Bio-inspired algorithms for diagnosis of breast cancer

Moolchand Sharma*, Shubbham Gupta, Prerna Sharma and Deepak Gupta

Department of Computer Science and Engineering,
Maharaja Agrasen Institute of Technology,
Delhi, 1234, India
Email: moolchand@mait.ac.in
Email: shubbham.gupta28@gmail.com
Email: prernasharma@mait.ac.in
Email: deepakgupta@mait.ac.in
*Corresponding author

Abstract: Most commonly found cancer among women is breast cancer. Roughly 12% of women grow breast cancer during their lifetime. It is the second prominent fatal cancer among women. Breast cancer diagnosis is necessary during its initial phase for the proper treatment of the patients to lead constructive lives for an extensive period. Many different algorithms are introduced to improve the diagnosis of breast cancer, but many have less efficiency. In this work, we have compared different bio-inspired algorithms including artificial bee colony optimisation, particle swarm optimisation, ant colony optimisation and firefly algorithm. The performances on these algorithms have been measured for UCI Dataset of Wisconsin Diagnostic Breast Cancer, and the results have been calculated using different classifiers on the selected features. After the experiment, it is seen that BPSO has shown maximum accuracy of 96.45% and BFA has shown considerable results of 95.81% with six features which is minimum of all algorithms.

Keywords: breast cancer; bio-inspired algorithms; artificial bee colony optimisation; particle swarm optimisation; ant colony optimisation; firefly algorithm; feature selection; decision tree; linear support vector machines; K-nearest neighbour; random forest classifiers.

Reference to this paper should be made as follows: Sharma, M., Gupta, S., Sharma, P. and Gupta, D. (2019) 'Bio-inspired algorithms for diagnosis of breast cancer', *Int. J. Innovative Computing and Applications*, Vol. 10, Nos. 3/4, pp.164–174.

Biographical notes: Moolchand Sharma is an Assistant Professor in Maharaja Agrasen Institute of Technology, Delhi. He is pursuing his PhD from Deenbandhu Chhotu Ram University of Science and Technology, Haryana. He completed his MTech in 2012 and BTech in 2010. His research area includes search engine optimisation, intelligent data analysis, nature-inspired computing, big data and machine learning. He has published scientific research publications in reputed international journals and conferences including SCI indexed journals (Elsevier). He is also the co-convener of 'ICICC' Springer conference series. He has also authored book chapters with International level publisher (Wiley, Elsevier).

Shubbham Gupta is an outstanding student of Computer Science Department at Maharaja Agrasen Institute of technology, Delhi. He is a passionate and dedicated data science and machine learning enthusiast and developer who loves to make impact to the society. He will be pursuing his MSc in Data and Computational Science in University College Dublin, Dublin. He is also looking forward for more challenging tasks.

Prerna Sharma has a rich academic background and teaching experience of eight years. She is serving as an Assistant Professor in Maharaja Agrasen Institute of Technology, Delhi. She has completed her MTech in 2011 from USIT, GGSIPU and BTech in 2009 from GPMCE, GGSIPU. She has published a number of research papers in the journals and conferences of national and international repute which includes Scopus and SCI indexed journals of Springer and Elsevier. Her area of interest includes artificial intelligence, machine learning, nature-inspired computing and soft computing. She has also authored book chapters with International level publisher (Wiley, Elsevier).

Deepak Gupta is an eminent academician; plays versatile roles and responsibilities juggling between lectures, research, publications, consultancy, community service, PhD and post-doctorate supervision etc. He has contributed massive literature in the fields of human-computer interaction, intelligent data analysis, nature-inspired computing, machine learning and soft computing. He has published 63 scientific research publications in reputed international journals and conferences including 26 SCI indexed journals of IEEE, Elsevier, Springer, Wiley and many more. He has served as editor-in-chief, guest editor, associate editor in sci and various other reputed journals (Elsevier, Springer, Wiley and MDPI). He has actively been an organising end of various reputed International conferences. He has completed his Post-Doc from Inatel, Brazil, and PhD from Dr. APJ Abdul Kalam Technical University. He has authored/edited 33 books with national/international level publisher (Elsevier, Springer, Wiley, Katson).

1 Introduction

Breast cancer is an illness that originates by a mutation in the DNA genes of breast tissue and these abnormal cells group together to form a tumour. Some tumours may be inherited, develop randomly over time, while others may be the result of environmental exposures or lifestyle factors. Symptoms of the disease may include a mass formation in the breast, abnormality of the skin texture, a transformation in breast shape, or scaly or red patch of skin. There are several ways of detecting breast cancer at its early stages so that the disease doesn't change the quality of a patient's life and increase the chances of survival significantly and improve the prognosis. Accurate classification of breast cancer into a benign and malignant tumour is necessary to prevent the patient from experience avoidable treatments. Machine learning is extensively recognised as a procedure of choice in critical feature detection in BC pattern classification and forecast modelling. But instead of training ML models on full dataset, we train the model on most prominent features, which are selected by using various bio-inspired algorithms. In this study, we have correlated the efficiency of four bio-inspired optimisations which are artificial bee colony optimisation, particle swarm optimisation, ant colony optimisation, and firefly algorithm and calculated the accuracy of each algorithm on Wisconsin Diagnostic Breast Cancer (WDBC) Database from UCI machine learning repository.

Evolutionary computation algorithms have intrigued a lot of great minds for determining optimisation tasks in the present time. These algorithms are the general bio-inspired optimisation algorithm based on living organism's population which mimics growth processes such as mutation reproduction, selection to resolve optimisation tasks. Many algorithms have been designed by studying from nature in modern researchers, like neural networks based on human neural system, genetic algorithm based on natural selection, particle swarm optimisation uses the behaviour of bird flocks or swarms, firefly algorithm based on natural behaviour of fireflies, ant colony optimisation based on in-direct communication of ants and artificial bee colony optimisation based on collective foraging of honeybees.

