Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 590614, 18 pages
http://dx.doi.org/10.1155/2013/590614

Research Article

Selecting Optimal Feature Set in High-Dimensional Data by Swarm Search

Simon Fong,[1] Yan Zhuang,[1] Rui Tang,[1] Xin-She Yang,[2] and Suash Deb[3]

[1] *Department of Computer and Information Science, University of Macau, Macau*
[2] *Faculty of Science and Technology, Middlesex University, UK*
[3] *Department of Computer Science and Engineering, Cambridge Institute of Technology, Ranchi, India*

Correspondence should be addressed to Simon Fong; ccfong@umac.mo

Received 22 July 2013; Revised 1 October 2013; Accepted 8 October 2013

Academic Editor: Zong Woo Geem

Selecting the right set of features from data of high dimensionality for inducing an accurate classification model is a tough computational challenge. It is almost a NP-hard problem as the combinations of features escalate exponentially as the number of features increases. Unfortunately in data mining, as well as other engineering applications and bioinformatics, some data are described by a long array of features. Many feature subset selection algorithms have been proposed in the past, but not all of them are effective. Since it takes seemingly forever to use brute force in exhaustively trying every possible combination of features, stochastic optimization may be a solution. In this paper, we propose a new feature selection scheme called Swarm Search to find an optimal feature set by using metaheuristics. The advantage of Swarm Search is its flexibility in integrating any classifier into its fitness function and plugging in any metaheuristic algorithm to facilitate heuristic search. Simulation experiments are carried out by testing the Swarm Search over some high-dimensional datasets, with different classification algorithms and various metaheuristic algorithms. The comparative experiment results show that Swarm Search is able to attain relatively low error rates in classification without shrinking the size of the feature subset to its minimum.

1. Introduction

With the advances of information technology, it is not uncommon nowadays that data that are stored in database are structured by a wide range of attributes, with the descriptive attributes in columns and instances in rows. Often among the attributes, a column is taken as a target class. Therefore a classifier can be built by training it to recognize the relation between the target class and the rest of the attributes, with all the samples from the instances. This task is generally known as supervised learning or model training in classification. The model is induced by the values of attributes which are known as features because each one of these contributes to the generalization of the model with respective to its dimension of information.

For example, in the wine dataset [1] which is a popular sample used for benchmarking classification algorithms in machine learning community, thirteen attributes (features) of chemical analysis are being used to determine the origin of wines. Each one of these features is describing the origin of the wines which is the target class, pertaining to their respective constituents found in each of the three types of wines. The features essentially are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The features include variables of alcohol content, malic acid level, magnesium, flavonoids, phenols, proline, and color intensity. The multiple features essentially give rise to thirteen different dimensions of influences. In mutual consideration of these influences, a classifier is able to tell the type of wine when it is subject to an unknown sample (which again is described by these thirteen attributes). The multidimensions of these features are visualized in Figure 1. It can be seen that classification is a matter of computing the relations of these variables in a hyperspace to the respective classes; and of course the computation gets complicated as the amount of feature grows.

In machine learning, feature selection or input selection is often deployed to choose the most influential features

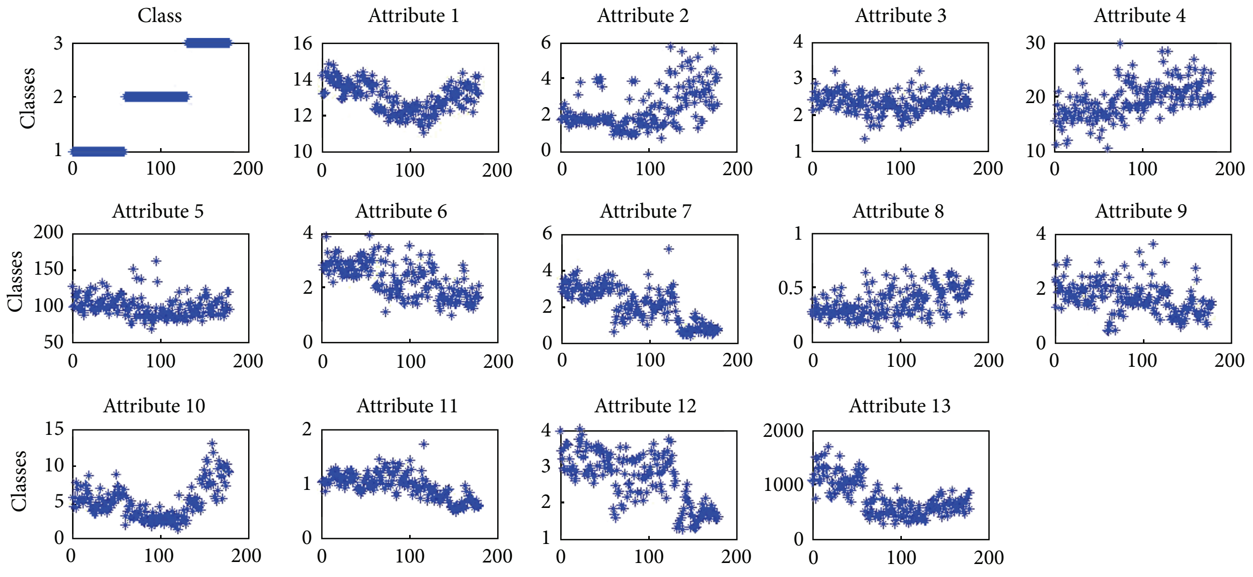

FIGURE 1: Visualization of multidimensions of features that describe the types of wine.

(inputs) from the original feature set for better classification performance. Feature selection is to find an optimal feature subset from a problem domain while improving classification accuracy in representing the original features. It can be perceived as an optimization process that searches for a subset of features which ideally is sufficient and appropriate to retain their representative power describes the target concept from the original set of features in a given data set. Through the process of feature selection, dimensions in the data are reduced, and the following advantages can be potentially attained: accuracy of the classifier is enhanced; resource consumption is lowered, both in computation and memory storage; it is easier for human to cognitively understand the causal relationship between features and classes with only few significant features.

In general, there is no best method known so far that can be unanimously agreed on among data mining researchers in implementing the optimum feature selection. Due to the nonlinear relations of the features and the concept targets, sometimes removing an attribute which has little correlation with the target variable (but unknowingly it is an important factor in other composite dependencies with the other attributes) may lead to worse performance. Tracing along the nonlinearity of the relation for evaluating the worthiness of the attributes is an exhaustive search in the high-dimensional space. Heuristics are often used to overcome the complexity of exhaustive search.

In the literature, many papers have reported on the use of heuristics to tackle the feature selection problems. However it is noticed by the authors that many proposed methods are limited to one or more of the following constraints in their designs. See Table 1 for details.

(1) The size of the resultant feature set is assumed to be fixed. Users are required to explicitly specify the maximum dimension as the upper bound of the feature set. Though this would significantly cut down the search time, the search is confined to only the combinations of feature subsets in the same dimensions. Assume that there is a maximum of k features in the original dataset. There are 2^k possible combinations of arranging the features in a subset of any variable size. With a bound s (for all $s \geq 1$) imposed on the size of the feature subset, it becomes a combinational problem of picking exactly s features into the subset from a total of k features. The number of combinations reduces from 2^k to $k!/(k-s)!s!$. The major drawback is that users may not know in advance what would be the ideal size of s.

(2) The feature becomes minimal. By the principle of removing redundancy, the feature set may shrink to its most minimal size. Features are usually ranked by evaluating their worthiness with respective to the predictive power of the classifier, and then eliminating those from the bottom of the ranked list. Again, there is no universal rule on how many features ought to be removed. Users may have to arbitrarily define a threshold value above which the ranked features form a feature set.

(3) The feature selection methods are custom designed for some particular classifier and optimizer. An optimizer is referred to some heuristics that explore the search space looking for the right feature set. As reported in most of the papers in the literature, a specific pair of classifier and optimizer is formulated out of many possible choices of algorithms.

The limitations above gave impetuses to the objective of research paper. We opt to design a flexible metaheuristic called Swarm Search that can efficiently find an optimal group of features from the search space; at the same time, there is no need for the user to fix the feature set size as Swarm Search would provide an optimal length of the feature set as well.

Table 1: A brief survey on feature selection models.

Reference	Classifier	Metaheuristic	No. of features	Fixed subset size	Domain
Talbi et al. [2]	SVM	PSO, GA	N/A	N/A	Gene microarray
Vieira et al. [3]	SVM	BPSO, GA	12, 28	No	SEPSIS data
Wang et al. [4]	RS	SS	14, 15	No	Credit scoring
Abd-Alsabour and Moneim [5]	SVM	ACO	17–70	No	General
Casado et al. [6]	DA	TS	54–121	Yes (5–8)	General
Jona and Nagaveni [7]	SVM	ACO, Cuckoo	78	Yes (5)	Mammogram
Unler et al. [8]	SVM	PSO	10–267	Min	General
Korycinski et al. [9]	BHC	TS	242	Yes (3–8)	Hyperspectral
Yusta [10]	N/A	GRASP, TS, MA	18–57	Yes (3–7)	General
Unler and Murat [11]	LR	PSO, SS, TS	8–93	Yes (3–8)	General
García-Torres et al. [12]	NB	TS	9–70	No	General
El Ferchichi and Laabidi [13]	SVM	TS, GA	24	No	Urban transport
Al-Ani [14]	ANN	ACO	40, 50	No	Speech, image

Legends: Particle swarm optimization (PSO), genetic algorithm (GA), support vector machines (SVM), binary particle swarm optimization (BPSO), scatter search (SS), rough set (RS), ant colony optimization (ACO), tubu search (TS), discriminant analysis (DA), binary hierarchical classifier (BHC), memetic algorithm (ma), greedy randomized adaptive search (GRASP), logistic regression (LR) classifier: naive Bayes (NB), Classifier, and Artificial neural network (ANN).

