updating a current population using a fitness test, and continuing on with creating new generations/iterations until some stopping criterion is met. Unlike the genetic algorithm that uses operators such as mutation, crossover, and selection methods or particle swarm algorithm, such as firefly, that uses attractiveness and brightness of prey, the wolf uses attractiveness of prey within its visual range. Furthermore, each wolf instinctively flocks together in a pack, which is collective, and organizes individual searches of an individual wolf. Therefore, the swarming behavior of WSA is delegated to each individual wolf and this could form multiple leaders swarming from multiple directions towards the best solution rather than a single flock searching for an optimum in one direction at a time [22]. This swarm behavior of wolves could be used to estimate the approximate value close to known values in a missing value at a random situation.

A variation of WSA is the Wolf Search Algorithm with Step Minus Previous (WSAMP). This WSAMP allows wolves to remember a previous best position and avoid the old positions which do not produce the best solution.

BAT algorithm [24] is a bioinspired method based on the behavior of microbats in their natural environment. The unique behavior that characterizes bats is their echolocation mechanism. This mechanism helps bats orient and find prey within their environment. The search strategy of bats is controlled by the pulse rate and loudness of their echolocation mechanism. The change in the pulse rate helps to improve on the previous position, while the loudness alerts each other bat on the best position that has been found [25]. The bat behavior has been applied in several optimization problems to find the best optimal solution. The BAT algorithm search process starts with random initialization of the population, evaluation of the new population using a fitness function, and finding the best population. Unlike the wolf algorithm that uses attractiveness of prey to govern its search, the BAT algorithm uses the pulse rate and loudness to control the search for the optimal solution.

Bioinspired search strategies are controlled by randomization, efficient local search, and global best solution [24]. The contribution of this paper is that the random encircling behavior of certain birds that is required in achieving an optimal solution for missing values is first proposed as a new computational method and then examined in comparison with other metaheuristic algorithms such as Wolf Search Algorithm with a step Minus Previous (WSAMP), Firefly algorithm, and BAT algorithm. The advantage of random encircling is that it maximizes the search space, thus creating a wider range from a hovering position for the best possible solution. We also evaluated the quality of the proposed computational method, the random encircling of birds such as Kestrel, using a fitness function.

The remainder of this article is organized as follows. In Section 2, we describe the behavior of Kestrel birds. Section 3 discusses the proposed computational model. The model consists of mathematical formulations on the Kestrel's characteristics. Section 4 discusses the experimental results as well as comparisons of the proposed algorithm with the existing approach. Section 5 presents statistical analysis of experimental results. Section 6 contains conclusions and future work.

2. Description of the Behavior of Kestrel Birds

The bioinspired algorithm is based on the behavior of Kestrel birds when hunting for prey. The Kestrel is a kind of bird that hunts by hovering (i.e., flight-hunt) or from a perch. These birds are strongly territorial and hunt individually [26, 27]. Reference [27] indicated that, during a hunt, Kestrels are imitative rather than cooperative. This suggests that Kestrels prefer not to communicate with each other but rather they imitate the behavior of other Kestrels with better hunting techniques and improve their hunting technique even though the hunting technique can change based on the type of prey, prevailing weather conditions, and energy requirements (for gliding or diving) [28].

During hunting, Kestrels use their eyesight to watch small and agile prey within their circling radius or coverage area referred to as the visual circling radius. The minute air disturbance from flying prey and the trail of urine and faeces from ground prey give an indication of the availability of prey. Once available prey is detected using these indications, the Kestrel positions itself to hunt. Kestrels are able to hover in changing airstream, maintain a fixed forward looking position with their eyes on the prey, and use random bobbing of

the head to find the least distance between their position and the position of the prey. Also, the Kestrels possess excellent ultraviolet sensitive eyesight characteristics to visually locate trails because these trails of urine and faeces reflect ultraviolet light. Consequently, trails of prey such as voles become visible to Kestrels [29].

In hovering, Kestrels perform a wider search (global exploration) across territories within their visual circling radius, maintain a motionless position with a forward looking eye fixed on the prey, detect minute air disturbances from flying prey (particularly flying insects) to best position themselves to hunt the prey, and mostly move with precision through changing airstream.

Kestrels are able to flap their wings and adjust their long tails to stay in a place that is referred to as a still position in changing airstream. While in perch, mostly from high fixed structures, the Kestrel changes its perch every few minutes, performs a thorough search (a local exploitation using its individual hunt behavior) of its local territory with less energy requirements than a hovering hunt, and uses its ultraviolet sensitive capabilities to detect mammals such as voles closer to a perched area. This behavior suggests that, in perch, Kestrels conserve some of their energy and direct their ultraviolet sensitive capabilities to detect slowly moving prey on the ground. Moreover, an individual Kestrel with better perch and hovering skills in wider search area stands a better chance to move faster on its prey or disperse sooner from its enemy than individual Kestrels that develop hunting skills in local territories [27]. Therefore, it is significant to combine both types of hunting skills for a successful hunt. The characteristics of Kestrels are summarized as follows:

(1) Soaring: this gives a larger search space (global exploration) within the visual coverage area.

  (a) They maintain a still (motionless) position with eyesight fixed on the prey.
  (b) They encircle the prey beneath with keen eyesight.

(2) Perching: each Kestrel does a thorough search (local exploitation) within its visual coverage area.

  (a) They perform frequent bobbing of the head.
  (b) They get attracted to the prey using the detected visible trail and then glide to capture this prey.

The following assumptions are made on the characteristics:

(i) The still position gives a near-perfect circle, and thus frequent changes in a circle direction depend on the position of the prey in shifting the center of its circling direction.

(ii) Frequent bobbing of the head gives a degree of magnified or binocular vision that helps in measuring the distance to the prey, which then enables the Kestrel to move with a speed to strike.

(iii) Attractiveness is proportional to light reflection; thus, the higher or the longer the distance from the Kestrel to the trail, the less the trail brightness. This distance rule applies to both hovering height and distance away from perch.

(iv) New trails are more attractive than an old trail. Thus, trail decay or trail evaporation depends on the half-life of the trail.

3. The Proposed Computational Model

The proposed computational model for Kestrel's missing value estimation is based on the description of Kestrels' behavior and characteristics. The following mathematical expressions depict the characteristics of the Kestrel.

(i) Encircling. Encircling is when the Kestrel randomly shifts (or changes) the center of circling direction to recognize the current position of the prey. As the prey changes its current position, Kestrels randomly use the encircling behavior to encircle the prey. This movement of the prey determines the best possible position assumed by the Kestrel. The encircling $\overrightarrow{D}$ [30] is expressed as

$$\overrightarrow{D} = \left| \overrightarrow{C} * \overrightarrow{x_p}(t) - \overrightarrow{A} * \overrightarrow{x}(t) \right|. \tag{1}$$

Thus,

$$\overrightarrow{C} = 2 * \overrightarrow{r1}, \tag{2}$$

where $\overrightarrow{C}$ is the coefficient vector, $\overrightarrow{D}$ is the encircling value obtained to indicate best position, $\overrightarrow{x_p}(t)$ is the position vector of the prey, $\overrightarrow{A}$ is coefficient vector, and $\overrightarrow{x}(t)$ indicates the position vector of a Kestrel, and $r1$ and $r2$ are random numbers generated between 0 and 1.

(ii) Current Position. The current best position of the Kestrel is expressed as

$$\overrightarrow{x}(t+1) = \overrightarrow{x_p}(t) - \overrightarrow{A} * \overrightarrow{D}. \tag{3}$$

Thus,

$$\overrightarrow{A} = 2 * \overrightarrow{z} * \overrightarrow{r2} - \overrightarrow{z}, \tag{4}$$

where $\overrightarrow{A}$ is the coefficient vector, $\overrightarrow{D}$ is the encircling value obtained, $\overrightarrow{x_p}(t)$ is the position vector of the prey, and $\overrightarrow{x}(t+1)$ represents the current best position of Kestrels. $\overrightarrow{z}$ linearly decreases from 2 (upper bound value) to 0 (lower bound value) and it is used to control the randomness in iteration. $\overrightarrow{z}$ is expressed as follows:

$$\overrightarrow{z} = \overrightarrow{z}_{\text{hi}} - \left(\overrightarrow{z}_{\text{hi}} - \overrightarrow{z}_{\text{low}}\right) \frac{\text{itr}}{\text{Max\_itr}}, \tag{5}$$

where itr is the current iteration and Max_itr represents the maximum number of iterations to terminate the search. z_{hi} represents the higher bound value while z_{low} represents the lower bound value. Other Kestrels that are involved in the

search update their position according to the best position of the leading Kestrel. Also, the change in position of a Kestrel in airstream depends on the frequency of bobbing, attractiveness, and trail evaporation. This is expressed as follows.