In modern years, data has increased in both volumes of instances and features, which leads to noisy and inaccurate data. The noisier databases can lead a program to raise complexity, time-consuming data training, rise computational costs, decreases in the accuracy anticipated by models. Therefore feature selection is a critical process for machine learning before preparing the model. Attribute selection, or feature selection, is a technique to select a subset from the provided full set of features and fewer decline in the performances of the system; such that the subset of features anticipates the target with accuracy pertaining to the performance of the original set of features, and with the lessening of computational cost. Feature selection can be mainly classified as wrapper-based and filter-based algorithm. The filter-based algorithm uses a statistical approach for scoring each feature and wrapper-based algorithm, computational costlier than the filter method, uses a machine learning algorithm for evaluation. Bio-inspired algorithms can be modified to perform feature selection.

The main contributions of this work are:

- Performance of four popular bio-inspired algorithms, i.e., BABCO, BACO, BFA, BPSO, is compared here.
- For classification goal, we used four different algorithms:
  1. decision tree
  2. random forest
  3. K-nearest neighbours (K-NN)
  4. linear support vector machines classifiers.
- All four algorithms are compared on three bases which are accuracy, computational time and number of features selected.
- The outcomes of the research show that the best solution is found by using the random forest algorithm used along with BABCO.
- Overall performance seen in BPSO is best among all four algorithms.

The paper is methodised as follows: In the literature review, relevant concepts and work related to the bio-inspired algorithm and machine learning classifiers are described. The proposed procedure for each used algorithm is then described in detail. Experimental results achieved through the application of the proposed algorithm to each four of classifiers on breast cancer dataset are described and discussed. Finally, the rest section includes the paper with final remarks.

2 Literature review

According to Technopedia, An evolutionary algorithm works through the selection process where the least fit members of the population set are eliminated from the algorithm, whereas the fit members are allowed to survive and continue until a better solution is determined. In other words, Evolutionary algorithms are a computational algorithm that is inspired and mimic biological processes in order to solve complex problems. There are various bio-inspired algorithms currently in working but the common idea behind all of them is the equivalent: provide a population of individuals, the natural process causes common selection (survival of the fittest), and this causes a spike in the fitness of the population (Nayyar et al., 2018). Apart from having a common set of features in evolutionary computational techniques, they each have distinctive features that can be seen by reading further into the level of abstraction of each technique (Eiben and Smith, 2007).

As per the unified approach suggested by De Jong (1975), the evolutionary algorithm must have these components, i.e., representation, fitness function, parent's selection, population, crossover operator, mutation operator, survivor selection and termination condition. Representation includes genotype-phenotype mapping of real-world objects to binary representation. The population is a set of individuals which may be a possible solution of given problem and fitness function also known as evaluation function optimises the population to be an optimal solution. Parent's selection is the selection of a subset of individuals responsible for the production of offspring of next generation for the next iteration. Crossover and mutation operator are the operators which frame new individuals for the next generation from the previous generation. Survivor selection is similar to parent's selection but works after the reproduction stage. It selects the best individuals from the population with higher fitness value to optimise the solution. This whole process repeats until the termination condition is met. If the program cannot find an optimal solution then termination condition comes into action and stops the program.

3 Proposed work/methodology

In this paper, we have used four bio-inspired algorithms and four machine learning classifiers all following a general step by step process described in the flowchart as shown in Figure 1.

3.1 Artificial bee colony optimisation

Artificial bee colony optimisation (Karaboga, 2005; Gao and Liu, 2012) is a bio-inspired algorithm searches the best solution among a huge number of substitutes. The main proposal is derived from the honey bees' behaviour of collective exploration. The honey bees' behaviour based on nest site allocation, mating, task allocation, communication, navigation behaviour and pheromone laying is mimicked in this algorithm.