The mechanism of Swarm Search is universal in a sense that different classifiers and optimizers can just plug and play. As a comparative study, several popular classification algorithms coupled with several different most recent metaheuristic methods are integrated into Swarm Search in turn; their performances are evaluated over a collection of datasets of various multitudes of features. In essence the contribution of the paper is a Swarm Search feature selection method which can find optimal features as well as feature set size. The remaining paper is structured as follow. Some related feature selection methods are reviewed in Section 2. The Swarm Search method is described in detail in Section 3. Comparative experiments and their results are presented and discussed in Section 4. Section 5 at the end concludes the paper.

2. Related Work

In the past decades, many methods of feature selection have been proposed and studied extensively [15]. In general, there are three types of feature selection designs at the operation level and two types of feature searches at the data exploration level.

The three different operational designs are filter-based, wrapper-based, and embedded-based. Often features are evaluated individually in filter-based feature selection model. Features that are found redundant by means of some statistical computation are eliminated. Classification model that adopts filter-based approach evaluates and selects the features during the preprocessing step but ignores the learning algorithm. Hence the aspect of predictive strength of the classifier resulted from different combinations of features is not taken in account during the selection. These are simple selection procedures based on statistical criteria that can be found in a number of commercial data mining software programs. It was pointed out [16] that such methods are not very efficient; especially, when the data carry features, the maximum accuracy is rarely attained. Some typical measure criteria include information gain [17, 18] mutual information like redundancy relevance and class conditional interaction information, for measuring net relevance of features. Another classical filter method is correlation-based filter method (CFS) from [19] where correlation is used as selection criterion. CFS uses a scoring mechanism to rate and include those features that are highly correlated to the class attribute but have low correlation to each other. However, all the filter-based methods are considering relations among individual attributes and/or between pairs of attributes and target classes. The main drawback of this filter-type method is that it ignores the effect of the subset of features in the induction algorithm, as claimed in [20].

The other type of model is called wrapper-based model, which is popularly on par with the filter-type but generally yields better performance. Wrapper-based method works in conjunction with the induction algorithm using a wrapper approach [20]. The wrapper-based model selects features by considering every possible subset of them; the subsets of features are ranked according to their predictive power, while treating the inbuilt classifier as a black box. An improved version of wrapped-based model is proposed in [21]. It considers information of relevance and redundancy of the features in addition to their influences on the predictive power. The last type is embedded model, which in contrast to wrapper approaches, selects features while taking into account the classifier design [22]. This type is relatively rare for it involves specialized software design in order to enable its dual. Its popularity is hampered due to the complexity of the coding and implementation.

It is apparent that exhaustive search under all the wrapper-based and perhaps embedded approaches can be performed, if the number of feature is not too large. However, the problem is known to be NP-hard (Nondeterministic Polynomial Time), and the search quickly becomes computationally intractable. This is especially true for wrapper-based

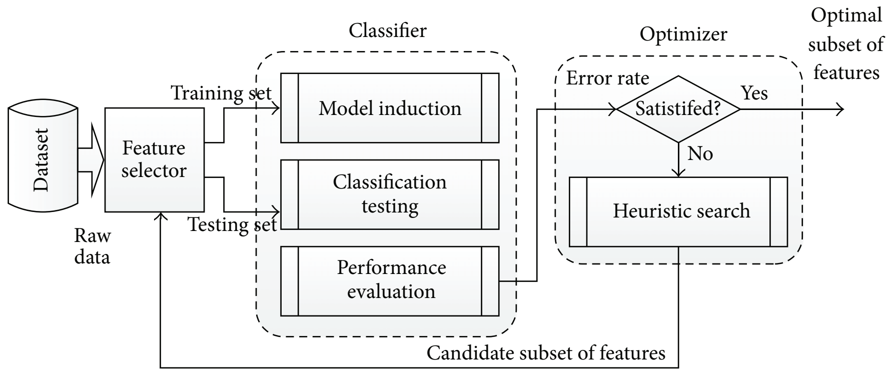

Figure 2: Process model of the proposed swarm search.

models that do rely on a classifier to determine whether a feature subset is good or not. The classifier is being induced continually as the model tries out a new feature subset in a vast search space. The repetition of classifier induction surely incurs high computational costs when the search of feature subsets is brute-force.

In order to alleviate the problem of intensive computation, metaheuristic search instead of brute-force search is augmented with certain classifiers in the wrapper-based models. Therefore, an optimal (reasonably good) solution is obtained in lieu of deterministic solution which may take forever to discover. In the literature, a number of contributions belong to this category of hybrid models. Not meant to be exhaustive, Table 1 lists some works in which metaheuristic algorithms are used together with classifiers, in a wrapper approach. In common they are proposed to be targeted at solving the high-dimensionality problems in feature selection by searching for optimal feature subset stochastically. However, some of the works assume that a user-input threshold value is used for defining the size of the feature subset, which was already mentioned as one of the disadvantages in Section 1.

Comparing to the earlier research in the 80s and 90s where mostly feature selection belonged to the filter-type focusing mainly on the pairwise relations between the features and the targets recent advances trend towards using wrapper methods to guide the feature selection based on the direct predictive influences by an augmented classifier. Different metaheuristics are used to generate an optimal solution. However, most of the works concentrate on a single type of classifier as well as one or a few specific metaheuristics only. As it can be seen from Table 1, the amount of original features that were being verified by their models in experiment, limit from 8 to 267, except may be the work [2] on microarray. These relatively small feature numbers may not be sufficiently for stress-testing the performance of the proposed algorithms. Lastly but importantly, the model designs in quite a number of works [6, 7, 9–11] are constrained by the need of inputting a pre-defined value for the length of the resultant feature subset; [8], on the other hand, programmed the selection to produce the least amount of features in a feature subset. Often the globally optimal solution, that yields the highest classification accuracy or other performance criterion by the fitness function, resides in feature subsets beyond this specific predefined subset length.

Given these shortcomings as observed from the literature review, it inspires to design an improved feature selection model that is flexible so that different classifiers and metaheuristic optimizer can be easily and compatibly interchanged; the optimal feature subset would be sought on the fly without the need of setting an arbitrary subset length and putting the model under tougher tests of more features (hence higher dimensionality).

3. Proposed Model

3.1. System Work Flow. The design of our model is wrapper-based that consists of two main components—classifier and optimizer as shown in Figure 2. Each of these components can be installed with the corresponding choice of algorithm at the user's will. Generically, a classifier can embrace software codes of SVM, ANN, decision tree, and so forth. optimizer can be implemented by any metaheuristic algorithm which searches for the optimal feature subset in the high-dimensional search space.

Over the original dataset which is formatted by all the original features vertically and the full set of instances horizontally, the feature selector module retrieves and cuts the data as per requested by the classifier module and optimizer module. The optimizer module signals the feature selector with the requested features, as resulted from its latest search. The feature selector then harvests the required data of only those feature columns and sends them to the classifier. The classifier uses a 10-fold cross validation method for dividing the corresponding dataset chosen by the feature selector, for training and testing a classifier model (in memory). Classification performance results such as accuracy % and/or other measures could be used as the fitness to the fitness function. In this case, the quality of the classifier model is regarded as the fitness function. As usual a convergence threshold can be set, so that if the fitness value falls below the threshold which is the stopping criterion, the iteration of the stochastic search would stop.

Overall, we can see that the process model of our proposed Swarm Search is comprised of two phases. In the first phase, which is called feature subset exploration and selection, the optimizer searches for the next best feature subset according to the logics of the metaheuristic algorithm installed at the optimizer module which selects the best subset. At the first run, the feature subset would be chosen randomly. In the second phase, which is called learning and testing, at the classifier module, a classifier model is induced from the training data, that is partitioned by feature selector with the best feature subset, and is tested on the test data.

The two phases are repeated continually until some stopping criterion is reached, either the maximum number of runs is reached or the accuracy of the classifier is assessed to be satisfactory. In each repetition, the relative usefulness of the chosen subset of features is evaluated and supposedly will improve in comparison to that of the previous round. The classifier here is taken as a black box evaluator, informing how useful each chosen subset of features is by its accuracy.

This wrapper approach by default could be brute-force by starting with the first feature subset and tests on all other possible subsets until the very last subset. Obviously this involves massive computation time, or even intractable. For only 100 features, there are $2^{100} \approx 1.2677 \times 10^{30}$ possible subsets for the classifier model to be repeatedly built and rebuilt upon. This high computation costs may further intensify if the training data has many instances.

In this regard, we propose to use coarse search strategies like those described in Section 3.1 for the Swarm Search model. Instead of testing on every possible feature subset, the Swarm Search which is enabled by multiple search agents who work in parallel would be able to find the most currently optimal feature subset at any time. It was however pointed out by Yang [23] that there is no free lunch in achieving the optimal feature subset. The feature subset may be the best so far the search has been outreached in the search space, but the absolute global best solution will not be known until all have been tried which may never occur because of the computational infeasibility.

3.2. Swarm Search. A flexible metaheuristic called Swarm Search (SS) is formulated for feature selection by combining any swarm-based search method and classification algorithm. Given the original feature set of size k, $A = \{a_1, a_2, \ldots, a_k\}$, SS supports constructing a classifier with classification accuracy rate e by providing an optimal feature subset $S \in A$ such that $e(A) \geq e(S)$, where S is one of all the possible subsets that exist in the hyperspace of A.

The core of SS is the metaheuristic algorithm that scrutinizes the search space for an optimal feature subset. Three metaheuristics are used in the experiments in this paper. They are particle swarm optimization (PSO), bat algorithm (BAT), and wolf search algorithm (WSA).

3.2.1. Metaheuristics Algorithms. PSO is inspired by the swarm behaviour such as fish, birds, and bees schooling in nature. The original version was developed by Kennedy and Eberhart in 1995 [24]. The term called Swarm Intelligence was coined since PSO became popular. The dual movement of a swarming particle is defined by its social sense and its own cognitive sense. Each particle is attracted towards the position of the currently global best g^* and its own locally best position l_i^* in history. While at the same time with the effect of the attraction, it has a tendency of moving randomly.