(a) Frequency of Bobbing. The frequency of bobbing f is used for sight distance measurement in the search space. This is expressed as

$$f_{t+1}^{k} = f_{\min} + \left(f_{\max} - f_{\min}\right) * \alpha, \tag{6}$$

where $\alpha \in [0, 1]$ is a random number generated from lower and upper end points to control the frequency of bobbing within a visual range. $f_{\max}$ represents the maximum frequency and $f_{\min}$ is the minimum frequency both between 1 and 0, respectively.

(b) Attractiveness. Attractiveness β indicates the light reflected from a trail, which is defined by

$$\beta(r) = \beta_o e^{-\gamma r^2}, \tag{7}$$

where β_o represents the attractiveness, γ represents variation of light intensity in the range $[0, 1]$, and r represents the sight distance $s(x_i, x_c)$ measurement which is expressed using Minkowski distance formulation as follows:

$$s\left(x_i, x_c\right) = \left(\sum_{k=1}^{n} \left|x_{i,k} - x_{c,k}\right|^{\lambda}\right)^{1/\lambda}. \tag{8}$$

Thus,

$$V \leq s\left(x_i, x_c\right), \tag{9}$$

where x_i is the current sight measurement, x_c are all potential neighboring sight measurements near x_i, n is the total number of neighboring sights, λ is the order (1 or 2), and V is the visual range.

(c) Trail Evaporation. A definition of a trail is the formation and maintenance of a line [31]. In metaheuristic algorithms, ants use trails both to trace the path to a food source and to prevent themselves from getting stuck in a single food source.Thus, ants, using these trails, can search many food sources in a search space [14]. As ants continue to search, trails are drawn and substances are deposited in the trail. These substances help ants to communicate with each other about the location of food sources. Therefore, other ants continuously follow this path and also deposit substances for the trail to remain fresh. Similar to ants, Kestrels use trails in search of food sources. However, these trails are rather deposited by prey, which provides an indication to Kestrels on the availability of food sources. The assumption is that the substances deposited by these types of prey are similar to substances deposited on ants' trails. Additionally, when the source of food depletes, Kestrels no longer follow this path. Consequently, the trail substance begins to diminish with time at an exponential rate causing trails to become old. This diminishment denotes the unstable nature of the trail substances which can be theoretically stated as follows: if there are N unstable elements with an exponential decay rate γ, then an equation can be formulated to describe how N substance decreases in time t [32]. This equation is expressed as follows:

$$\frac{dN}{dt} = -\gamma N. \tag{10}$$

In other words, since the substances are unstable, this introduces randomness in the decay process. Thus, the decay rate (γ) with time (t) is reexpressed as

$$\gamma_t = \gamma_o e^{-\lambda t}, \tag{11}$$

where γ_o is a random initial value of substance that is decreased at each iteration and where t is the number of iterations or time steps. $t \in [0, \text{Max\_itr}]$, where Max_itr is the maximum number of iterations. The decay rate γ_t at time t to indicate a new trail or old trail is expressed as

$$\text{if } \gamma_t \longrightarrow \begin{cases} \gamma_t > 1, & \text{trail is new} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Again, the decay constant λ is expressed by

$$\lambda = \frac{\phi_{\max} - \phi_{\min}}{t_{1/2}}, \tag{13}$$

where λ is the decay constant, $\phi_{\max}$ is the maximum number of substances in the trail, $\phi_{\min}$ is the minimum number of substances in the trail, and $t_{1/2}$ is the half-life period of a trail which shows that the trail is old and unattractive.

Finally, the Kestrel updates its position using the following equation:

$$x_{i+1}^{k} = x_i^k + \beta_o e^{-\gamma r^2}\left(x_j - x_i^k\right) + f_i^k, \tag{14}$$

where x_{i+1}^{k} is the current best position of the Kestrel which represents the candidate solution and x_i^k is the previous position of the Kestrel. $\beta_o e^{-\gamma r^2}$ represents the attractiveness as expressed in (7). x_j represents a Kestrel with a better position while f_i^k is the frequency of bobbing as expressed in (6).

(iii) Fitness Function. The fitness function is used to evaluate how well the algorithm performs in terms of the quality of estimation. This performance is measured in terms of minimizing the deviation of data points from the estimated value without considering the direction (negative or positive) of the fitness value. Thus, the performance measurement method used the mean of absolute error (MAE) as fitness function evaluation because it allows the model to fine-tune absolute values and improve on the performance of values into much finer positive values without consideration of negative values. The MAE is expressed in

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \left|o_i - x_i\right|, \tag{15}$$

(i) Set parameters
(ii) Initialize population of n Kestrels using equation (3) and evaluate fitness of population using equation (18)
(iii) Start iteration (loop until termination criteria is met)
Compute Half-life of trail using equation (11)
Compute frequency of bobbing using equation (6)
Evaluate position for each Kestrel as in equation using equation (14)
If $f(x_i) < f(x_j)$ then
Move Kestrel i towards j
End if
Update position $f(x_i)$ for all $i = 1$ to n as in equation (20)
Find the current best value
(iv) End loop

Algorithm 1: The proposed algorithm for KSA.

where o_i is the observed data point at the ith position in the sampled dataset, x_i is the estimated value at the ith position in the dataset, and n is the number of data points in the sampled dataset. There are other evaluation methods such as root mean square error (RMSE) and mean square error (MSE).

The root mean square error (RMSE) measures the mean square error in the original data point and estimated value. The RMSE is expressed as the square root of the variance (i.e., standard deviation) in

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(o_i - x_i\right)^2}, \tag{16}$$

where o_i is the observed data point at the ith position in the sampled dataset, x_i is the estimated value at the ith position in the sampled dataset, and n is the number of data points in the sampled dataset.

The mean square error (MSE) measures the square of the deviation between the estimated values and the actual data point for the variable being considered; the smaller the MSE value, the better the accuracy of estimation, and vice versa. The MSE is expressed in

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(o_i - x_i\right)^2, \tag{17}$$

where o_i is the observed data point at the ith position in the sampled dataset, x_i is the estimated value at the ith position in the sampled dataset, and n is the number of data points in the sampled dataset.

The difference between the RMSE and the MSE is that MSE minimizes the error between the observed data and estimated value, but RMSE further minimizes the variance, while the mean of absolute error (MAE) measures the magnitude of errors without considering the direction of the fitness value.

In our comparison, we expressed the fitness function using the mean of absolute error as follows:

$$\text{fitness}\,(x) = \frac{1}{n}\sum_{i=1}^{n}\left|o_i - x_i\right|, \tag{18}$$

where o_i is the observed data point at the ith position in the sampled dataset, x_i is the estimated value at the ith position in the dataset, and n is the number of data points.

(iv) Velocity. The velocity of a Kestrel moving from its current best position in a changing airstream is

$$v_{t+1}^{k} = v_t^k + x_t^k, \tag{19}$$

where v_{t+1}^{k} is the current best velocity, v_t^k represents the initial velocity, and x_t^k represents the best position of the Kestrel. The change in velocity is controlled by the inertia weight ω (which is also referred to as the convergent parameter). This inertia weight has a linearly decreasing value. The final velocity is thus expressed to include this inertia weight as expressed in

$$v_{t+1}^{k} = \omega v_t^k + x_t^k, \tag{20}$$

where ω is the convergence parameter, v_t^k is the initial velocity, x_t^k is the best position of the Kestrel, and v_{t+1}^{k} is the current best velocity of the Kestrel. A Kestrel search through the search space in order to find an optimal solution requires the continuous update of the velocity, random encircling, and position towards the best estimate.

The proposed algorithm to implement KSA is expressed in Algorithm 1.