Figure 1 General algorithm followed in each algorithm

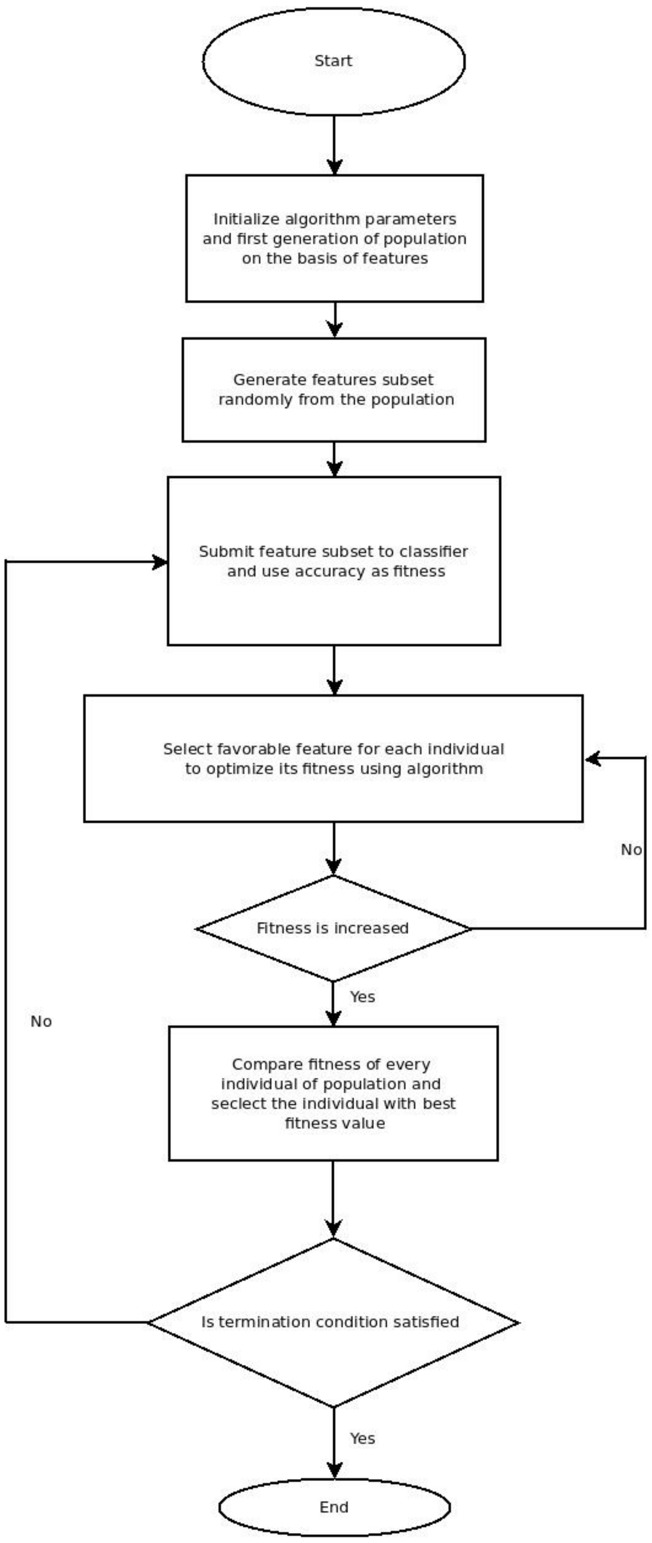
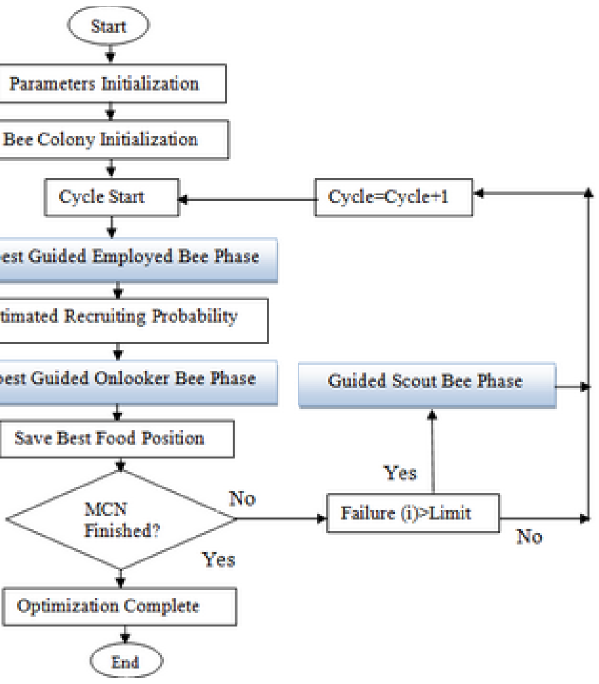

Start
Initialize algorithm parameters and first generation of population on the basis of features
Generate features subset randomly from the population
Submit feature subset to classifier and use accuracy as fitness
Select favorable feature for each individual to optimize its fitness using algorithm
Fitness is increased
No
Yes
Compare fitness of every individual of population and seclect the individual with best fitness value
Is termination condition satisfied
No
Yes
End

Figure 2 Artificial bee colony optimisation flowchart (see online version for colours)


Start
Parameters Initialization
Bee Colony Initialization
Cycle Start
Cycle=Cycle+1
Gbest Guided Employed Bee Phase
Estimated Recruiting Probability
Gbest Guided Onlooker Bee Phase
Guided Scout Bee Phase
Save Best Food Position
Yes
MCN Finished?
No
Failure (i)>Limit
No
Yes
Optimization Complete
End

At the first step of this algorithm as shown in Figure 2, bees initialise the probable food sources that are recognised from the population. Using random catalyst employed bees hunt for different food origin initially. When a food origin (a possible result) is found, the fitness of the solution is calculated. In the later stage, if a different food origin is determined with a greater fitness value, the different source is accepted otherwise it is denied. This updated fitness value is now shared with the all other onlooker bees. They select food origin for them on the basis of occurrence of food source (occurrence is calculated as the proportion of the evaluation function of an origin to the sum of evaluation functions of all the food origins). The results are rejected if bees are inadequate to increase the fitness of the food source.

A general pseudo-algorithm for ABC optimisation (Karaboga et al., 2012) algorithm is shown in Algorithm 1.

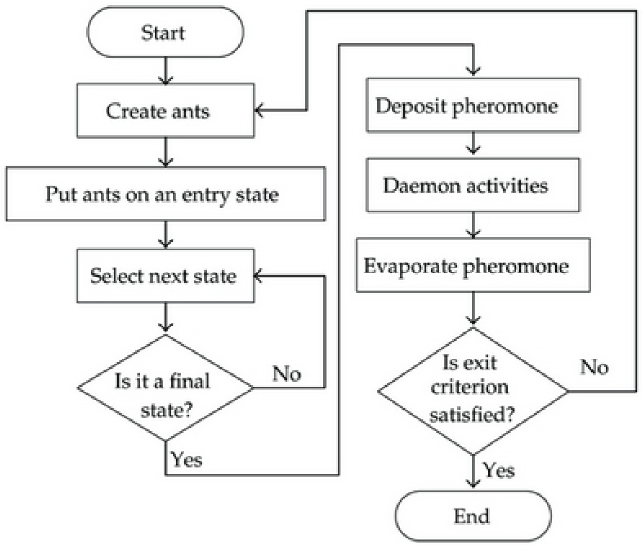
Algorithm 1

```
Start
    Initialisation stage
    While iter < termination condition, i.e., maximum CPU time
        Employed bee stage
        Onlooker bee stage
        Scout bee stage
        Select best result achieved so far
    End While
End
```

3.2 Ant colony optimisation

Ant colony optimisation (Dorigo and Birattari, 2011; Dorigo et al., 2006) is an optimisation algorithm, for solving combinational optimisation questions. The algorithm is derived from the indirect communication of ants (factor) exploring for knowledge, arbitrated by artificial pathways (known as pheromones trials in ants). Ants derive results based on probabilistic exploration using pheromone trials which provide scattered numerical information. This algorithm performs the optimisation in modest search time. The solution is frequently attempted in a progression of iterative steps in this algorithm (Ghasab et al., 2015; Dorigo and Blum, 2005) as shown in Figure 3. The parametric probability distribution is used to find possible results are identified from a set of results using a pa. To perform this, a set of decision variable is defined by a set of newly initialised ants. These decision variables are then chosen by ants for generating possible solutions. While ants analyse these possible solutions, pheromone updation is executed locally on the result based on its fitness. Pheromone trails are the altered by these possible results according to the local pheromone updation in such way that set of ants prefer the higher pheromone result in the successive experiment for possible solutions. The random possible results generated in the first stage, introduce the course for optimal results.