Let l_i and v_i be the location vector and velocity for particle i, respectively. The new velocity and location and location updating formulas are determined by

$$\begin{aligned} v_i^{t+1} &= v_i^t + \alpha\varepsilon_1\left[g^* - l_i^t\right] + \beta\varepsilon_2\left[l_i^* - l_i^t\right], \\ l_i^{t+1} &= l_i^t + v_i^{t+1}, \end{aligned} \tag{1}$$

where ε_1 and ε_2 are two random vectors, and each entry taking the values between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, which can typically be taken as [25], $\alpha \approx \beta \approx 2$.

BAT is a relatively new metaheuristic, developed by Yang in 2010 [25]. It was inspired by the echolocation behaviour of microbats. Microbats use a type of sonar, called echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depending on the species and often increases by using more harmonics. Three generalized rules govern the operation of the bat algorithm, and they are as follow.

(1) All bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers by instinct.

(2) Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$, varying wavelength λ, and loudness A_0 to search for prey. They can automatically adjust the wavelength of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target.

(3) Although the loudness can vary in many ways, we assume that the loudness varies from a large and positive A_0 to a minimum constant value $A_{\min}$.

The new solution x_i^t and velocities v_i^t at time step t are given by

$$\begin{aligned} f_i &= f_{\min} + \left(f_{\max} - f_{\min}\right)\beta, \\ v_i^t &= v_i^{t-1} + \left(x_i^t - x_*\right) f_i, \\ x_i^t &= x_i^{t-1} + v_i^t, \end{aligned} \tag{2}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution. Here x_* is the current global best location which is located after comparing all the solutions among all the n bats. As the product $\lambda_i f_i$ is the velocity increment, we can use either f_i or λ_i to adjust the velocity change while fixing the other factor. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{\min}, f_{\max}]$. For local search, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t, \tag{3}$$

where $\varepsilon \in [-1,1]$ is a random number, while $A^t = \langle A_i^t \rangle$ is the average loudness of all the bats at this time step. The update of the velocities and positions of bats have some similarity to the procedure in the PSO as f_i essentially controls the pace and range of the movement of the swarming particles. The loudness and rate of pulse emission have to be updated accordingly as the iterations proceed. As the loudness decreases once, a bit has found its prey, while the rate of pulse emission increases. Consider

$$\begin{gathered} A_i^{t+1} = \alpha A_i^t, \qquad r_i^{t+1} = r_i^0 \left[1 - \exp(-\gamma t)\right], \\ A_i^t \longrightarrow 0, \quad r_i^t \longrightarrow r_i^0, \quad \text{as } t \longrightarrow \infty. \end{gathered} \tag{4}$$

WSA [26] is one of the latest metaheuristic algorithms by the authors; its design is based on wolves' hunting behavior. Three simplified rules that govern the logics of WSA are presented as follow.

(1) Each wolf has a fixed visual area with a radius defined by v for X as a set of continuous possible solutions. In 2D, the coverage would simply be the area of a circle by the radius v. In hyperplane, where multiple attributes dominate, the distance would be estimated by the Minkowski distance, such that

$$v \le d(x_i, x_c) = \left(\sum_{k=1}^{n} \left| x_{i,k} - x_{c,k} \right|^{\lambda} \right)^{1/\lambda}, \quad x_c \in X, \tag{5}$$

where x_i is the current position, x_c are all the potential neighboring positions near x_i and the absolute distance between the two positions must be equal to or less than v, and λ is the order of the hyperspace. For discrete solutions, an enumerated list of the neighboring positions would be approximated. Each wolf can only sense companions who appear within its visual circle, and the step distance by which the wolf moves at a time is usually smaller than its visual distance.

(2) The result or the fitness of the objective function represents the quality of the wolf's current position. The wolf always tries to move to better terrain, but rather than choosing the best terrain it opts to move to better terrain that already houses a companion. If there is more than one better position occupied by its peers, the wolf will chose the best terrain inhabited by another wolf from the given options. Otherwise, the wolf will continue to move randomly in Brownian motion.

The distance between the current wolf's location and its companion's location is considered. The greater this distance is, the less attractive the new location becomes, despite the fact that it might be better. This decrease in the wolf's willingness to move obeys the inverse square law. Therefore, we get a basic formula of betterment:

$$(r) = \frac{I_o}{r^2}, \tag{6}$$

where I_o is the origin of food (the ultimate incentive) and r is the distance between the food or the new terrain and the wolf. It is added with the absorption coefficient, such that using the Gaussian equation, the incentive formula is

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

where β_o equals I_o.

(3) At some point, it is possible that the wolf will sense an enemy. The wolf will then escape to a random position far from the threat and beyond its visual range. The movement is implemented using the following formula:

$$x(i) = x(i) + \beta_o e^{-r^2} \left(x(j) - x(i)\right) + \text{escape}(), \tag{8}$$

where escape() is a function that calculates a random position to jump to with a constraint of minimum length, v, x is the wolf which represents a candidate solution, and $x(j)$ is the peer with a better position as represented by the value of the fitness function. The second term of the above equation represents the change in value or gain achieved by progressing to the new position. r is the distance between the wolf and its peer with the better location. Step size must be less than the visual distance.

Given these rules, we summarize the WSA in pseudocode as in Algorithm 1.

3.2.2. Searching for Optimal Feature Subsets. Each search agent in the metaheuristic (PSO/BAT/WSA) is coded as a solution vector which contains a list of feature indices that form a feature subset. The length of a solution vector is variable and it represents the length of the feature subset. At initialization, the vector length is randomized for each search agent. So they have different lengths to start with, in a way similar to placing the search agents in random dimensions of the search space. Each search agent has a variable k, which is the current length of feature subset it represents, where $1 \le k \le K$. K is a constant that is the maximum cardinality of the search space that is also the maximum number of the original features. During the Swarm Search, the search agent steps in the same dimension in each local step movement. The agent explores feature subsets in same dimension. It will only change its dimension at exceptional events. The exceptional events are similar in context to the escape function in WSA, mutation in GA, where a search agent suddenly deviates largely from its local search area (dimension). The exceptional events are usually programmed to happen randomly with

```
Objective function f(x), x = (x_1, x_2,..., x_d)^T
Initialize the population of wolves, x_i (i = 1, 2, ..., W)
Define and initialize parameters:
r = radius of the visual range
s = step size by which a wolf moves at a time
α = velocity factor of wolf
p_a = a user-defined threshold [0..1], determines how frequently an enemy appears
WHILE (t < generations && stopping criteria not met)
  FOR i = 1 : W  // for each wolf
    Prey_new_food_initiatively();
    Generate_new_location();
    // check whether the next location suggested by the random number generator is new. If
not, repeat generating random location.
    IF (dist(x_i, x_j) < r && x_j is better as f(x_i) < f(x_j))
      x_i moves towards x_j // x_j is a better than x_i
    ELSE IF
      x_i = Prey_new_food_passively();
    END IF
    Generate_new_location();
    IF (rand () > pa)
      x_i = x_i + rand () + v; // escape to a new pos.
    END IF
  END FOR
END WHILE
```

Algorithm 1: Pseudocode for the WSA algorithm.

a user-defined probability. At the occurrence of such events, the k variable changes randomly to other value. In other words, the search agent "jumps" out of its local proximity to another dimension of the search space. For example, an agent has the current dimension $k = 4$; in the next iteration k may change to 5, 9785 or any other dimension when an exceptional event occurs. Or else, it would remain the same dimension and explore other combinatorial subsets in the same dimension. The vector of a search agent with $k = 4$ may take a sample form $\langle 1, 2, 5, 8 \rangle$ that means that the vector consists of the 1st, 2nd, 5th, and 8th features as the representative indices. A sample snapshot of a search agent exploring the search space and its possible moves is shown in Figure 3.

In our design, search agents normally explore in the same dimension in each movement. This is shown as the grey arrows for the case of WSA, hunting for food (looking for a feature subset that produces higher accuracy) randomly in the same dimension. In this way, it only can achieve a local optimal in the same dimension. Because the agents have the tendency to swarm towards their neighbour that have a better fitness function, an active agent (a_1, a_3) may merge to its peer agent (a_1, a_3, a_4) provided that the peer agent is having a better fitness.

At times, a dimension change function gets activated by the randomizer; the active agent, for example, may "jump" out of its dimensions to another dimension. The random function rand() is a standard Matlab function which produces uniformly distributed pseudorandom numbers. Another random function is to compute about how far the dimension displacement can take, with the direction of (+/−) randomly generated too. The length of jump, however, cannot exceed beyond the boundary. For instance, the active search agent is now at (a_1, a_3), it cannot fall beyond dimension 1. Likewise it can never go beyond the maximum dimension K.

The operation of the optimizer called Swarm Search takes the following steps.

(1) Initialize search agents. For each search agent, randomly assign a different feature combination as a subset.

(2) Calculate the fitness of the objective function for each agent, rank the agents by their fitness values, and record the one that has the highest fitness as the most promising solution.

(3) Perform local search and optimize the current best solution. This step may vary slightly for different metaheuristic algorithms. However, generaly, it tries to update the promising solution unless some satisfactory terminal condition is met. In PSO, the swarming particles have velocities. So we need not only to update their positions, but also to update their velocities. Recording the best local solution and global solution in each generation is required. In BAT, for each generation, rank the current best position according to the pulse rate and loudness; then update the velocity and position for each agent. In WSA, each search agent is updated by three intermediate procedures: first randomly move to update the position; second, check if they are ready to swarm towards to the closest and best companions; third, activate the escape function at random-relocate agents that are qualified to jump dimensions.

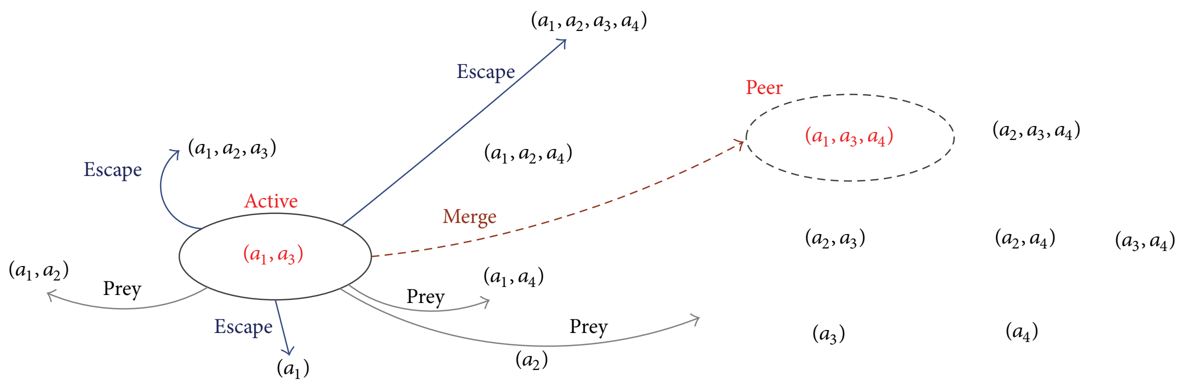

Figure 3: An illustration of possible moves of a search agent in the search space.