The Kestrel formulation also adopts the aspect of swarm behavior in terms of individual searching, moving to a better position, and fitness evaluation. However, what makes the Kestrel distinctive is the individual hunt through its random encircling of prey and its imitation of the best individual Kestrel. Since Kestrels hunt individually and imitate the best features of successful individual Kestrels, it is suggested that Kestrels are able to remember the best solution from a particular search space and continue to improve upon the initial solution until the final best is reached.

In comparison of the unique characteristics of the Kestrel algorithm with the Firefly, the wolf, and the BAT algorithms, the following can be stated: the Firefly algorithm is based on attractiveness, collective behavior, and brightness; the wolf algorithm is also based on attractiveness, collective behavior, and escape; the BAT algorithm is based on pulse

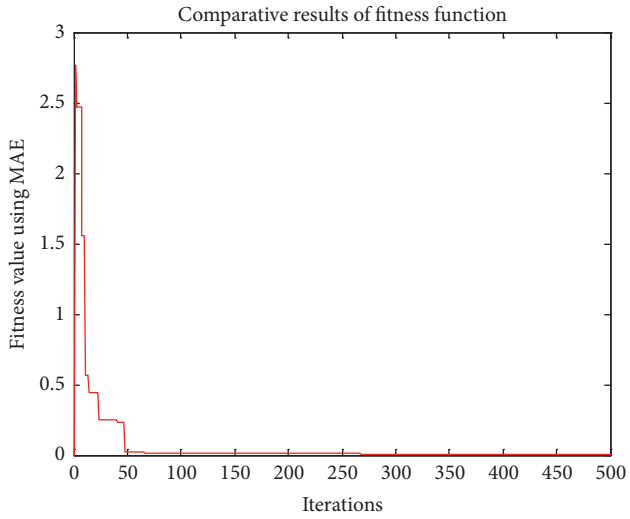

Figure 1: KSA fitness.

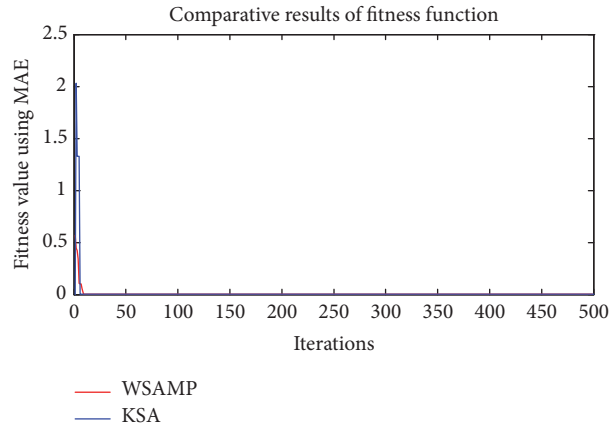

Figure 2: Comparison of KSA with WSAMP algorithm.

rate and loudness; and the Kestrel algorithm is based on attractiveness, brightness of the trail which is dependent on the half-life period, and encircling. This encircling behavior allows Kestrels to be adaptable in searching multiple missing values within a particular search space. The basis for the comparison is to assess the interesting behavior of the newly developed algorithm (i.e., KSA) and show how different this newly developed algorithm is from previous algorithms.

4. Experimental Results

The proposed algorithm was implemented in MATLAB 2012A and the quality of estimation was evaluated with the MAE method.

The initial parameters for KSA were set as $\beta_o = 1$ and visual range = 1. As expressed in (5), the following parameters were set for the lower and higher bound as z min = 0.2 and z max = 0.9, respectively. Representative data was used to test our algorithm and a maximum of 500 iterations/generations were done to have a greater chance to further refine the best value in each run. A sample set of data (46×9 matrix problem dimension/scale) with multiple missing values in the row matrix was used in order to provide a thorough test of missing values in each row of a matrix. Figure 1 shows the test results.

Figure 1 depicts the single graph on the fitness value of KSA after 500 iterations. The curve ascends and descends steeply during the start of iteration and then gradually converges at the optimal solution at the end of the iterations. The nature of the curve at the initial iteration suggests that KSA quickly maximizes the search space and gradually minimizes until its convergence to a global optimum value of **0.007421**.

Figure 2 depicts the comparison of fitness value of KSA as expressed in (18). During the comparison process, 500 iterations/generations were performed. The basis of this comparison is to demonstrate how the proposed algorithm performs in larger generations and thus allow the adequate refinement of the best fitness value.

Table 1: Comparative results on KSA and WSAMP.

Algorithm	Fitness using MAE
KSA	**7.9912e − 05**
WSAMP	5.6978e − 07

The Wolf Search Algorithm with a step Minus Previous (WSAMP) [22] was used as a comparative algorithm because it allows wolves to have a memory of the previous best position that has been visited; hence, the old positions are avoided when a new position is being generated during the search, where Minus Previous means wolves can only remember up to a previous visited step. In WSAMP, the randomness (σ) parameter was set to 0.2 while escape from the local minimum was also set to 0.25. Figure 2 shows the comparison of fitness evaluation of KSA and WSAMP algorithm both using MAE as a fitness function.

Figure 2 shows the curve on comparison of the fitness evaluation of KSA with WSAMP in 500 iterations/generations. The fitness curve gradually slopes down and maintains constant fitness, indicating quick convergence at the start of iteration. Table 1 indicates the fitness values of the curve on both KSA and WSAMP.

Table 1 shows the comparative results on KSA with WSAMP. The resultant fitness values showed WSAMP having a minimum fitness value of 5.6978e − 07 as compared with KSA which has a fitness value of 7.9912e − 05. In several iterations that were performed, WSAMP maintained the minimum fitness values as compared to KSA. The results showed that WSAMP outperformed KSA in terms of the minimum error because WSAMP was able to remember the best position previously visited.

In BAT algorithm, both the loudness and the pulse rate were set to 0.5 without fine-tuning these parameters. Also, the arbitrary frequency range was set to a minimum of 0.2 and a maximum of 0.9. This frequency range determines the frequency scaling of a bat. The BAT algorithm was compared with KSA and the comparative curve of the fitness value is illustrated in Figure 3.

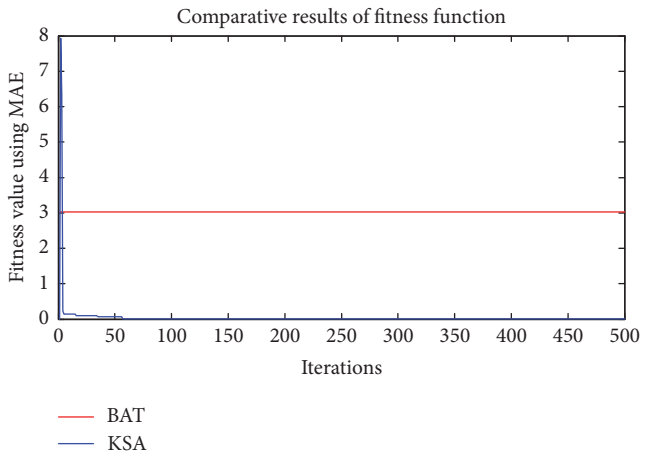

Figure 3: Comparison of KSA with BAT algorithm.

Table 2: Comparative results on KSA and BAT algorithm.

Algorithm	Fitness using MAE
KSA	**0.0029716**
BAT	3.0326

Figure 3 shows the curve on comparison of the fitness evaluation of KSA with BAT algorithm in 500 iterations/generations. The fitness curve for KSA peaks at the initial iteration and gradually slopes down and maintains a constant fitness value, thus indicating convergence. Table 2 indicates the fitness values of KSA and BAT algorithm.

Table 2 shows the comparative results on KSA with BAT algorithm. The resultant fitness values show KSA having a fitness value of **0.0029716** while the BAT algorithm has a fitness value of 3.0326. The BAT algorithm, however, showed a horizontal line from the initial iteration to the end of the iterations. This suggests that the BAT algorithm was not able to converge to a global minimum. Thus, the BAT algorithm is used for estimating missing values at random results in a high error of estimation. Thus, KSA outperformed the BAT algorithm in finding the optimal value.

In the Firefly algorithm, the randomness (σ) and absorption coefficient (γ) were set to 0.2 and 1.0, respectively. This setting allowed a small interval between the random numbers being generated. However, randomness reduction was set to 0.97 (similar to an annealing schedule).