Figure 3 Ant colony optimisation flowchart



A general pseudo-algorithm for ant colony optimisation algorithm (Kumar and Reddy, 2006) is shown in Algorithm 2.

Algorithm 2

```
Start
    Generate initial population
    While termination criteria not satisfied do
        Place each ant at starting point
        Repeat
            For each ant perform
                Choose the next point by applying state
                transition rule
                Update pheromone at each step
            End For
        Until each ant has found a solution
        Update the best solution
        Update pheromone globally
    End while
End
```

3.3 Firefly algorithm

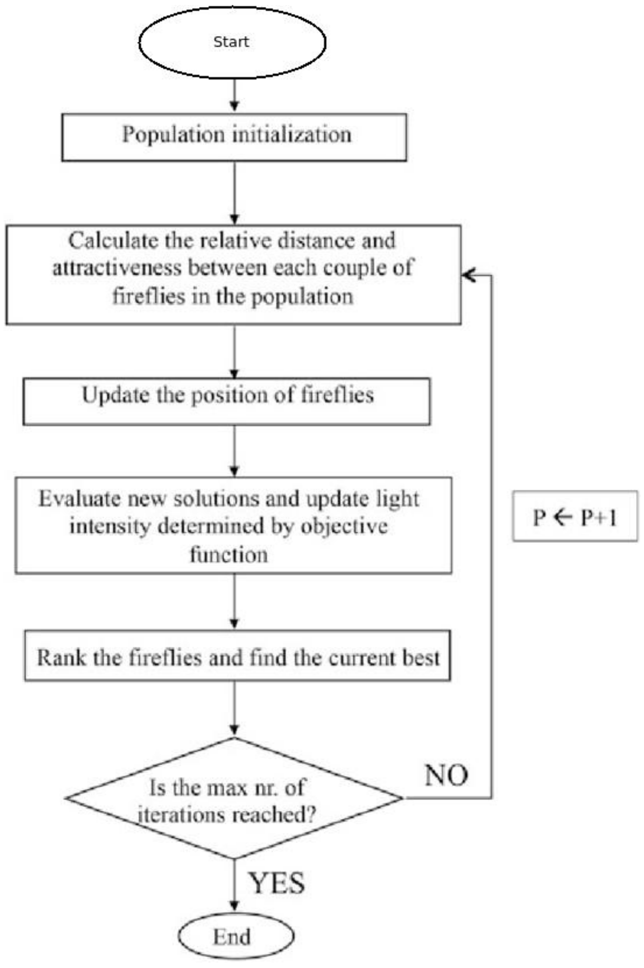
Firefly algorithms (Gandomi et al., 2011; Yang, 2009; Yang et al., 2012) were proposed to solve NP-hard dilemma with non-convex target functions which have inequality and equality based restraints. Firefly algorithms work on a multi-modal objective with higher accuracy than any different swarm optimisation till introduced (Yang, 2009). Firefly algorithm operates on a population derived exploration and hence possible results profit from building segments from very several results. The technique provides better learning of parameter training to maintain equilibrium in search versus using. This algorithm is derived from the fireflies corresponding behaviour which includes bio-luminescence signalling to each firefly and for elude danger. These fireflies show attributes of swarm intelligence by decentralised decision making and self-organising. Fireflies use bioluminescence signalling for exploring for food, and they use signals for breeding purposes. The fitness of male firefly is checked by the illumination of the tail flash. Still, for the general process, they are treated as unisex and the appeal of firefly is a direct function of the magnitude of the light, which indicates the fitness of a likely 'possible results'. At the first stage, a primary population of fireflies is generated. After this stage, fitness is changed by one or more condition and is again calculated for every firefly of the population. Then fireflies are sorted by fitness and top k fireflies in the current population are selected for the next evaluation stage as shown in Figure 4. Firefly algorithm is also useful as it can be used alongside other algorithms in hybrid approaches to improve results (Rahmani and MirHassani, 2014).

A general pseudo-algorithm for firefly algorithm (Ahammed et al., 2018) is shown in Algorithm 3.

Algorithm 3

```
Start
    Initialise $\alpha$, $\beta$, $\gamma$ and max_iter
        Specify objective function f(x)
        Establish Intensity(I) at cost(x) for each individual
        specified by f(x)
        While t < max_iteration
            For x = 1 to n
                For y = 1 to n
                    if ($I_y > I_x$)
                        In k^th dimension move firefly y to
                        firefly x
                    end if
                    Calculate new results and update light
                    intensity
                end for y
            end for x
        Sort fireflies on fitness and find best current best
        end while
    Post transform results and visualisation
End
```

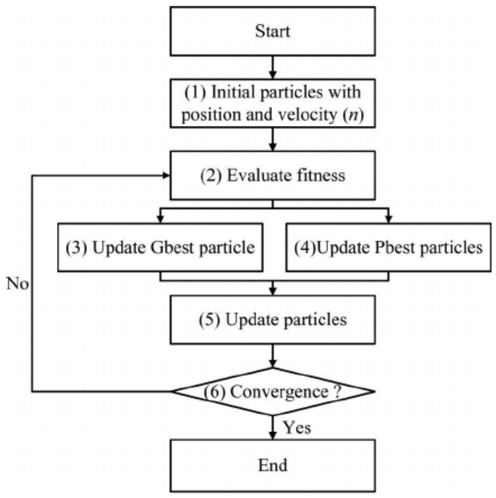
Figure 4 Firefly algorithm flowchart


Start
Population initialization
Calculate the relative distance and attractiveness between each couple of fireflies in the population
Update the position of fireflies
Evaluate new solutions and update light intensity determined by objective function
Rank the fireflies and find the current best
Is the max nr. of iterations reached?
NO
YES
P ← P+1
End