Table 2: List of feature selection methods used in experiments.

Method abbreviation	Classifier	Metaheuristic	Approach
FS-PSO-PN	Pattern Network	Particle swarm optimization	Swarm
FS-PSO-DT	Decision Tree	Particle swarm optimization	Swarm
FS-PSO-NB	Naïve Bayes	Particle swarm optimization	Swarm
FS-BAT-PN	Pattern Network	Bat algorithm	Swarm
FS-BAT-DT	Decision Tree	Bat algorithm	Swarm
FS-BAT-NB	Naïve Bayes	Bat algorithm	Swarm
FS-WSA-PN	Pattern Network	Wolf search algorithm	Semiswarm
FS-WSA-DT	Decision Tree	Wolf search algorithm	Semiswarm
FS-WSA-NB	Naïve Bayes	Wolf search algorithm	Semiswarm
FS-Cfs-PN	Pattern Network	Correlated-based filter	Filter
FS-Cfs-DT	Decision Tree	Correlated-based filter	Filter
FS-Cfs-NB	Naïve Bayes	Correlated-based filter	Filter

(4) Change dimension randomly.

(5) Check if the terminating criteria are met; if not go back to step (2).

(6) Terminate and output the result.

4. Experiment

The Swarm Search feature selection is designed for achieving the following merits (1) low error rate for the classifier as a feature selection tool to harvest the optimal feature subset out from a prohibitively large number of subsets; (2) reasonable time for the convergence of the metaheuristic search when compared to brute-force exhaustive search (whose time-taken we can safely assume to be extremely long); (3) an optimal size of feature subset that leads to optimal accuracy for the particular classifier; and (4) its universality in working across different classifiers and metaheuristic optimizers. Concerning the proposed four merits, experiments by computer simulation are then carried out accordingly. We target to evaluate the performance with respect to the first three virtues: (i) average error rate which is simply the number of incorrectly classified instances divided by the total number of testing instances; (ii) the total time consumed in finding the optimal feature subset including the training and testing of the respective classifier, in unit of seconds; and (iii) the length of the feature subset as the percentage of the maximum dimension of the original features.

In order to validate the universality of the proposed Swarm Search model, three popular classification algorithms have been chosen to be the kernels of the Classifier, and three contemporary metaheuristic methods are utilized as the core optimizer to perform the Swarm Search exploration. Furthermore, we use an industrial standard feature selection method called *correlation-based filter selection* (Cfs) for as a baseline benchmark for comparing with the proposed methods. Total there are twelve different feature selection methods that are programmed and tested in our experiments; see Table 2.

The classification algorithms as mentioned in Table 2 have been defined and studied in [27]. For the metaheuristic methods, readers who want to have the full details are referred to [24] for PSO, [25] for BAT, and [26] for WSA. Both PSO and BAT have substantial swarming behaviour for their movements that are governed with velocity; a single leader who has the greatest attractiveness is among the agents the remaining agents follow the leader's major direction, while at the same time they roam locally. In contrast, the agents in WSA are autonomous; they do merge nevertheless only when they come in contact with their visual range. So

Table 3: Datasets that are used in feature selection experiments.

Dataset	Attributes	Instances	Abstract/URL
Lung Cancer	56	32	The data described 3 types of pathological lung cancers http://archive.ics.uci.edu/ml/datasets/Lung+Cancer
Heart Disease	75	303	The data refers to the presence of heart disease in the patient http://archive.ics.uci.edu/ml/datasets/Heart+Disease
Libra Movement	91	360	The data refers to hand movement type in LIBRAS, Brazilian signal language http://archive.ics.uci.edu/ml/datasets/Libras+Movement
Hill Valley	101	606	Either a Hill (a "bump" in the terrain) or a Valley (a "dip" in the terrain) http://archive.ics.uci.edu/ml/datasets/Hill-Valley
CNAE-9	875	1080	Nine categories of free text business descriptions of Brazilian companies http://archive.ics.uci.edu/ml/datasets/CNAE-9

their approach is type of semiswarm. Cfs [19] evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low intercorrelation are preferred. The operation of Cfs is filter-based and sequential, hence no swarm.

4.1. Experimental Data and Setup. The twelve feature selection methods with combinations of different classifiers and optimizers are put under test across five well-known dataset from UC Irvine Machine Learning Repository (short for UCI). UCI is popular data archive for benchmarking the performance of machine learning algorithms. The five datasets that are chosen contain various numbers of features, ranging from 56 to 875. They are Lung Cancer, Heart Disease, Libra Movement, Hill Valley, and CNAE-9. Details of these datasets are outlined in Table 3. In addition to a variety of attribute numbers for stress testing the feature selection algorithms, the five datasets are chosen with purposes. Lung cancer is solely a categorical classification problem, where all predictive attributes are nominal, taking on integer values 0–3. The data described 3 types of pathological lung cancers. However this dataset is having relatively few instances available for model training in respective to attributes, with the ratio of attributes-to-instances 56 : 32. It represents a tough disproportional problem in training an accurate classification model. The dataset Heart Disease represents a mixed multivariate data-type problem. Libra Movement dataset represents a challenging hand movement recognition problem, where the data attributes are coordinates, solely numeric, and the data are in sequential order as they were extracted from video frames. Similarly, Hill Valley is a purely numeric and sequential time-series. Sampling at regular intervals over the x-axis, a variable value fluctuates on y-axis over time. An accurate prediction model for recognizing whether the sequential points are forming a hill or valley is known to be hard to build in the context of pattern recognition with a single sequential variable. CNAE-9 is extremely challenging, because of the large number of attributes that resulted from the word frequency counts. The words and their combinations that appear in each document may have subtle relations to the nature of the document. All these five chosen datasets present different levels and perspectives of challenges in constructing an accurate classification model and hence in testing the efficacy of each feature selection algorithm.

The main performance indicator in the experiments is classification error rate which is also used as stop criterion. The Swarm Search is programmed to explore and rank the solution continually until the difference of the average error rate of the current iteration and that of the recorded best are less than 1.0×10^{-8}. The other criterion is the maximum iteration which is set arbitrarily at 100,000,000 cycles. The maximum run cycle is to prevent the Swarm Search from running exceedingly long. The dataset, CNAE-9 with the greatest amount of features 875, potentially has 875 dimensions in the search space, and this amounts to 2^{875} ($\approx 2.519 \times e^{+263}$) combinatorial feature subsets. This enormously large number obviously prevents any normal computer from an exhaustive search. Even using Swarm Search, it will take a very long time indeed, while a deterministic solution is not guaranteed.

For the optimizer, default parameter settings are adopted for the metaheuristic methods; the default values are shown in Table 4.

We have to be very careful in choosing the parameters as they are sensitive to performance. For different metaheuristic methods, the parameters vary. In BAT, the bat position is influenced by the bat's echo loudness and echo pulse rate, so adjusting the loudness and emitted pulse depends on the proximity of their target. Here, we use 0.5 as their values, which shall result in a balance of intensification and exploration. The PSO has two learning factors, c_1, c_2. Generally, $c_1 = c_2$ and the value is smaller than 2 as this has been widely used in the literature. In our experiment, we use 1.5 which is moderately less than 2. For WSA, each search agent has a visual circle and the visual distance is the visual radius. In WSA, there is an escape function that simulates how a wolf runs far away from the current position when danger is encountered. The jump over is assumed farther than its visual radius. In our experiment, we set 5 as the escape distance that is five times larger than

Table 4: Parameter settings for metaheuristic methods.

FS using BA		FS using PSO		FS using WSA	
Parameter	value	Parameter	value	Parameter	value
A (loudness)	0.5	Populations	5	Visual distance	1
R (pulse rate)	0.5	c_1	1.5	Escape distance	5
Populations	5	c_2	1.5	Escape probability	0.25
$Q_{\min}$	0			Population	5
$Q_{\max}$	0.2				

Table 5: Comparative results in classification error for all the classification algorithms with Cfs feature selection and swarm search feature selection and without any feature selection.

Algorithm	Lung Cancer (56)	Heart Disease (75)	Libra Movement (91)	Hill Valley (101)	CNAE (875)
PN	0.5938	0.4653	0.1944	0.4447	0.8741
DT	0.5	0.4719	0.3028	0.505	0.112
NB	0.375	0.4422	0.3722	0.5149	0.0685
Cfs-PN	0.3125	0.462	0.1972	0.5017	0.187
Cfs-DT	0.3438	0.4818	0.2583	0.4951	0.1889
Cfs-NB	0.2813	0.4224	0.3528	0.4884	0.1824
PSO-PN	0.125	0	0.075	0.3465	0.0083
PSO-DT	0.225	0	0.2806	0.2112	0.0407
PSO-NB	0.1513	0.105	0	0.0495	0.8889
BAT-PN	0.1188	0	0.0833	0.1997	0.037
BAT-DT	0.21	0	0.3444	0.2442	0.0491
BAT-NB	0.1625	0.27	0.1028	0.5132	0.8889
WSA-PN	0.1338	0	0.0694	0.1997	0.037
WSA-DT	0.0407	0	0.2028	0.2178	0.037
WSA-NB	0.1575	0	0	0.1151	0.8889

the visual range. In addition, the escape probability is the probability of encountering danger. For a fair comparison, each metaheuristic method has 5 searching agents which are supposed to run in parallel. They are however programmed to run in sequential on a normal computer (instead of a parallel computer); the five agents move stochastically in round robin under the optimization loop. The programs are coded in Matlab R2010b. The computing environment is a MacBook Pro (with CPU: 2.3 GHZ, RAM: 4 GB).