Figure 4 shows the comparison of KSA with the Firefly algorithm. The curve indicates that KSA converges to a global minimum after the end of the iterations, while the curve on the Firefly algorithm shows several local minimum curves during the start of iteration, and then the curve smooths until the final iteration, suggesting that the curve moves from a local minimum and then gradually lessens to a global minimum. Table 3 indicates the fitness values of both KSA and Firefly algorithm.

Table 3 shows the comparative results on KSA with the Firefly algorithm. The fitness value of KSA converges to a value of **0.0054204** while the Firefly algorithm converges to a fitness value of 1.0000. This suggests that KSA produces the

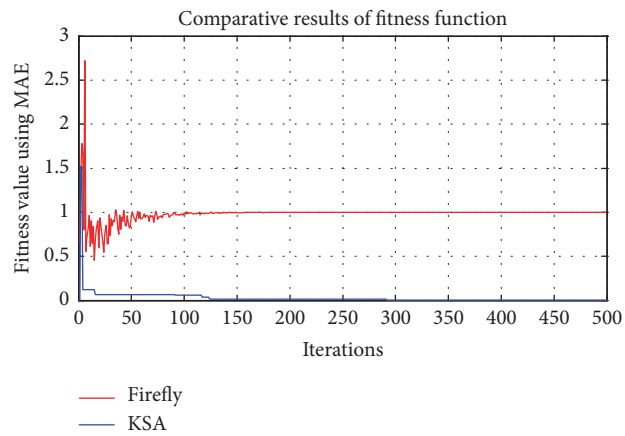

Figure 4: Comparison of KSA with Firefly algorithm.

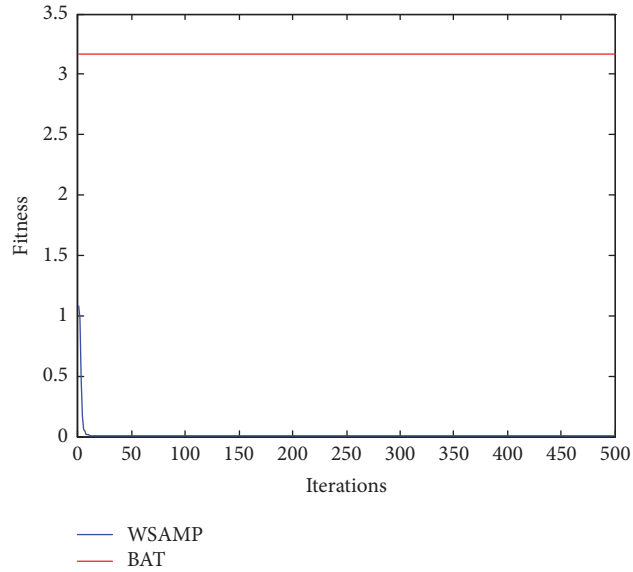

Figure 5: A figure showing the comparison between the BAT algorithm and the WSAMP.

Table 3: Comparative results on KSA and Firefly algorithm.

Algorithm	Fitness using MAE
KSA	**0.0054204**
Firefly	1.0000

minimum error when estimating missing values as compared with the Firefly algorithm.

The comparative curve of the fitness value is illustrated in Figure 5.

Figure 5 shows the curve between BAT algorithm and WSAMP; while the curve on BAT algorithm is constant, the curve on WSAMP gradually slopes until convergence. Table 4 shows the values of both BAT algorithm and WSAMP.

Table 3 shows the comparative results on BAT algorithm with the WSAMP algorithm. The fitness value of WSAMP converges to a value of **1.4864e − 7** while the BAT algorithm was constant at 3.1703. This suggests that WSAMP produces

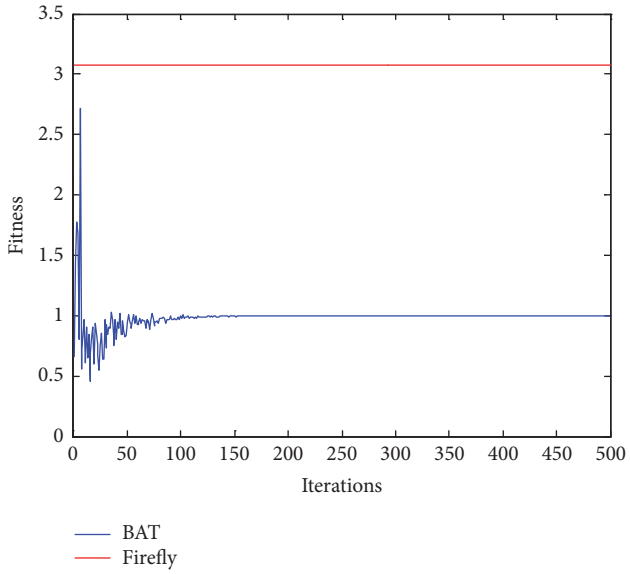

Figure 6: A figure showing the curve between the BAT algorithm and Firefly.

Table 4: Comparative results on BAT and WSAMP algorithm.

Algorithm	Fitness using MAE
BAT	3.1703
WSAMP	**1.4864e − 7**

Table 5: Comparative results on BAT and Firefly algorithm.

Algorithm	Fitness using MAE
BAT	3.1703
Firefly	**1.000**

the minimum error in estimating values that are missing as compared with BAT algorithm.

Figure 6 shows the curve between BAT algorithm and Firefly algorithm. The curve on the Firefly algorithm shows several local minimum curves from the start of iteration to approximately the 150th iteration; thereafter, the curve maintained a constant horizontal line until the final iteration, suggesting a global minimum; the curve on BAT algorithm has a constant horizontal line from the start of iteration to the end of iteration. Table 5 shows the result of comparison between BAT algorithm and Firefly algorithm.

The fitness value of the Firefly algorithm converges to 1.000 while the BAT algorithm was constant at 3.1703. This suggests that the Firefly algorithm produces the minimum error when estimating missing values. Thus, the Firefly algorithm outperforms BAT algorithm in terms of minimum error.

Figures 7 and 8 show the curve between WSAMP algorithm and Firefly algorithm, respectively, so as to show a clear figure on the nature of the curve.

Although both algorithms converged to a minimum, the fitness value of WSAMP converges to a value of 8.0889e − 07,

Table 6: Comparative results of WSAMP and Firefly algorithm.

Algorithm	Fitness using MAE
WSAMP	8.0889e − 07
Firefly	**1.000**

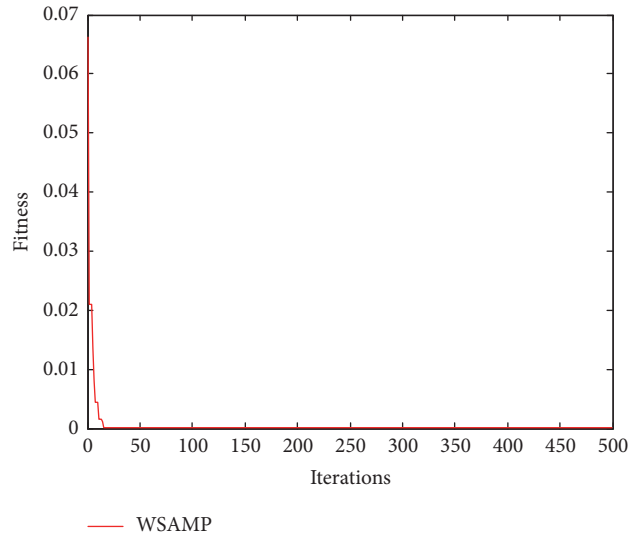

Figure 7: WSAMP algorithm.

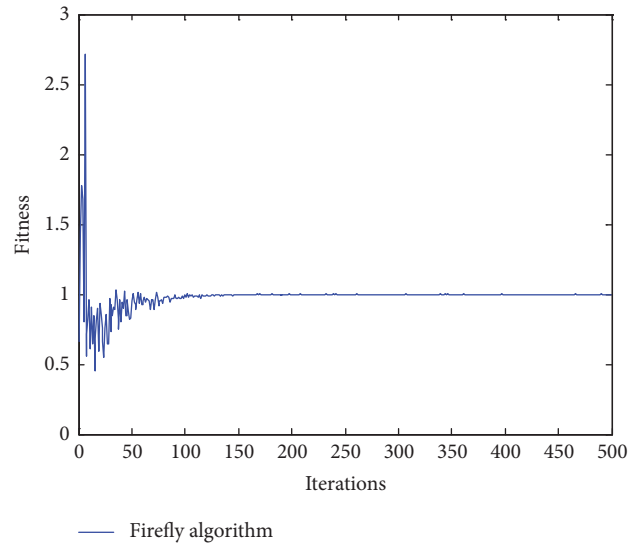

Figure 8: Firefly algorithm.

while the curve on the Firefly algorithm shows several local minimum curves and gradually lessens to a global minimum.