3.4 Particle swarm optimisation

Particle swarm optimisation (Shi and Eberhart, 1998, 1999) is derived from the shared group behaviour of creatures such as birds flocking, insect swarming or fish schooling, where the collective objective meets based on feedback from all other units of the group. A swarm is a huge number of homogeneous, scattered agents that communicate mutually among themselves and the environment, where an ideal result is sought to be found. The main purpose of this optimisation is optimising functions with discontinuous, non-differential with several nonlinearly related parameters (Floreano and Mattiussi, 2008). These algorithms operate in few progressive stages defined on the behaviour of the creature it mimics. Each particle in the swarm attempts to find a possible result at any position in the process. Each particle broadcasts a signal comparable to the fitness of probable result to different units in the pack. The strength of the signal broadcasted is then recognised by every swarm particle, and hence the fitness of the possible result based on a fitness function is known to every particle. When a particle attempts to find a better possible result from currently derived results, based on various learning methods (Zhan et al., 2011), a new course is found with an inertial weight to constantly direct the particles to an ideal solution if achievable. Optimal possible results are computed using real numbers, this optimisation supports a simpler complexity of application (Kar, 2016). Particularly in such swarm optimisation main role is of Levy walks and Levy flights, where the behaviour of likely swarm particles that shift spontaneously between successive locations is picked up and used for guessing the direction of convergence of possible results as shown in Figure 5.

Figure 5 Particle swarm optimisation flowchart


Start
(1) Initial particles with position and velocity (n)
(2) Evaluate fitness
(3) Update Gbest particle
(4)Update Pbest particles
(5) Update particles
(6) Convergence ?
No
Yes
End

A general pseudo-algorithm for particle swarm optimisation algorithm (Sahib and Ahmed, 2015) is shown in Algorithm 4.

Algorithm 4

```
Start
    Initialise particles population
    while termination criterion is not satisfied
        for every particle p at $x_p$ do
            evaluate fitness value $f(x_p)$
            if $f(x_p)$ is better than $pbest_p$ then
                $pbest_p \leftarrow x_p$
            end if
        end for
        Define $gbest_p$ as best position found by any of p's
        neighbours so far
        for every particle p do
            $v_p \leftarrow$ compute_velocity($x_p$, $pbest_p$, $gbest_p$)
            $x_p \leftarrow$ update_position($c_p$, $v_p$)
        end for
    end while
end
```

4 Implementation and results

In this section, the planned algorithm has been executed. Every one of four bio-inspired algorithms is executed with each four machine learning classifiers. Each combination of algorithm and classifier is run for 20 times and mean of all 20 is considered as the final result.

4.1 Experimental setup

The entire experiment was performed on a mobile computing setup, a laptop with 6th generation i5 core clocked at 2.4 GHz with 3 MB cache memory coupled with 16 gigabytes of RAM and 1 TB hard disk, running on Ubuntu OS (ver. 18.04, 64 bit). The software setup included Jupyter Notebook with Python 3.2 with libraries such as numpy, scikit-learn, and Pandas.

4.2 Input parameters

4.2.1 Machine learning classifiers

Classification is the technique of anticipating the class or label of provided data points. It can be executed on any data type, i.e., unstructured or structured data. Classification is mainly used for the purpose of identifying the category/class to which a new data will belong according to the training data. Classification is a kind of supervised learning in which the targets are given along with the input data. In this paper, we have used these four following classifiers

- *Decision tree*

  A decision tree generates a set of courses that are used to distribute the data together with its classes. Decision Tree is easier to visualise and understand, can work on both categorical and numerical data and take comparatively low data preparation. Complex trees which are not difficult to understand and in generalised form are the output of decision tree classifier. The main upside to decision trees is that they are unstable in nature that, i.e., even a small variation will lead to a fully separate tree than it is before the change, hence high computational cost with every change in training data.

- *K-nearest neighbours*

  Neighbours-based classification is a kind of lazy learning as it learns without constructing a classification model but directly keep dataset of the training data. Classification is computed by comparing the k nearest neighbours of each point and taking those results into consideration. This classification is working on a huge dataset, easy to achieve, and ineffective to noisy training data. It takes high computational time and cost because it compares the distance of every test point with full training dataset.

- *Random forest*

  Random forest classifier is a classification method which forms a set of a number of decision trees on the separate smaller subset of training data and formulates result by taking an average of the accuracy of each tree in the model. The subsample size and the original input sample size are constantly equal but the samples are drawn with replacement. Reduction in overfitting seen in this and random forest classifier shows better results than decision trees in many tests. Random Forest is a time consuming slow prediction and it is complicated to achieve due to its complex algorithm.

- *Support vector machine*

  Support vector machine is a depiction of training dataset as different points in hyperplane distributed into classes. Test data points are then charted into that hyperplane made earlier and the class is anticipated according to the closeness of the test data with the clusters formed in the hyperplane. It is beneficial if working on high dimensional data and it computes result from using a subset of training dataset in the decision function, thus it is memory efficient. The algorithm provides probability estimates by calculating them using a high-cost five-fold cross-validation. The input parameters for each classifier are presented in Tables 1 to 5.