4.2. Experimental Results. The experiments are mainly grouped into three kinds, testing for the performance of feature selection in classification error, in time consumption, and in amount of features obtained in the feature subset. For classification errors, Table 5 tabulates the results for classifiers using algorithms of Pattern Network (PN), which is also known as Artificial Neural Network, Decision Tree (DT), and naïve Bayes (NB), as well as the results that produced with their feature selection counterparts. Around the five different datasets, the results pertaining to classification errors, are plotted in radar charts visually in Figures 4, 5, and 6, respectively. Likewise, the results for consumption time are in Table 6 and Figures 7, 8, and 9. Table 7 and Figures 10, 11, and 12 are for feature amount in feature subgroup. Please note that for clarify in radar charts, the feature amounts are normalized to percentage with respect to the original feature

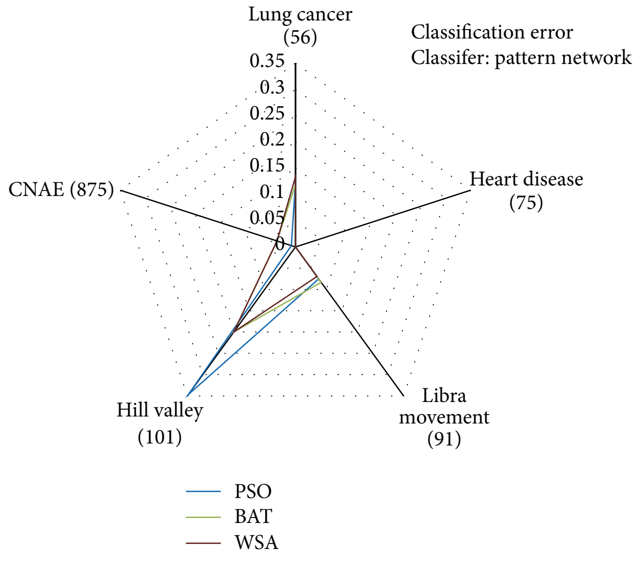

Figure 4: Results in classification error for Pattern Network are shown visually in radar chart.

numbers when charted. While the tabulated results allow comparison between swarm search and no swarm search at easy glance, the radar charts show individually the efficacy of

Table 6: Comparative results in time consumption (in seconds) for all the classification algorithms with Cfs feature selection and swarm search feature selection and without any feature selection.

Algorithm	Lung Cancer (56)	Heart Disease (75)	Libra Movement (91)	Hill Valley (101)	CNAE (875)
PN	3.07	2.35	18.08	58.73	5066.72
DT	0.02	0.08	0.32	0.16	3.39
NB	≈0	0.01	0.05	0.08	0.26
Cfs-PN	0.18	≈0	3.79	0.3	10.51
Cfs-DT	≈0	0.01	0.04	≈0	0.08
Cfs-NB	≈0	1.26	0.02	≈0	≈0
PSO-PN	853.23123	382.3217	1714.34495	921.28525	70538.3472
PSO-DT	177.3407	11.9377	232.6246	491.7232	2886.9232
PSO-NB	22.9799	3.92775	82.7123	34.344	2358.2769
BAT-PN	1532.321	347.0041	686.557	524.367	26365.1231
BAT-DT	179.1062	16.5652	240.737	448.7332	2341.6668
BAT-NB	25.8876	2.436514	28.6922	31.001802	1441.7241
WSA-PN	1521.8085	1305.6519	3005.4028	1159.367	26365.1231
WSA-DT	1521.8085	49.2341	106.6606	176.6559	82688.1231
WSA-NB	42.2628	23.3551	404.6113	24.7167	236365.123

Table 7: Comparative results in feature amount in feature subgroup for all the classification algorithms with Cfs feature selection and swarm search feature selection and without any feature selection.

Algorithm	Lung Cancer (56)	Heart Disease (75)	Libra Movement (91)	Hill Valley (101)	CNAE (875)
PN/DT/NB	56	75	91	101	875
Cfs-PN/DT/NB	11	9	26	1	28
PSO-PN	41	21	75	96	853
PSO-DT	39	44	66	79	854
PSO-NB	37	18	19	9	76
BAT-PN	41	17	39	88	849
BAT-DT	29	50	79	84	781
BAT-NB	38	5	6	43	243
WSA-PN	41	21	45	30	849
WSA-DT	41	23	40	44	851
WSA-NB	35	19	20	82	323

each swarm search variant with respect to the five datasets in visual comparisons.

4.3. Result Analysis. The classification error rates vary a lot across different datasets and with different classifiers. For instance, perfect accuracy is achieved for Heart Disease dataset with all three different optimizers and three different classifiers, except PSO-NB and BAT-NB where their accuracies are 89.5% and 73%, respectively. Moderate error rates are observed for Lung Cancer dataset, but relatively high error rate is incurred on the remaining datasets like Libra Movement, Hill Valley, and CNAE. In one extreme case, for CNAE dataset, Naïve Bayes classifier suffers at almost 89% error rate in all situations of optimizers, but over the same CNAE dataset, classifiers of Pattern Network and Decision Tree can achieve very low error rate less than 5%. This extreme performance is shown in Figure 6. As CNAE represents the worst case of highest dimensionality, Pattern Network classifier managed to conquer the problem of high dimensionality quite well especially with PSO optimizer (less than 1% error); See Figure 4. It works quite well with BAT and WSA too (at 3.7% error) considering that the dataset is text mining in nature and there are many variables for determining the category of documents that the text would belong to. Naïve Bayes apparently is rated down upon this CNAE dataset. However, it works well for the rest except Hill Valley which is basically time-series dataset with 101 data points forming a shape of hill or valley. The hills and valleys in the time series form nonlinear recursion which is post-nonlinear combination of predecessors. The classifier needs to recognize the shape by processing the values of the data points. The mathematical challenge is due to the problem of post-nonlinear multidimensional data projection. Naïve Bayes did not do well with Bat but otherwise with PSO (<5% error) and with WSA (<12% error), as shown in Figure 6. An interesting phenomenon is observed here, that BAT seems work quite well with Pattern Network and Decision Tree with relatively low or moderate errors. But in general BAT does not perform well with Naïve Bayes, especially over datasets that are very unstructured in nature like text mining and

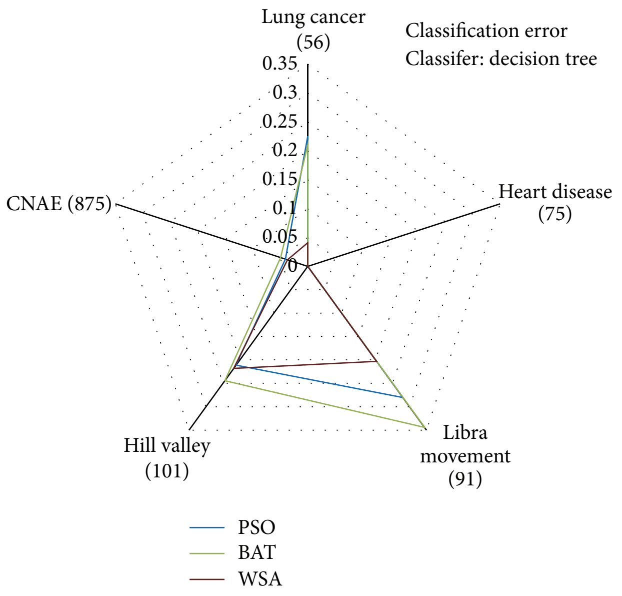

Figure 5: Results in classification error for Decision Tree are shown visually in radar chart.

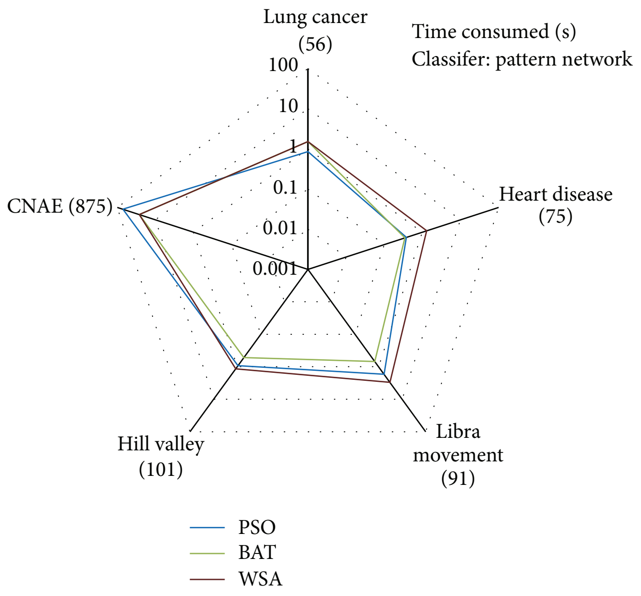

Figure 7: Results in consumption time for Pattern Network are shown visually in radar chart.

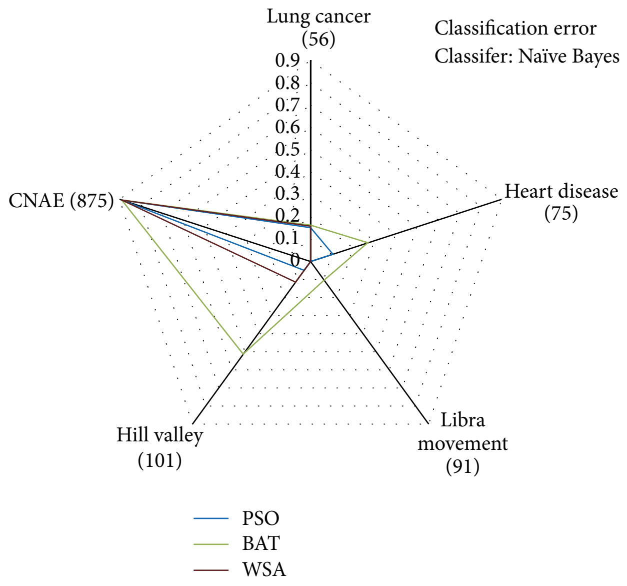

Figure 6: Results in classification error for Naïve Bayes are shown visually in radar chart.