Table 6 shows the comparative results on WSAMP with the Firefly algorithm. While the fitness value of the Firefly algorithm converges to 1.000, the WSAMP algorithm converges at 8.0889e − 07. This suggests that WSAMP algorithm produces the minimum error. Therefore, the WSAMP algorithm outperforms the Firefly algorithm when estimating missing values. Table 7 shows a summary of the results on

Table 7: Summary of MAE results obtained on a 46 × 9 sample matrix problem dimension.

Algorithm	Problem dimension	MAE1	MAE2	MAE2
KSA	46 × 9	**7.9912e − 05**	**0.0029716**	**0.0054204**
WSAMP	46 × 9	*5.6978e − 07*	*1.4864e − 7*	*8.0889e − 07*
BAT	46 × 9	3.0326	3.1703	3.1703
Firefly	46 × 9	1.0000	1.0000	1.0000

Table 8: MAE results from comparative algorithms on different problem scales.

Problem dimension	KSA MAE	BAT MAE	Firefly MAE	WSAMP MAE
40 × 9	7.09E − 05	3.0326	0.90723	8.16E − 07
30 × 9	0.012553	3.0843	1	2.51E − 07
20 × 9	0.04752	3.0655	0.15362	9.22E − 06
25 × 9	0.023865	3.3836	1	1.34E − 07
10 × 9	0.39469	3.536	0.6943	1.73E − 05

mean absolute errors obtained from each paired algorithm in the 46 × 9 sample matrix problem dimension.

Table 7 shows a summary of the results where each column represents the MAE obtained from each iteration in the 46 × 9 matrix. The results indicate that BAT algorithm performed poorly as compared to WSAMP and Firefly algorithm.

Different problem scale/dimension of the dataset was applied to each algorithm and the corresponding fitness value (i.e., MAE) was computed. The dimensions that were selected help to observe the behavior of each algorithm on different problem scales. Table 8 shows the results on MAE values obtained from the comparative algorithms on different problem dimensions.

Table 8 indicates that KSA obtained optimal results as compared with BAT algorithm and Firefly algorithm in different problem scales. The results suggest that KSA is the best for random encircling and this algorithm is one of the best/optimal in estimating multiple missing values in any big data analysis environment.

5. Statistical Analysis of Experimental Results

The basis for the statistical analysis of experimental results on the comparative algorithms is to find the significance of results obtained from each algorithm. In order to achieve this comparison in an accurate manner, the study used a profile on all the test functions used in each of the algorithms and the MAE results (i.e., the quality of estimation) in Table 8. The nonparametric statistical procedure was used to analyze the significance of the comparison. This nonparametric statistical procedure was used as it does not make underlying assumption on parameters such as mean and variance of the algorithm being assessed. In contrast, parametric statistical procedures make assumptions on parameters that are being assessed. In this section, the profiling of test functions and the nonparametric statistical procedure adopted for the analysis were discussed. This section is structured into two: statistical analysis on profiling of test functions and statistical analysis on MAE results (quality of estimation).

(i) Statistical Analysis on Profiling of Test Functions. Profiling is a technique used to measure time spent on aspects of a program such as a function [33]. This technique helps to optimize functions and improve on the performance of the algorithms. During profiling, the following are considered: function name, the number of times a function was called upon (i.e., calls), the total time spent on each function including subfunctions (total_time), and the total time spent on a function excluding the time spent on subfunctions (i.e., self_time). It is possible for functions that are less time intensive to call other functions that are time intensive. The profiling technique is important as it determines which functions are responsible for calling other functions.

The study applied profiling as a technique to extract functions and to group the functions into two categories, namely, major functions and basic functions. While the major functions are functions that were written to implement the behavior of the algorithms, the basic functions are the in-built functions that work alongside the major function. Table 9 indicates how the functions are grouped during profiling on each comparative algorithm.

Table 9 shows the different test functions in each comparative algorithm. The WSAMP has 2 major functions categorized into the main function (represented by f1) and a subfunction (represented by f2). All main functions are represented in each algorithm by f1. The Firefly algorithm has 6 major functions (one main function f1 and 5 subfunctions). The BAT algorithm consists of 3 major functions (one main function f1 and 2 subfunctions), while KSA consists of 4 major functions (one main function f1 and 3 subfunctions).

In order to have a true reflection on the nature of in-built functions that were extracted, all in-built functions were considered for analysis. It is observed that when some in-built functions were called, the total_time is zero seconds, thus making those in-built function calls inconsequential in terms of execution time. However, these inconsequential in-built functions were taken into consideration so as not to lose track of any function calls made.

A statistical procedure was applied to analyze the significance of profile results obtained in Table 9. The study conducted a nonparametric statistical test to assess which of the algorithms has better performance in terms of the behavior of test function call time. The basis for the statistical analysis is find out the significance of the profiled results from each algorithm. Reference [34] indicated that nonparametric or distribution-free statistical procedures help to perform pairwise comparison on related algorithms even in the case

Table 9: Major function names of the comparative algorithms.

	Function name	Calls	Total_time (s)	Self_ time (seconds (s))
	WSAMP algorithm			
f2	WSAMPnew>fnc_fitness_mae	172517	1.298	1.298
f1	WSAMPnew>main	1	14.426	12.993
	Mean	*86259*		
	Firefly algorithm			
f2	fireflyApproachnew>fnc_fitness_MAE	4522	0.018	0.018
f3	fireflyApproachnew>ffa_move	500	0.415	0.404
f4	fireflyApproachnew>findrange	500	0.011	0.011
f5	fireflyApproachnew>newalpha	500	0.002	0.002
f1	fireflyApproachnew>main	1	47.811	29.291
f6	fireflyApproachnew>init_fireflyalg	1	0.001	0.001
	Mean	*1004*		
	KSA algorithm			
f2	KSAapproachnew>fnc_fitness_MAE	23046	0.094	0.094
f3	KSAapproachnew>fnchalflife	500	0.002	0.002
f4	KSAapproachnew>fncbobbing	500	0.011	0.011
f1	KSAapproachnew>main	1	1.329	1.036
	Mean	*6011.8*		
	BAT algorithm			
f2	BATApproachnew>fnc_fitness_MAE	23046	0.101	0.101
f3	BATApproachnew>simplebounds	23000	0.317	0.317
f1	BATApproachnew>main	1	1.353	0.718
	Mean	*15349*		

where the sample size of a dataset is small such as where sample size $n < 30$. When there are multiple comparisons of algorithms, the Wilcoxon signed-rank test helps to rank algorithms and test how significantly the algorithms outperform each other [34]. In this article, the mean of the time to call test functions of each algorithm was used in order to suggest that two algorithms are equivalent. In order to indicate the probability of error in the median of the two algorithms, the p value was used [35]. The advantage of Wilcoxon test is that there is no need to make an assumption about the population of functions being used since the Wilcoxon test can guarantee about 95% (i.e., 0.05 level of significance) of efficiency if the population is normally distributed, meaning that if there are 500 observations on test function calls, then the Wilcoxon signed-rank test is efficient to about 499 observations on test function calls. Reference [36] indicated that the Wilcoxon signed-rank test is analogous to the related sample t-test; however, the t-test is unsuitable for this type of analysis, while Wilcoxon is suitable. Samples are related if one sample matches the other sample, while the rank is a number assigned to an individual sample according to its order in a list of algorithms. Thus, the Wilcoxon statistical technique helps to assign ranks to algorithms in order to identify the best ranked evolutionary algorithms behavior [37] and to determine the significance of each algorithm. The following steps are applied in computing the Wilcoxon signed-rank test.

Step 1. Compute the difference D of paired samples in each algorithm. Any pairs with a difference of 0 are discarded.

Step 2. Find the absolute D.