Table 1 Input parameters for classifiers

Classifier	*Parameter*
Decision Tree	Minimum depth = 30 Minimum sample split = 20 Maximum sample leaf = 1
K nearest neighbours	Neighbours = 3 Leaf size = 30
Random forest	Estimators = 25 Minimum sample split = 2
Linear support vector machine	Maximum iteration = 1,000

Table 2 Input parameters for BPSO

Parameters	*Value*
Number of population	20
Maximum iteration	200
Move rate	0.5
Limit search range	4

Table 3 Input parameters for BFA

Parameters	*Value*
Number of population	20
Maximum iteration	200
Alpha	0.25
Beta	0.5
Gamma	0.2

Table 4 Input parameters for BABCO

Parameters	*Value*
Number of population	10
Maximum iteration	25
Employee percent	0.5
Maximum limit of search	5

Table 5 Input parameters for BACO

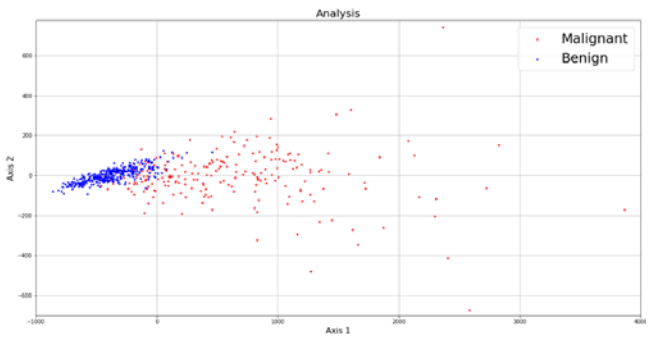
Parameters	*Value*
Number of population	30
Maximum Iteration	25
Initial pheromone	0.1
Evaporation of pheromone	0.75
Best subset influencing next iteration	10

4.3 Dataset

In this paper, we have used Wisconsin Diagnostic Breast Cancer Dataset made on November 1995.

Features are calculated from a digital image of a breast mass fine needle aspirate (FNA). They illustrate the cell nuclei characteristics of the breast mass present in the image. The dataset contains 569 instances which contain 32 attributes (two of them are ID and diagnosis). Each instance is diagnosed into either malignant or benign. Out of 569 instances, 357 are benign and rest 212 is malignant as shown in Figure 6.

Figure 6 Principal component analysis of dataset (see online version for colours)



The 32 attributes have 30 real-valued features which will be used in feature selection.

Ten real-valued features are calculated for every cell nucleus of breast mass:

a radius of nucleus (mean of distances from the centre to endpoints on the perimeter)

b texture of cell (standard deviation)

c perimeter of nucleus

d area of nucleus

e smoothness (variation in radius lengths)

f compactness (perimeter^2/area – 1.0) g concavity (severity of concave portions)

h concave points (number of concave portions)

i symmetry

j fractal dimension ('coastline approximation' – 1).

The mean, 'worst' or largest and standard error of these features are calculated for every cell, resulting in 30 features.

The mentioned dataset is divided in the ratio of 3:1 for training and testing purpose, i.e., training data = 75% and testing data = 25%.

4.4 Results

In the experiment, we have set termination condition according to iteration count as it gives us better results rather than accuracy required condition. While executing an algorithm with particular classifier gives different accuracy than any other combination of algorithm and classifier. When termination condition is set to accuracy required, then sometimes it may leads to infinite loop as algorithm is unable to reach that accuracy. So, it is difficult to initialise the accuracy as termination condition. It is an ad hoc method to determine what accuracy should be set to get desired results.

The benefit of iteration count is that it provides the maximum accuracy found till that run. To get more efficient results we can increase iteration. Hence the experiment is programmed with iteration count as termination condition.

4.4.1 BPSO

BPSO has shown the best results among the four compared algorithms as shown in Table 6. BPSO has the highest accuracy with random forest classifier with 96.45% and select minimum features (12 features) with Decision Tree Classifier as shown in Table7. It produces the fastest results when used with K nearest neighbours as shown in Figure 7.

Table 6 BPSO features selected with accuracy and time

Algorithm	*Classifier*	*Accuracy*	*Time (average in sec)*	*Feature selected*
BPSO	DTC	95.28%	134.5	12
	KNN	92.3%	27.5	16
	RFC	96.45%	965	13
	SVM	96.13%	527	13

Figure 7 BPSO accuracy graph (see online version for colours)

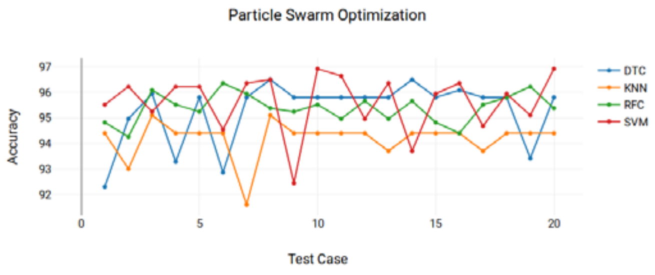


Table 7 Features repeated most using BPSO

Feature number	*Description*
2	texture_mean
5	smoothness_mean
8	concave points_mean
12	texture_se
15	smoothness_se
17	concavity_se
18	concave points_se
21	radius_worst
22	texture_worst
24	area_worst
27	concavity_worst
28	concave points_worst
29	symmetry_worst

4.4.2 BFA

BFA is the considerable choice among these algorithms. BFA has the highest accuracy with linear support vector machine with 95.64% along with minimum five features as shown in Tables 8 and 9. It produces the fastest results when used with K-nearest neighbours as shown in Figure 8.

Table 8 BFA features selected with accuracy and time

Algorithm	*Classifier*	*Accuracy*	*Time (average in sec)*	*Feature selected*
BFA	DTC	93.13%	217	6
	KNN	94.19%	23	7
	RFC	95.39%	612	8
	SVM	95.64%	706	5

Figure 8 BFA accuracy graph (see online version for colours)

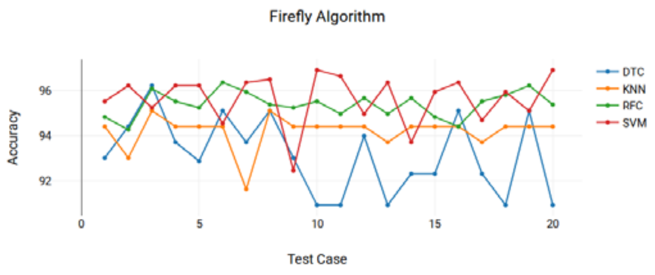


Table 9 Features repeated most using BFA

Feature number	*Description*
2	texture_mean
21	radius_worst
22	texture_worst
23	perimeter_worst
25	smoothness_worst
28	concave points_worst

4.4.3 BABCO

BABCO has shown nominal results in comparison. BABCO has the highest accuracy with linear support vector machine with 94% and shown minimum features of ten in both decision tree and K-nearest neighbours as shown in Tables 10 and 11. It also produces the fastest results when used with K nearest neighbours as shown in Figure 9.