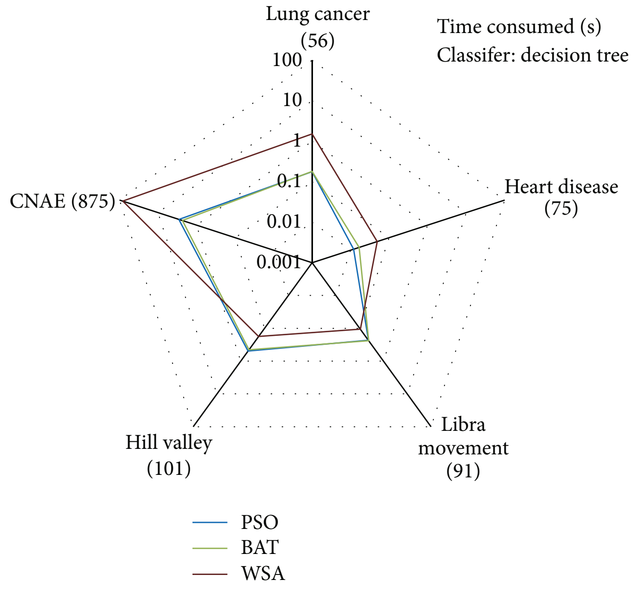

Figure 8: Results in consumption time for Decision Tree are shown visually in radar chart.

time-series data points. This may be due to the algorithmic characteristic of BAT where it may become too intensified in local search and trapped in some local optima; hence low quality feature subsets were found as solutions. PSO seems to have the same problem too, but its strong swarming ability might have led the search out local optima relatively easily, for example, Hill Valley with Naïve Bayes, and the same with DT where PSO shows its superiority by swarming out of the local optimal time-series data points. In general, WSA seems to work best with Naïve Bayes, except for CNAE. Nevertheless for relatively well-structured datasets like Lung Cancer and Heart Disease, in general, all the three classifiers and the tree optimizers generate good results. For time-series data like Hill Valley and Naïve Bayes with PSO and WSA are good choices. For text mining like CNAE that has huge dimensionality, Pattern Network is superior regardless of what optimizer is to be used; this may be attributed to the high capacity of nonlinearity of a neural network where the relations in the dimensional data variables can be modeled well in layers of complex neurons and their weights. On the other hand, the Swarm Search methods do not work well with Naïve Bayes as classification error is more than 80%

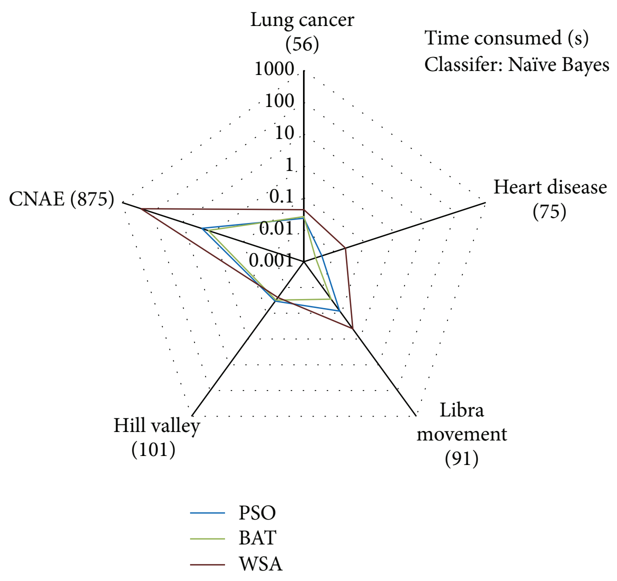

FIGURE 9: Results in consumption time for Naïve Bayes shown visually in radar chart.

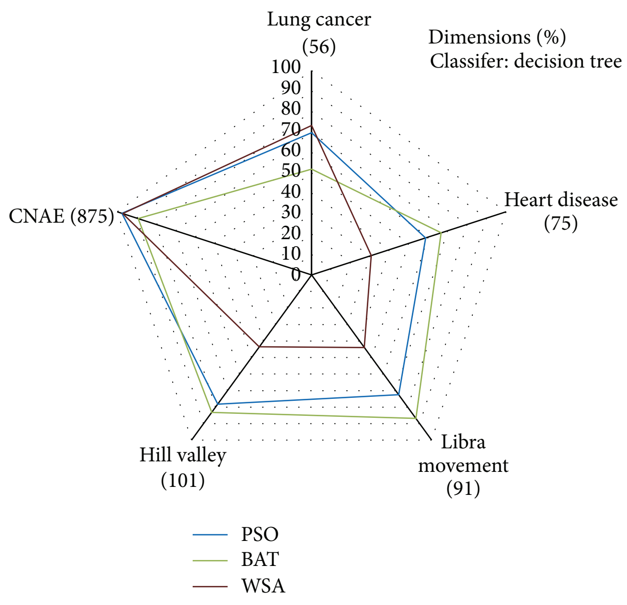

FIGURE 11: Results in % dimension for Decision Tree shown visually in radar chart.

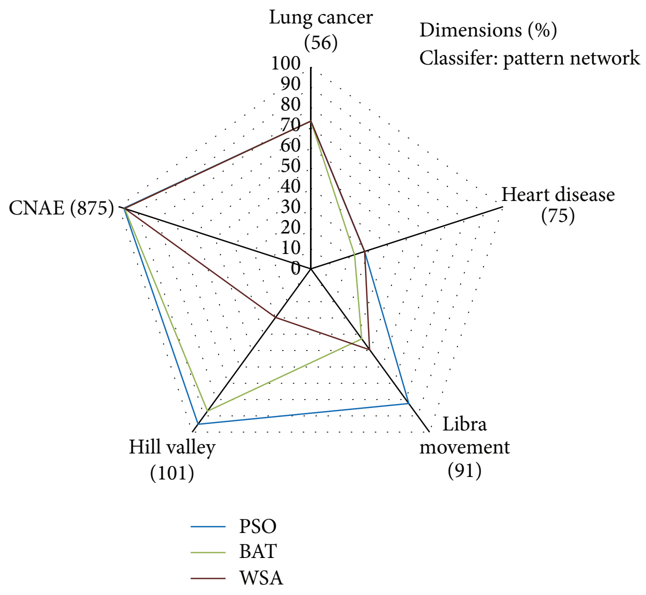

FIGURE 10: Results in % dimension for Pattern Network are shown visually in radar chart.

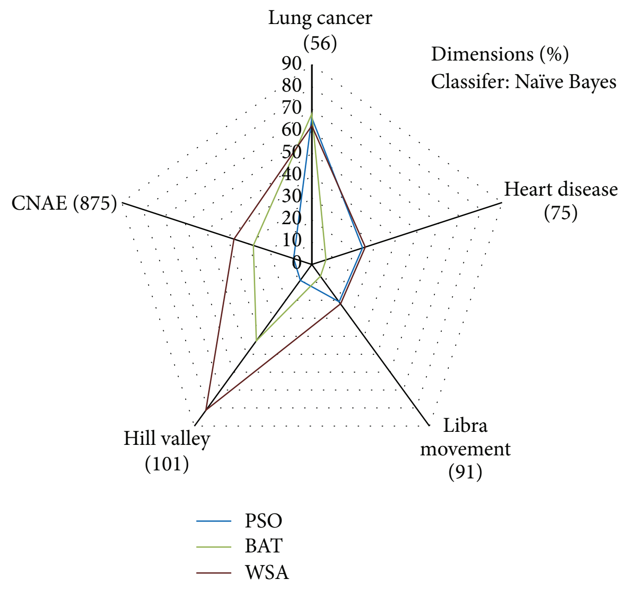

FIGURE 12: Results in % dimension for Naïve Bayes shown visually in radar chart.

for CNAE. That is basically because Naïve Bayes classifier has an assumption where all the attributes are independent. However for the data of which the attributes represent, it is not actually the case. Attributes for CNAE indeed have contextual relations though they may be very subtle and very nonlinear. Hence, the actual relations between attributes are never random; for example, certain frequently appearing words in a document follow some semantic grammars. Words and sentences do exist together in some sequential orders. Swarm Search however operates in stochastic manner where the order of the features in the subset is randomized in each move, so is the change of feature subset size at random intervals.

As shown in Figure 5, Decision Tree may not be a good candidate in Swarm Search as it incurs high error rates in three datasets such as Lung Cancer, Libra, and Hill Valley, though it can overcome the high-dimensionality problem imposed by CNAE. However, in Table 5, PSO-DT has lower error rate than PSO-PN, and DT works well for CNAE for all Swarm Search methods. As shown in Figures 7, 8, and 9, the time consumption for the combined process of feature selection and classifier induction generally increases as the amount of attributes and instances increases. Generally, the time taken is highest for Pattern Network and shortest for Naïve Bayes, which is quite common as it is well know that neural network takes a long time to train. While PSO and BAT took about the same and reasonable time in processing every dataset, WSA shows its disadvantage prominently in CNAE and Libra. It implies that in search space that represents data of very high dimensionality, WSA takes exceptional long time to converge. This is clearly due to its semiswarm characteristics; the search agents are actively out exploration in wide dimensions, and they hardly would converge until a global optimum is found which may be rare. The unusually long time is observed at processing Lung Cancer and Heart disease datasets as well; these datasets are considered relatively easy and lead to fast convergence for the other optimizer but not for WSA, as shown in Figure 8. But for time-series data like Hill Valley, WSA can perform on-par or even outperform the other optimizers despite the fact that Hill Valley has high 101 dimensions. The data points in the time series have relatively low variation and high continuation, and the search agents by WSA are able to track along the feature subsets because the agents rely much on their visual range to swarm and search. BAT, in general, is fast and consumes the least amount of time in all the datasets. PSO nevertheless took slightly longer time than WSA in processing CNAE and Hill Valley when the classifiers Pattern Network and Decision Tree are used respectively.