Step 3. Compute the rank of signs (R_+ difference and R_- difference) from the lowest to the highest.

The sum of ranks is expressed by

$$\sum R_+ + R_- = \frac{n(n+1)}{2}, \tag{21}$$

where n is the sample size.

Step 4. Compute the test statistic T. Thus, $T = \min\{R_+, |R_-|\}$. Thus, the test statistic T is the smallest value.

Step 5. Find the critical values based on the sample size n. If T is less than or equal to the critical value at a level of significance (i.e., $\alpha = 0.05$), then a decision is made that algorithms are significantly different [34]. In order to accomplish this, the Wilcoxon signed-rank table is consulted, using the critical value ($\alpha = 0.05$) and sample size n as parameters, to obtain the value within the table. If this value is less than the calculated value of the algorithmic comparison, this means that the algorithmic difference is significant.

In order to apply the Wilcoxon signed-rank test, an analysis was performed on the time to call a function name (both in-built functions and major functions) as follows.

(a) Time Analysis of the In-Built Function Calls. The performance of the comparative algorithms was based on test

Table 10: Wilcoxon rank on profile extracts on in-built function calls of algorithms.

	Algorithms	Sum of calls	Total number of in-built functions	Sum of self_time	Sum of total_time	D	R_+/R_-	Rank	Sum of signed ranks
(1)	WSAMP	123	31	0.132	0.401	0.269	1	1	1
(2)	Firefly	99780	229	49.228	160.73	111.503	1	4	4
(3)	KSA	1043	34	0.184	0.471	0.287	1	2	2
(4)	BAT	105	69	0.215	0.953	0.738	1	3	3

Table 11: Mean and standard deviation on in-built functions.

Sum	N	Mean	Std. deviation	Minimum	Maximum
Self_time	4	12.4398	24.52552	.13	49.23
Total_time	4	40.6387	80.06121	.40	160.73

function call time differences between the self_time and total_time of in-built functions in each algorithm. Based on the steps in computing Wilcoxon signed-rank test, the time for the test function call per each algorithm is shown in Table 10.

From Table 10, D represents the difference between the sum of total_time and the sum of self_time. It is observed that the sum of the signed positive ranks R_+ is 10 while the sum of negative ranks is 0. Since the sample size n (4) is less than 30 and the Wilcoxon signed-rank table shows that there is no critical region on the subfunction at $\alpha = 0.05$, the Wilcoxon signed-rank table suggests that in-built functions are equivalent. In terms of ranking of algorithms, the WSAMP was ranked first while KSA was ranked second.

Since n is small, it is tedious to find a critical value for small values of n. Table 11 shows the mean and standard deviation on the sum of self_time and sum of total_time on the subfunction.

Table 11 shows sample size (N) and the mean on sum of self_time as 12.4398 and sum of total_time as 40.6387 with their corresponding standard deviation. Since the results show a standard deviation of 80.06121 on total_time, it suggests a high deviation of total_time on the in-built function calls as compared with low standard deviation of 24.52552 on self_time in-built function calls. Thus, there is a high total_time spent on in-built function calls in the algorithms as compared with self_time on in-built function calls, meaning that the algorithms spent an intensive amount of time in calling an average of 40.6387 in-built functions and an average time of 12.4398 in excluding other in-built function calls. Table 12 illustrates the Wilcoxon signed-rank test between total_time and self_time of in-built function calls computed using the Statistical Package for the Social Sciences (SPSS). Table 12 contains the sample size (N), mean rank, and sum of ranks.

Table 12 shows Wilcoxon signed ranks on the comparison of sum of total_time and sum of self_time. There were 4 samples between total_time and self_time. The basis is to find whether the differences between total_time and self_time are significantly different from zero and whether the differences that were observed in the mean rank (0.00 versus 2.50) can be located in the population of in-built function calls. In order

Table 12: Wilcoxon signed ranks on in-built functions.

		N	Mean rank	Sum of ranks
Sum of total_time − sum of self_time	Negative ranks	0[a]	.00	.00
	Positive ranks	4[b]	2.50	10.00
	Ties	0[c]		
	Total	4		

[a]Sum of total_time < sum of self_time. [b]Sum of total_time > sum of self_time. [c]Sum of total_time = sum of self_time.

Table 13: Test statistics[b] on in-built functions.

	Sum of total_time − sum of self_time
Z	−1.826[a]
Asymptotic sig. (2-tailed)	.068

[a]Based on negative ranks. [b]Wilcoxon signed-rank test.

to locate the value between the mean ranks (0.00 versus 2.50), the test of significance of time on performance on in-built functions is computed as in Table 13.

Table 13 shows the test statistic that was obtained. The asymptotic sig. (2-tailed) in the table represents the p value for the test, while the Wilcoxon signed-rank test was computed using the z statistic. Thus, the Wilcoxon signed-rank test was used on 4 samples to find out whether there is a significant change of total_time with self_time showing $z = -1.826$ and $p = 0.068$. Since $p > \alpha$ (0.05) within the mean rank (0.00 versus 2.50), the value of 0.068 indicates that, statistically, time did not result in a significant change in performance of in-built function calls.

(b) Time Analysis on the Major Function Calls. In this study, we also conducted a statistical analysis to determine whether time might be significant in enhancing the performance of major function calls of each algorithm. Table 14 shows the summary of the Wilcoxon signed-rank test on major functions between self_time and total_time of each algorithm.

Table 14 shows the difference D between the sum of total_time and the sum of self_time of each algorithm. It is observed that the sum of signed positive ranks R_+ is 4 while the sum of negative ranks is 0. Since the sample size n (4) is less than 30 and from the Wilcoxon signed-rank table, it shows that there is no critical region, on the significance level of $\alpha = 0.05$, on the Wilcoxon signed-rank table, to suggest that the major functions are significantly different in terms of total_time and self_time of calling the major functions. All

Table 14: Wilcoxon rank on profile extracts on major functions total_time and self_time of algorithms.

	Sum of calls	Total number of major functions	Sum of self_time	Sum of total_time	D	R_+/R_-	Rank	Signed rank
(1) WSAMP	172518	2	14.291	15.724	1.433	1	1	1
(2) Firefly	6024	6	29.727	48.258	18.531	1	1	1
(3) KSA	24047	4	1.143	1.436	0.293	1	1	1
(4) BAT	46047	3	1.136	1.771	0.635	1	1	1

Table 15: Mean and standard deviation on major functions.

	N	Mean	Std. deviation	Minimum	Maximum
Sum of self_time	4	11.5742	13.59744	1.14	29.73
Sum of total_time	4	16.7973	22.00520	1.44	48.26

Table 16: Wilcoxon signed ranks on major functions.

		N	Mean rank	Sum of ranks
Sum of total_time − sum of self_time	Negative ranks	0[a]	.00	.00
	Positive ranks	4[b]	2.50	10.00
	Ties	0[c]		
	Total	4		

[a]Sum of total_time < sum of self_time. [b]Sum of total_time > sum of self_time. [c]Sum of total_time = sum of self_time.

Table 17: Test statistics[b] on major functions.

	Sum of total_time − sum of self_time
Z	−1.826[a]
Asymptotic sig. (2-tailed)	.068

[a]Based on negative ranks. [b]Wilcoxon signed-rank test.

the major functions of each algorithm were ranked equally. The Wilcoxon signed-rank test was conducted to test whether there was a significant difference. Firstly, Table 15 indicates the mean and standard deviation on both the sum of self_time and the sum of total_time on the major functions.

Table 15 shows sample size (N) and the mean on sum of self_time as 11.5742 and sum of total_time as 16.7973 with their corresponding standard deviation. The results indicate a standard deviation for total_time as 22.00520 and self_time as 13.59744. Thus, there is a high variation in total_time as compared with low variation in self_time of major function calls. Table 16 illustrates the Wilcoxon signed-rank test between total_time and self_time of major function calls.

Table 16 shows Wilcoxon signed ranks on the comparison of sum of total_time and sum of self_time. There were 4 samples between total_time and self_time. The basis of this comparison is to find out whether the differences between total_time and self_time are significantly different and whether the differences that were observed in the mean rank (0.00 versus 2.50) can be located in the population of major function calls. In order to locate the value between the mean ranks (0.00 versus 2.50), the test of significance of time on performance is computed in Table 17.