Table 10 BABCO features selected with accuracy and time

Algorithm	*Classifier*	*Accuracy*	*Time (average in sec)*	*Feature selected*
BABCO	DTC	92.82%	16.25	10
	KNN	92.62%	13.5	10
	RFC	96.5%	245	14
	SVM	94%	31.75	11

Figure 9 BABCO accuracy graph (see online version for colours)

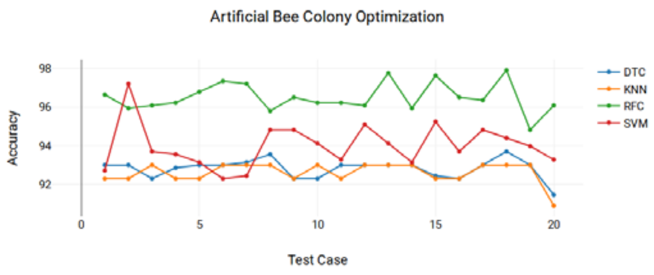


Table 11 Features repeated most using BABCO

Feature number	*Description*
2	texture_mean
5	smoothness_mean
7	concavity_mean
8	concave points_mean
9	symmetry_mean
10	fractal_dimension_mean
12	texture_se
17	concavity_se
22	texture_worst
25	smoothness_worst
29	symmetry_worst

4.4.4 BACO

BACO has shown the lowest results among these algorithms. BACO has the highest accuracy with random forest classifier with 95.18% and has selected minimum features with five features with linear support vector machine as shown in Tables 12 and 13. It produces fastest results when used with decision tree classifier as shown in Figure 10.

Table 12 BACO features selected with accuracy and time

Algorithm	*Classifier*	*Accuracy*	*Time (average in sec)*	*Feature selected*
BACO	DTC	90.91%	85.6	19
	KNN	91.43%	87.5	17
	RFC	95.18%	682.5	16
	SVM	83.19%	237.4	9

Figure 10 BACO accuracy graph (see online version for colours)

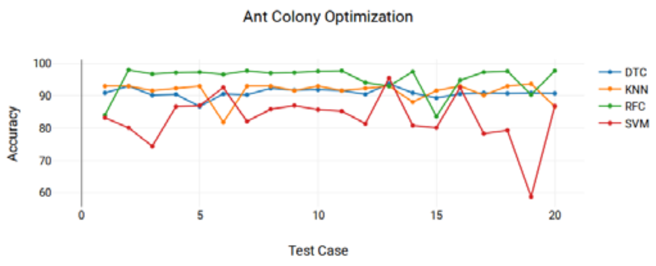


Table 13 Features repeated most using BACO

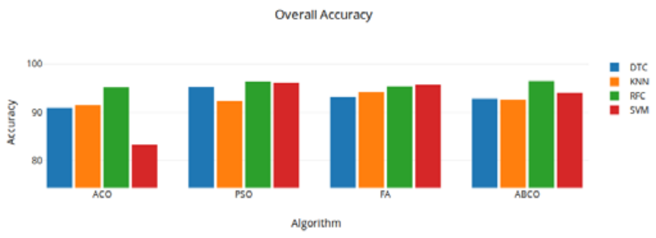
Feature number	*Description*
3	perimeter_mean
5	smoothness_mean
6	compactness_mean
7	concavity_mean
9	symmetry_mean
11	radius_se
13	perimeter_se
14	area_se
15	smoothness_se
16	compactness_se
18	concave points_se
19	symmetry_se
22	texture_worst
24	area_worst
28	concave points_worst

5 Comparison between BFA vs. BACO vs. BPSO vs. BABCO

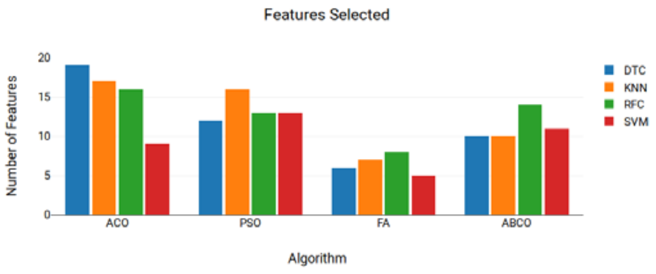
The results of each four algorithms have been compared in the following figures. The comparison is done on the basis of three criteria, i.e., overall accuracy of the algorithm over all four classifiers, the number of features selected and the average of time taken in each test case. After comparison, we have come to the conclusion that particle swarm optimisation shows the highest accuracy of 96.45%. In second criteria firefly algorithm select minimum of 6 features. After comparing algorithms on computational time, it can be seen that artificial bee colony optimisation evaluates results in minimum time, although it is also observed that average computational time is higher in all algorithms when random forest classifier is used.

The overall performance of the firefly algorithm as it selects minimum features and accuracy (95.81) is almost comparative to highest (PSO with 96.45%). It also provides result in nominal time as shown in Figures 11(a) to 11(c).

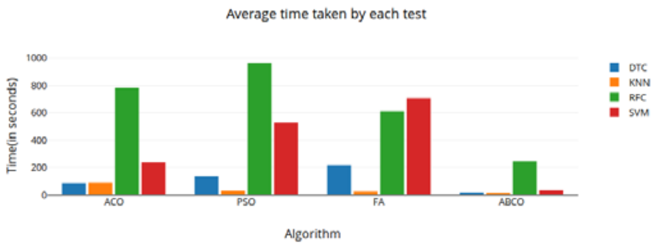
Figure 11 Comparison of four algorithms on the basis of (a) overall accuracy, (b) features selected and (c) average time per test case (see online version for colours)



(a)



(b)



(c)

6 Conclusions and future scope

In this paper, the comparison is performed between artificial bee colony optimisation, particle swarm optimisation, firefly algorithm, and ant colony optimisation along with different machine learning classifiers. Classifiers used are decision tree, linear support vector machines, K-nearest neighbour and random forest classifiers. After analysis, we can announce that among these four optimisation methods, particle swarm optimisation has shown best results with random forest classifiers (96.45%).