Figures 10, 11 and 12 essentially show about how much in percentage of the size of the original feature set that the selected optimal feature subset contain. In other words, this is the length of the selected feature subset in percentage of the maximum dimension of the original feature set. For example, if there are originally 100 features in the data and the optimal feature subset is found to be $\langle a_x, a_y, a_z \rangle$, the *% dimension* is 3% for the optimal feature subset. As it was pointed out in Section 2, sometimes it may not be desirable to obtain the very minimum number of features but rather the sufficient amount of features in the right dimension that contribute to the greatest accuracy. This is based on a fair assumption that the best solution may not always reside in the low dimensions. Reaching out high to high dimension is one of the important abilities for Swarm Search.

As it can be shown in the results, classifiers Pattern Network and Decision Tree generally can retain most of the feature lengths (>50%); but the methods with Naïve Bayes tend to concentrate on solutions in the lower dimensions for finding the resultant feature subset. But there is an exception in Figure 12, WSA paired with Naïve Bayes retains over 80% of the maximum dimension in the Hill Valley dataset. It shows that this particular combo is able to reach out to a very high dimension for obtaining the optimal feature subset. It does not mean however that the absolute optimum solution must be there, because verifying so would be computationally intractable. PSO + NB on the other hand in Figure 12 acquires its solutions in relatively low dimensions in datasets of Heart Disease, Libra, and Hill Valley. Similar but not so obviously the same phenomenon is observed with Decision Tree in Figure 11, and in contrast PSO can acquire solutions at high dimension with Pattern Network in Figure 10. This demonstrates an interesting phenomenon that PSO being a strongly swarming method, when equipped with very nonlinear classifiers like Pattern Network and Decision Tree, its search agents can swarm their ways far and wide to high dimensions for scouting for a solution. On the other hand, when paired with probabilistic model like Naïve Bayes, PSO does not swarm far and wide like how it did with nonlinear functions. The same phenomenon is observed in BAT as well; BAT swarms relatively well in Decision Tree and Pattern Network, except for the Libra dataset in Figure 10. In fact, BAT did not swarm out far with Pattern Network in Figure 10, but it did so in superior with Decision Tree in Figure 11. With Decision Tree, BAT also beats the other optimizers in datasets like Heart Disease, Libra, and Hill Valley, in outreaching to space of high dimensions. See Figure 11. Perhaps the decision rules in Decision Tree model guide the balance of intensification and exploration well during the navigation of the agents in the search space for the BAT operation. In conclusion, WSA relatively does not reach out high because it does not swarm well in Pattern Network and Decision Tree, but it did extremely well in time-series data, Hill Valley when paired with Naïve Bayes, which is known as coarse strategy.

The results over different datasets are now aggregated for comparison of feature selection method groups. In detail, the results are tabulated in Table 8. The corresponding bar charts that show each performance aspect (error, time and % dimension) are shown in Figures 13, 14, and 15 for easy comparison. The error performance and % dimension of correlation-based feature selection method (Cfs) are included as well, as benchmarking references in comparison.

Obviously, it can be seen that, in Figure 13, the errors by Cfs are very high compared to those of all the other swarm-based methods. As a general trend, Pattern Network yields the lowest error rate given its complex and nonlinear weight distribution in its interconnected neuron networks. It seems to be able to handle classification problems very well. Second is Decision Tree and third is Naïve Bayes. Overall, as in Figure 13, WSA yields relatively lowest error among the other optimizers. But these benefits of low errors are out-weighted by the very high time consumption as shown in Figure 12 for WSA. Regarding the classifier algorithm, Naïve Bayes when paired with WSA has a very high time consumption; it shows that the searching agents in WSA have difficulty in converging when Naïve Bayes is being used, while, at the same time, a high cost in error rate too for WSA-NB exists. This may be due to the semiswarm behavior in WSA. But paring Naïve Bayes with strong swarming optimizers like PSO and BAT, they converge more quickly. As shown in Figures 13 and 14, however, pairing Pattern Network with PSO incurs a much long computation time than pairing with BAT or WSA. That is when a highly nonlinear classifier meets the most swarming optimizer. Nevertheless, this pairing does not offer proportionally good error rate either. It is worth mentioning that Cfs executes extremely fast (<1 second), and relatively classifiers such as Decision Tree and Naïve Bayes when coupled with PSO and BAT work very fast too. The exception is WSA which is quite slow and it was already mentioned.

Table 8: Comparison of individual performances of different algorithm combinations.

		FS-PSO			FS-BAT			FS-WSA			FS-Cfs	
	FS-PSO-PN	FS-PSO-DT	FS-PSO-NB	FS-BAT-PN	FS-BAT-DT	FS-BAT-NB	FS-WSA-PN	FS-WSA-DT	FS-WSA-NB	FS-Cfs-PN	FS-Cfs-DT	FS-Cfs-NB
Average error (per classifier)	0.11096	0.1515	0.23894	0.08776	0.16954	0.38748	0.08798	0.09966	0.2323	0.46562	0.58604	0.62314
Average error (per algo. group)		0.167133333			0.214926667			0.13998			0.558266667	
St. dev. error (per algo. group)		0.065406582			0.154928957			0.080164471			0.082350811	
Time consumption (per classifier)	14881.90607	760.10988	500.44817	5891.07444	645.36168	305.9484432	6671.47066	16908.49644	47372.0138	≈0	≈0	≈0
Av. time con. (per algo. group)		5380.821372			2280.794854			23650.6603			≈0	
St. dev. time con. (per algo. group)		8229.204933			3131.196153			21171.35171			≈0	
% Dimension (per classifier)	75.23341747	75.33096362	25.70943097	64.57907591	75.53820549	30.29258042	55.47927538	57.73169913	45.58275197	16.94849599	16.9489599	16.94849599
Av. % Dimension (per algo. group)		58.75793735			56.80328727			52.93124216			16.94849599	
St. dev. % dim. (per algo. group)		28.62088765			23.60378812			6.462861923			≈0	

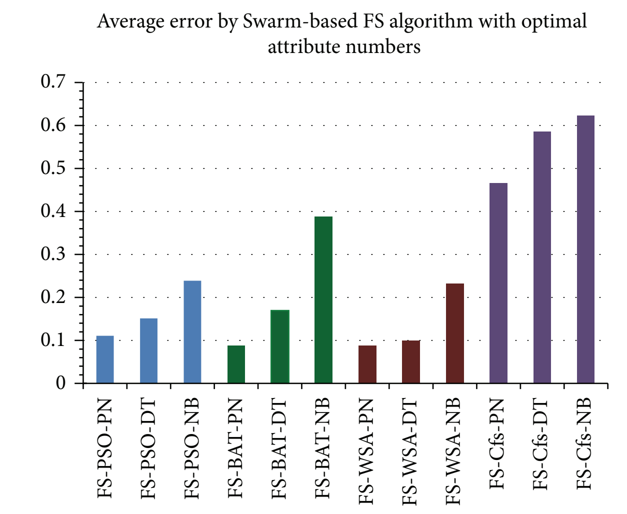

Figure 13: Comparison of different Swarm-based FS algorithms with optimal features in average error.

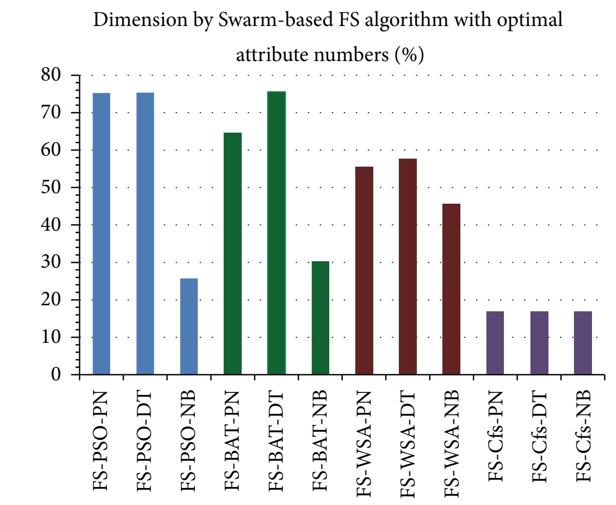

Figure 15: Comparison of different Swarm-based FS algorithms with optimal features in % dimension.

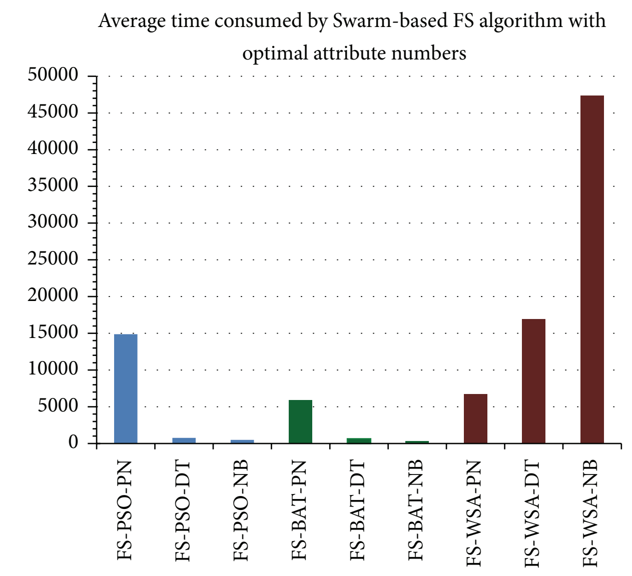

Figure 14: Comparison of different Swarm-based FS algorithms with optimal features in consumed time.

With respect to dimension exploitation, as shown in Figure 15, Cfs rates out as it only reaches to about 18% of the maximum dimension. The strongly swarming optimizers like PSO and BAT are able to climb up to 75% of the maximum dimension for acquiring optimal solutions. PSO with Pattern Network or Decision Trees is equally well, but PSO works poorly with Naïve Bayes in dimension exploitation. Similarly, BAT does not work well with Naïve Bayes too in outreaching to high data dimensions.

From the above results, it is verified that Swarm Search is in superior to the standard feature selection method, in terms of error rate and how high the data dimensions that the search agents can exploit. The advantage of Swarm Search however is at the cost of time consumption in the performing the heuristic search in high dimensional space.