Table 18: Results on accuracy from comparative algorithms using MAE.

Problem dimension	KSA MAE	BAT MAE	Firefly MAE	WSAMP MAE
46×9	$7.99E-05$	3.0326	1.000	$5.70E-07$
40×9	$7.09E-05$	3.0326	0.90723	$8.16E-07$
30×9	0.012553	3.0843	1.000	$2.51E-07$
20×9	0.04752	3.0655	0.15362	$9.22E-06$
25×9	0.023865	3.3836	1.000	$1.34E-07$
10×9	0.39469	3.536	0.6943	$1.73E-05$

Table 17 shows the test statistic that was obtained. The asymptotic sig. (2-tailed) in the table represents the p value for the test, while the Wilcoxon signed-rank test was computed using z statistic. The Wilcoxon signed-rank test is used to find whether there is a significant change in total_time and self_time at $z = -1.826$ and $p = 0.068$. The results indicate that, statistically, time did not result in a significant change in performance of major function calls.

(ii) Statistical Analysis on Output Results on Quality of Estimation. Wilcoxon signed-rank test was conducted on all the dimensions of results on quality of estimation in Table 18.

Table 18 consists of all problem dimensions of each algorithm and the respective MAE. Although the null hypotheses were formulated based on 46×9 dimension, the study tested the hypotheses on all problem dimensions of the MAE value. Earlier, the analysis on performance of each paired algorithm was made as follows:

(1) WSAMP outperformed KSA in terms of the minimum error (MAE).

(2) KSA outperformed BAT algorithm in finding the optimal value.

(3) KSA produces the minimum error when estimating missing values as compared with the Firefly algorithm.

(4) WSAMP produces the minimum error when estimating missing values as compared with the BAT algorithm.

(5) The Firefly algorithm outperforms BAT algorithm in terms of the minimum error.

(6) The WSAMP algorithm outperforms the Firefly algorithm based on their MAE.

In order to test the significance of quality of estimation, the Wilcoxon test statistic was computed using SPSS and the p value is shown in Table 19.

Table 19: Wilcoxon signed-rank test statistic on accuracy.

Comparative algorithm	Asymp. sig. (2-tailed) p value
KSA versus WSAMP	0.028
KSA versus Firefly	0.028
KSA versus BAT	0.028
Firefly versus BAT	0.028
WSAMP versus Firefly	0.028
WSAMP versus BAT	0.028

Table 19 shows the p value that was obtained from each comparing algorithm. Since the p values must be less than or equal to the level of significance of 0.05 in order to be significant, the results in Table 19 show that the quality of estimation was significant between the paired algorithms. In this case, the WSAMP significantly outperformed KSA in terms of the MAE.

Multiple Comparison of Output Results on Accuracy. Reference [38] indicated that the Wilcoxon signed-rank test is best used for pairwise comparisons between two algorithms. In a multiple comparison situation where two or more algorithms are compared, it is possible for errors to accumulate such that the performance of algorithms is significant. Reference [34] indicated that performing multiple comparison enables correcting the Family-Wise Error Rate (FWER) which occurs after multiple algorithms are combined. In order to perform comparison, [34] used the results of accuracy obtained by algorithms to perform statistical analysis on algorithms. The statistical significance of combining a pair of algorithms is computed by

$$\begin{aligned} p &= P\left(\text{Reject } H_o \mid H_o \text{ true}\right), \\ p &= 1 - P\left(\text{Accept } H_o \mid H_o \text{ true}\right), \\ p &= 1 \\ &\quad - P\left(\text{Accept } A_k = A_i,\ i = 1, \ldots, k-1 \mid H_o \text{ true}\right), \\ p &= 1 - \prod_{i=1}^{k-1} P\left(\text{Accept } A_k = A_i \mid H_o \text{ true}\right), \\ p &= 1 - \prod_{i=1}^{k-1} \left[1 - P\left(\text{Reject } A_k = A_i \mid H_o \text{ true}\right)\right], \\ p &= 1 - \prod_{i=1}^{k-1} \left(1 - pH_i\right). \end{aligned} \tag{22}$$

Using expression (22), p values of each algorithm are computed to find the final p value. If the p value is less than the critical value (e.g., $\alpha = 0.05$), then it forms the basis for rejection of a hypothesis. However, a final decision cannot be made to fully reject or fail to reject (accept) a hypothesis based on an analysis result without performing a test on the possible error that could be accumulated when comparing algorithms.

The Friedman test can be conducted to compare two or more evolutionary algorithms and find errors that have been accumulated when two or more algorithms are compared [39, 40]. The Friedman test is a two-way analysis of the variations in the ranking of algorithms. The Friedman test is a nonparametric procedure that aims to compare the median of a distribution in order to find whether significant differences between the behaviors of two or more algorithms have occurred. The null hypothesis of Friedman test applies the equality of medians [41], while the alternative hypothesis negates the null hypothesis. The Friedman test procedure can be summarized into the following steps.

Step 1. Rank the algorithms for dataset separately.

Step 2. The best performing algorithm with the least MAE gets the rank of 1, the second best rank 2, and so forth.

Step 3. If there is a tie between ranks, assign the average rank. Let r_i^j represent the rank of the jth of k algorithm on the ith of N dataset.

Step 4. Friedman test compares the average ranks of the algorithm as follows:

$$R_j = \frac{1}{N} \sum_i r_i^j. \tag{23}$$

The null hypothesis computes the equivalence and the ranks R_j, which is equal to the Friedman statistic [39] computed as

$$X_F^2 = \frac{12}{nk\,(k+1)} \left[\sum_j R_j^2 \right] - 3n\,(k+1), \tag{24}$$

where R_j is the rank and X_F^2 is distributed with $k-1$ degrees of freedom, such that n and k should have a large sample size (n) (as a rule of a thumb, $n > 10$ and $k > 5$) [41] since large sample sizes are significant in computing the degree of freedom on the rank of algorithms. k is the number of groups that are being compared.

Step 5. The calculated value of X_F^2 must be larger than or equal to the appropriate critical table value of X^2 or larger than or equal to the value of X_F^2 in the small samples table.

Reference [41] indicated that, to perform multiple comparison, two measures are used; firstly, check whether the results obtained from the algorithm have inequality and rank using the Friedman test. The Friedman test states that, under a null hypothesis, all the algorithms are equivalent, so a rejection of a hypothesis indicates existence of significant differences in performance of all the algorithms studied [41].

In our approach to identify the best algorithm (deemed to be the algorithm with the lowest ranking value) that can be used as a control algorithm, the results in Table 18 were applied and the Friedman test was conducted to identify the best algorithm. In order to rank the algorithms, the mean and standard deviation were computed (in Table 20) on the results on accuracy of performance shown in Table 18.

The results in Table 20 indicate that KSA has the least standard deviation among the comparative algorithms. Since

Table 20: Descriptive statistics.

	N	Mean	Std. deviation	Minimum	Maximum
KSA	6	.0000	.00001	.00	.00
WSAMP	6	.7925	.33471	.15	1.00
BAT	6	.0798	.15528	.00	.39
Firefly	6	3.1891	.21606	3.03	3.54

Table 21: Ranks of algorithms.

	Mean rank
BAT	2.00
Firefly	4.00
WSAMP	3.00
KSA	1.00

Table 22: Friedmantest statistics[a].

N	6
Chi-square X^2	18.000
df	3
Asymp. sig. (p value)	0.000

[a]Friedman test.

the standard deviation measures the amount of variation in a set of data [42], thus, the larger the standard deviation, the greater the variation in the data, whereas the smaller the standard deviation, the smaller the amount of variation in the data. Since KSA has a minimum standard deviation of 0.00001, thus there is a small variation in KSA. Based on the results in Table 20, the Friedman test ranked the algorithms in Table 21.

The rank in Table 21 indicates that KSA is the best algorithm among the comparative algorithms. Friedman's test statistic with a sample size N was computed in Table 22.

Table 22 shows the results on Friedman test, where X^2 obtained is 18.000, with 3 degrees of freedom and a significance (asymp. sig.) level of 0.0000. Since the significance level is α (0.05), thus, the computed value on X^2 must be larger than or equal to the critical value for significance of 0.05. Since df is 3 at 0.05 level of significance, the value that was read from the critical value of chi-square X^2 distribution table [43] is 7.82, thus $18 > 7.82$ at α (0.05). There is a significant difference in the results on the quality of estimation of missing values among the algorithms, meaning that the algorithms are not the same.