From studying the comparison it can be seen that some features are repeating in every result (texture mean, smoothness mean, texture_worst, concave points_worst). These features must be considered as these are crucial to the diagnosis of cancer.

In future, these algorithms can be used in prediction of more diseases as they provide results with higher accuracy in less computational time.

References

Ahammed, M.J., Swathi, A., Sanku, D., Vedula, C. and Ramesh, H. (2018) 'Performance of firefly algorithm for null positioning in linear arrays', *Proceedings of 2nd International Conference on Micro-Electronics, Elctromagnetics and Tele Communications*, pp.383–391, DOI: 10.1007/978-981-10-4280-5_40.

De Jong, K.A. (1975) 'An analysis of the behavior of a class of genetic adaptive systems (Doctoral dissertation, University of Michigan)', *Dissertation Abstracts International*, Vol. 36, No. 10, 5140B, University Microfilms No. 76-9381.

Dorigo, M. and Birattari, M. (2010) 'Ant colony optimization', in Sammut, C. and Webb, G.I. (Eds.): *Encyclopedia of Machine Learning*, Springer, Boston, MA, DOI: 10.1007/978-0-387-30164-8.

Dorigo, M. and Blum, C. (2005) 'Ant colony optimization theory: a survey', *Theoretical Computer Science*, Vol. 344, Nos. 2–3, pp.243–278, ISSN 0304-3975 [online] https://doi.org/10.1016/j.tcs.2005.05.020.

Dorigo, M., Birattari, M, and Stützle, T. (2006) 'Ant colony optimization', *IEEE Computational Intelligence Magazine*, November, Vol. 1, No. 4, pp.28–39, doi: 10.1109/MCI.2006.329691.

Eiben, A.E. and Smith, J.E. (2007) *Introduction to Evolutionary Computing*, Springer, Berlin, Germany.

Floreano, D. and Mattiussi, C. (2008) *Bio-inspired Artificial Intelligence: Theories, Methods, and Technologies*, MIT Press, Cambridge.

Gandomi, A.H., Yang, X.S. and Alavi, A.H. (2011) 'Mixed variable structural optimization using firefly algorithm', *Computers & Structures*, December, Vol. 89, pp.23–24 pp.2325–2336, DOI: 10.1016/j.compstruc.2011.08.002.

Gao, W.F. and Liu, S.Y. (2012) 'A modified artificial bee colony algorithm', *Computers & Operations Research*, March, Vol. 39, No. 3, pp.687–697, DOI: 10.1016/j.cor.2011.06.007.

Ghasab, M.A.J., Khamis, S., Mohammad, F. and Fariman, H.J. (2015) 'Feature decision-making ant colony optimization system for an automated recognition of plant species', *Expert Systems with Applications*, April, Vol. 42, No. 5, pp.2361–2370, DOI: 10.1016/j.eswa.2014.11.011.

Kar, A.K. (2016) 'Bio inspired computing – a review of algorithms and scope of applications', *Expert Systems with Applications*, doi: 10.1016/j.eswa.2016.04.018.

Karaboga, D. (2005) *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report-TR06, Vol. 200, Erciyes University, Computer Engineering Department.

Karaboga, D., Gorkemli, B., Ozturk, C. and Karaboga, N. (2012) 'A comprehensive survey: artificial bee colony (ABC) algorithm and applications', *Artif. Int. Rev.*, DOI:10.1007/s10462-012-9328-0.

Kumar, D.N. and Reddy, M.J. (2006) *Ant Colony Optimization for Multi-Purpose Reservoir Operation*, December, Vol. 20, No. 6, pp.879–898, DOI: 10.1007/s11269-005-9012-0.

Nayyar, A., Garg, S., Gupta, D. and Khanna, A. (2018) 'Evolutionary computation – theory and algorithms', *Advances in Swarm Intelligence for Optimizing Problems in Computer Science*, Chapter 1, CRC Press, Taylor & Francis Group.

Rahmani, A. and MirHassani, S.A. (2014) 'A hybrid firefly-genetic algorithm for the capacitated facility location problem', *Information Sciences*, November, Vol. 283, pp.70–78, DOI: https://doi.org/10.1016/j.ins.2014.06.002.

Sahib, M. and Ahmed, B. (2015) 'A new multi-objective performance criterion used in PID tuning optimization algorithms', *Journal of Advanced Research*, Vol. 115, DOI:10.1016/j.jare.2015.03.004.

Shi, Y. and Eberhart, R.C. (1998) 'Parameter selection in particle swarm optimization', *Evolutionary Programming VII*, pp.591–600, Springer, Berlin.

Shi, Y. and Eberhart, R.C. (1999) 'Empirical study of particle swarm optimization', *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*.

Yang, X.S. (2009) 'Firefly algorithms for multimodal optimization', *Stochastic Algorithms: Foundations and Applications*, pp.169–178, Springer, Berlin, Heidelberg.

Yang, X.S., Hosseini, S.S.S. and Gandomi, A.H. (2012) 'Firefly algorithm for solving nonconvex economic dispatch problems with valve loading effect', *Applied Soft Computing*, Vol. 12, No. 3, pp.1180–1186, ISSN 1568-4946, DOI: 10.1016/j.asoc. 2011.09.017.

Zhan, Z.H., Zhang, J., Li, Y. and Shi, Y.H. (2011) 'Orthogonal learning particle swarm optimization', *IEEE Transactions on Evolutionary Computation*, December, Vol. 15, No. 6, pp.832–847, doi: 10.1109/TEVC.2010.2052054.