Lastly,the performance results in terms of error, time, and % dimensions are further aggregated into groups characterized by the metaheuristic methods, such as FS-PSO, FS-BAT, FS-WSA, and FS-Cfs (which is just for reference). The ultimate aggregated results are tabulated in Table 9. The corresponding performance with respect to error rate versus time consumption is shown in Figure 16. Such that the methods that are closer to the zero coordinate the better (low error and short time).

Here we observe a very interesting phenomenon; FS-CFS correlation-based group is the quickest, but it has the highest error rate. FS-WSA group which represents the semiswarm heuristics conversely yields the lowest error rate but costs the longest time. The two remaining groups of strong swarm-type sit near the optimal position along the cost-and-benefit curve. When considering all the results aggregated over different data sets and different popular classifiers, FS-BAT group has a relatively higher error rate than FS-PSO, but they run at relatively shorter convergence time than FS-PSO.

5. Conclusion

Dimensionality reduction in input dataset is crucial in machine learning, especially when dealing with a high-dimensional feature space. The original feature space is mapped onto a new space of reduced dimensions. Identification of relevant features is extremely important for classification tasks (improving accuracy and reducing computational costs). Given the high dimensionality in the data, selecting a right subset of useful features from all the original features is a challenging problem. There is no golden rule of thumbs how this should be done albeit a lot of research efforts have been going on, advocating different feature selection methods.

However, to the best of the authors' knowledge, no universal method is being claimed, though recently a high surge in hybrid modes of integrating metaheuristics into

Table 9: Comparison of overall performances of different Swarm-based feature selection algorithms.

Performance	FS-PSO	FS-BAT	FS-WSA	FS-Cfs
Average error	0.16713333	0.21492667	0.13998	0.558267
Average time	5380.82137	2280.79485	23650.6603	≈0
Average % dim	58.7579374	56.8032873	52.9312422	16.9485

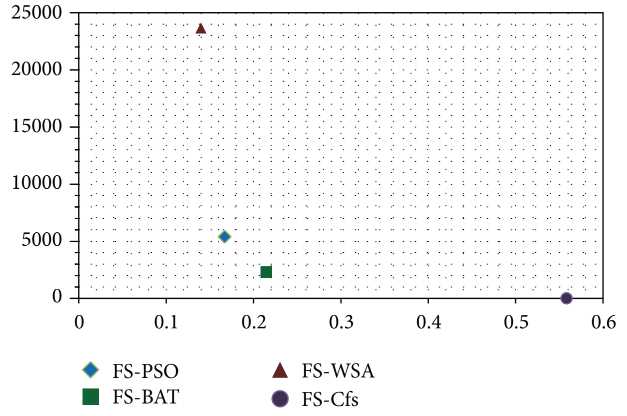

Figure 16: Overall comparison of Swarm-search models in error rates versus time consumption.

wrapper-based feature selection method is seen in the literature. The designs of the proposed feature selection models in most of those works focus on either a single classification algorithms or a single metaheuristic method. Also in some works, the feature selection is limited to a specific feature set length that has to be inputted manually by the user. Moreover, the experiments are subject to datasets of merely dozens of features in some of them.

In view of these, we proposed in this paper a general feature selection model, called Swarm Search which can potentially work with any classification algorithms as the fitness function and any swarm-based metaheuristics for searching in the search space stochastically for an optimal feature subset. The optimal feature subset needs not to be defined of its length in advance; we leave it to the search agents to scrutinize high dimensional data space to find it.

Under the framework of Swarm Search, nine metaheuristic feature selection methods are devised by combing three popular classifiers and three contemporary bioinspired optimizers. The proposed Swarm Search methods were put under test across five different types of datasets with a high dimensionality up to 875. Their performances were evaluated via the computer simulation in Matlab; the results suggest that Swarm Search models have superiority over correlation-based feature selections. In general, Swarm Search methods can yield relatively lower error rate and be able to outreach to high dimensional space for searching for an optimal feature subset. The performance of feature selection by Swarm Search is superb in nominal, pure numeric, and mixed-typed datasets. It pairs well with different classifiers with an exception of Naïve Bayes where the attributes in the data are not independent. The performance results are compared and contrasted in this paper, which can serve as a future reference guideline for scientists who want to choose these metaheuristic-based feature selection methods.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper. The authors of this paper do not have a direct financial relationship with the commercial identities mentioned in this paper that might lead to a conflict of interests.

Acknowledgments

The authors are thankful for the financial support from the research grant of Grant no. MYRG152(Y3-L2)-FST11-ZY, offered by the University of Macau, RDAO. Special thanks go to Mr. Rui Tang, who was a research assistant and MSc software engineering graduate of University of Macau, for programming the Matlab codes and conducting the data mining experiments.

References

[1] S. Aeberhard, D. Coomans, and O. de Vel, "Comparison of classifiers in high dimensional settings," Tech. Rep. 92-02, Department of Computer Science and Department of Mathematics and Statistics, James Cook University, North Queensland, Australia, 1992.

[2] E.-G. Talbi, L. Jourdan, J. García-Nieto, and E. Alba, "Comparison of population based metaheuristics for feature selection: application to microarray data classification," in *Proceedings of the 6th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA '08)*, pp. 45–52, Doha, Qatar, April 2008.

[3] S. M. Vieira, L. F. Mendonca, G. J. Farinha, and J. M. C. Sousa, "Metaheuristics for feature selection: application to sepsis outcome prediction," in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 1–8, Brisbane, Australia, June 2012.

[4] J. Wang, A.-R. Hedar, S. Wang, and J. Ma, "Rough set and scatter search metaheuristic based feature selection for credit scoring," *Expert Systems with Applications*, vol. 39, no. 6, pp. 6123–6128, 2012.

[5] N. Abd-Alsabour and A. Moneim, "Diversification with an ant colony system for the feature selection problem," in *Proceedings of the 2nd International Conference on Management and Artificial Intelligence (IPEDR' 12)*, vol. 35, pp. 35–39, IACSIT Press, 2012.

[6] S. Casado, J. Pacheco, and L. Núñez, "A new variable selection method for classification," *XV Jornadas de ASEPUMA y III Encuentro Internacional*, pp. 1–11, 2007.

[7] J. B. Jona and N. Nagaveni, "Ant-cuckoo colony optimization for feature selection in digital mammogram," *Pakistan Journal of Biological Sciences*, pp. 1–6, 2013.

[8] A. Unler, A. Murat, and R. B. Chinnam, "Mr2PSO: a maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification," *Information Sciences*, vol. 181, no. 20, pp. 4625–4641, 2011.

[9] D. Korycinski, M. M. Crawford, and J. W. Barnes, "Adaptive feature selection for hyperspectral data analysis," in *9th Image and Signal Processing for Remote Sensing*, vol. 5238 of *Proceedings of SPIE*, pp. 213–225, Barcelona, Spain, 2004.

[10] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 525–534, 2009.

[11] A. Unler and A. Murat, "A discrete particle swarm optimization method for feature selection in binary classification problems," *European Journal of Operational Research*, vol. 206, no. 3, pp. 528–539, 2010.

[12] M. García-Torres, C. F. García López, B. Melián-Batista, A. J. Moreno-Pérez, and J. M. Moreno-Vega, "Solving feature subset selection problem by a hybrid metaheuristic," *Hybrid Metaheuristics*, pp. 59–68, 2004.

[13] S. El Ferchichi and K. Laabidi, "Genetic algorithm and tabu search for feature selection," *Studies in Informatics and Control*, vol. 18, no. 2, pp. 181–187, 2009.

[14] A. Al-Ani, "Feature subset selection using ant colony optimization," *International Journal of Information and Mathematical Sciences*, vol. 2, article 1, pp. 53–58, 2006.

[15] L. C. Molina, L. Belanche, and À. Nebot, "Feature selection algorithms: a survey and experimental evaluation," in *Proceedings of the IEEE International Conference on Data Mining, (ICDM '02)*, pp. 306–313, Maebashi, Japan, December 2002.

[16] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall, Englewood Cliffs, NJ, USA, 3rd edition, 1992.

[17] M. Deriche and A. Al-Ani, "Feature selection using a mutual information based measure," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 4, pp. 82–85, Quebec, Canada, August 2002.

[18] G. Kumar and K. Kumar, "A novel evaluation function for feature selection based upon information theory," in *Proceedings of the 24th Canadian Conference on Electrical and Computer Engineering (CCECE '11)*, pp. 395–399, Niagara Falls, NY, USA, May 2011.

[19] M. A. Hall and L. Smith, "A. Practical feature subset selection for machine learning," in *Proceedings of the Australian Computer Science Conference*, pp. 181–191, Springer, New York, NY, USA, 1998.

[20] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[21] R. Ruiz, J. C. Riquelme, and J. S. Aguilar-Ruiz, "Heuristic search over a ranking for feature selection," in *Proceedings of the 8th International Workshop on Artificial Neural Networks, (IWANN '05)*, pp. 742–749, Barcelona, Spain, June 2005.

[22] T. N. Lal, O. Chapelle, J. Western, and A. Elisseeff, "Embedded methods," *Studies in Fuzziness and Soft Computing*, vol. 207, pp. 137–165, 2006.

[23] X. S. Yang, "Swarm-based metaheuristic algorithms and no-free-lunch theorems," in *Theory and New Applications of Swarm Intelligence*, R. Parpinelli and S. Heitor Lopes, Eds., InTech, 2012.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks. Part 1*, vol. 4, pp. 1942–1948, December 1995.

[25] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," *Nature Inspired Cooperative Strategies for Optimization (NICSO '10)*, vol. 284, pp. 65–74, 2010.

[26] R. Tang, S. Fong, X. S. Yang, and S. Deb, "Wolf search algorithm with ephemeral memory," in *Proceedings of the IEEE 7th International Conference on Digital Information Management (ICDIM '12)*, pp. 165–172, August 2012.

[27] S. Fong, K. Lan, and R. Wong, "Classifying human voices by using hybrid SFX time-series pre-processing and ensemble feature selection," *Biomed Research International*, vol. 2013, Article ID 720834, 27 pages, 2013.