6. Conclusion and Future Work

This paper presented a new bioinspired algorithm, the Kestrel-Based Search Algorithm (KSA), and compared it with other metaheuristic algorithms such as the Firefly, WSAMP, and BAT algorithms. The results of the comparison showed that the KSA demonstrated potential uniqueness in its search for the best value in comparison with other metaheuristic methods such as wolf, BAT, and Firefly algorithms. The statistical test conducted on the test function calls time based on Wilcoxon signed-rank test indicated that total_time did not result in a significant change in the performance of major function calls. Similarly, total_time did not result in a significant change in the performance of in-built function calls. Since the Wilcoxon test helps to assign ranks to algorithms in order to identify the best ranked evolutionary algorithms, the major functions and in-built functions were ranked. The results indicate that the KSA was ranked second while WSAMP was ranked first in terms of in-built function call. However, all algorithms were ranked equally in terms of major function call. Further tests conducted on the results of accuracy in multiple comparison applied Friedman test to detect the Family-Wise Error Rate. The results on Friedman test ranked KSA algorithm as the best algorithm that is used as a control algorithm for multiple comparison of algorithms. In the future, we intend to apply this algorithm in solving real-world problems and for other big data purposes such as association rule mining and intend to apply it to further enhance its performance of estimation of missing values.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] C. Longbottom and R. Bamforth, "Optimising the data warehouse," Dealing with large volumes of mixed data to give better business insights, Quocirca, 2013.

[2] N. S. D. Du Bois, "A Solution to the Problem of Linking Multivariate Documents," *Journal of the American Statistical Association*, vol. 64, no. 325, pp. 163–174, 1969.

[3] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[4] A. C. Acock, "Working with missing values," *Journal of Marriage and Family*, vol. 67, no. 4, pp. 1012–1028, 2005.

[5] F. V. Nelwamondo, S. Mohamed, and T. Marwala, "Missing data: a comparison of neural network and expectation maximisation techniques," https://arxiv.org/abs/0704.3474.

[6] R. B. Kline, *Principles and Practices of Structural Equation Modeling*, Guilford, New York, NY, USA, 1998.

[7] R. L. Carter, "Solutions for Missing Data in Structural Equation Modeling," *Research Practice in Assessment*, vol. 1, no. 1, 2006.

[8] J. R. Quinlan, "Unknown attribute values in induction," in *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 164–168, Morgan Kaufmann, Ithaca, NY, USA, 1989.

[9] P. D. Allison, *Handling Missing Data by Maximum Likelihood*, Statistical Horizons, Haverford, PA, USA, 2012.

[10] Y. Zhao, D. J. MacKinnon, and S. P. Gallup, "Big Data and Deep Learning for Understanding DoD Data," *Data mining and measurements*, 2005.

[11] X.-S. Yang, S. Deb, and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," in *Networked Digital Technologies: Third International Conference, NDT 2011, Macau, China, July 11–13, 2011. Proceedings*, vol. 136 of *Communications in Computer*

and Information Science, pp. 53–66, Springer, Berlin, Germany, 2011.

[12] C. Hurlin, *Maximum Likelihood Estimation Advanced Econometrics - HEC Lausanne*, Chapter 2, University of OrlØans, 2013.

[13] P.-J. Lu and T.-C. Hsu, "Application of autoassociative neural network on gas-path sensor data validation," *Journal of Propulsion and Power*, vol. 18, no. 4, pp. 879–888, 2002.

[14] I. E. Agbehadji, *Solution to the travel salesman problem, using omicron genetic algorithm. Case study: tour of national health insurance schemes in the Brong Ahafo region of Ghana [Master thesis]*, 2011.

[15] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA, 1975.

[16] M. Abdella and T. Marwala, "The use of genetic algorithms and neural networks to approximate missing data in database," *Computing and Informatics*, vol. 24, no. 6, pp. 577–589, 2005.

[17] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, pp. 1942–1948, Piscataway, NJ, USA, 1995.

[18] J. Krause, J. Cordeiro, R. S. Parpinelli, and H. S. A. Lopes, "A Survey of Swarm Algorithms Applied to Discrete Optimization Problems," *Swarm Intelligence and Bio-Inspired Computation*, pp. 169–191, 2013.

[19] X-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2008.

[20] X. S. Yang, "Firefly algorithm, Levy flights and global optimization," in *XXVI Research and Development in Intelligent Systems*, pp. 209–218, Springer, London, UK, 2009.

[21] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, Coimbatore, India, December 2009.

[22] R. Tang, S. Fong, X.-S. Yang, and S. Deb, "Wolf search algorithm with ephemeral memory," in *Proceedings of the 7th International Conference on Digital Information Management, ICDIM 2012*, pp. 165–172, Macao, August 2012.

[23] I. E. Agbehadji, S. Fong, and R. Millham, "Wolf search algorithm for numeric association rule mining," in *Proceedings of the 2016 IEEE International Conference on Cloud Computing and Big Data Analysis, ICCCBDA 2016*, pp. 146–151, Chengdu, China, July 2016.

[24] X.-S. Yang, *Firefly Algorithm, Levy Flights and Global Optimization*, 2010.

[25] I. Fister Jr., S. Fong, and J. Brest, "Towards the self-adaptation of the bat algorithm," in *Proceedings of the 13th IASTED International Conference on Artificial Intelligence and Applications, AIA 2014*, pp. 400–406, Innsbruck, Austria, February 2014.

[26] M. Shrubb, "The hunting behaviour of some farmland kestrels," *Bird Study*, vol. 29, no. 2, pp. 121–128, 1982.

[27] D. E. Varland, "Behavior and ecology of post-fledging American Kestrels," Retrospective Theses and Dissertations Paper 9784, 1991.

[28] C. Vlachos, D. Bakaloudis, E. Chatzinikos, T. Papadopoulos, and D. Tsalagas, "Aerial hunting behaviour of the Lesser Kestrel Falco naumanni during the breeding season in Thessaly (Greece)," *Acta Ornithologica*, vol. 38, no. 2, pp. 129–134, 2003.

[29] J. Honkavaara, M. Koivula, E. Korpimaki, H. Siitari, and J. Viitala, "Ultraviolet vision and foraging in terrestrial vertebrates," *Oikos*, vol. 98, no. 3, pp. 505–511, 2002.

[30] R. Kumar, Grey wolf optimizer (GWO), 2015, https://drrajeshkumar.files.wordpress.com/2015/05/wolf-algorithm.pdf.

[31] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[32] R. L. Spencer, Introduction to Matlab, 2002, https://www.physics.byu.edu/courses/computational/phys330/matlab.pdf.

[33] D. C. Sorensen, R. B. Lehoucq, C. Yang, and K. Maschhoff, *Profiling for improving performance*, Rice University, 2012.

[34] S. García, A. Fernández, A. D. Benítez, and F. Herrera, *Statistical Comparisons by Means of Non-Parametric Tests: A Case Study on Genetic Based Machine Learning*, 2007.

[35] J. H. Zar, *Biostatistical Analysis*, Prentice Hall, 1999.

[36] G. W. Heiman, *Basic Statistics for the Behavioral Sciences*, Houghton Mifflin Company, 2nd edition, 1996.

[37] S. García, D. Molina, M. Lozano, and F. Herrera, *A Study on The Use of Non-Parametric Tests for Analyzing The Evolutionary Algorithms' Behaviour: A Case Study on The CEC'2005 Special Session on Real Parameter Optimization*, Springer Science Business Media, LLC, 2008.

[38] B. Trawinski, B. Smętek, Z. Telec, and T. Lasota, "Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms," *International Journal of Applied Mathematics and Computer Science*, vol. 22, no. 4, pp. 867–881, 2012.

[39] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[40] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[41] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, Intelligent Systems Reference Library 72, Springer International Publishing, 1st edition, 2015.

[42] S. P. Gordon and F. S. Gordon, *Contemporary Statistics: A Computer Approach*, McGraw-Hill International Edition, 1994.

[43] P. R. Hinton, *Statistics Explained: A Guide for Social Science Students*, T. J. Press (Padstow) Ltd, Padstow, UK, 1995.