

**Adding Vector Sampling to Reinforcement Learning Artificial Bee Colony (R-ABC) Algorithm for Feature Selection**

Submitted September 2016, in partial fulfilment of

the conditions of the award of the degree **MSc in Computer Science.**

**Daniel Chen Kwok Lee**0

**ID Number: 20129420**

School of Computer Science

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:



**Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Table of Contents**

Abstract ..........................................................................................................................................2

Acknowledgements ........................................................................................................................4

Introduction ....................................................................................................................................5

Literature Review ............................................................................................................................7

Proposed Modifications to Existing Algorithm ..............................................................................18

Approach to work and Methodology .............................................................................................24

Results.............................................................................................................................................29

Conclusion ......................................................................................................................................37

Possible Future Work ......................................................................................................................39

References ......................................................................................................................................43

**Abstract**

ABC (Artificial Bee Colony) is a swarm-based metaheuristic proposed by Dr. Karaboga in his seminal 2005 paper[1]. As its name suggests, ABC is inspired by the swarm behavior of honeybees when they are searching for food. Though its explorative ability has earned it some attention, a recurring problem is the original algorithm’s need of improvement when it comes to solution convergence or exploitation [2]–[5]. Fairee et al. (2018) improved the algorithm’s exploitation capabilities by adding reinforcement learning to the algorithm so it could focus on features which resulted in greater increase in fitness values [6], but his paper did not test his new algorithm on real-world cancer data.

To address both of those problems, this research project proposes a modified version of their R-ABC algorithm that adds the ability to samples points along a vector (VS-RABC, short for Vector-Sampling R-ABC0). After a number of unsuccessful tries by the scout bees to find new sources of food as in the existing R-ABC algorithm, the midpoint of the cluster of known solutions or food sources will be taken, then another point in the solution space will be randomly chosen. An n-dimensional Euclidean vector is then calculated between the midpoint and the newly-generated point to create a vector which will be scaled to a set distance, and used to calculate vectors back to the existing points which will be inspected at regular intervals for fitness. If one of those points has a greater fitness value than the existing point, reinforcement learning will be applied to make the bee more likely to search in that direction, and the distance scaling for the inspection of future new vectors will be increased. Conversely, if the newly-inspected point has a lower fitness value, the bee will be less likely to search in that direction, and the scaling distance will be decreased.

The VS-RABC algorithm will be used as a wrapper with two clustering and four classification algorithms for performing feature selection on the Acute Lymphoblastic Leukemia and Acute Myeloid Leukemia dataset gathered by Golub et al. [7]. Its accuracy will be assessed, and it will be compared with R-ABC and PSO used as wrappers as well as the Pearson Correlation Coefficient and Signal-to-Noise ratio. VS-RABC will be compared to RABC, on the basis of accuracy with respect to the number of fitness evaluations.

The key results are that VS-RABC outperforms RABC when used with K-Means clustering, and shows a slight improvement when used with SVM-RBF, but when it comes to other methods, the difference is fairly small. Both RABC and VS-RABC outperformed the Signal-to-Noise Ratio filter by a large margin, and returned higher accuracies than the Correlation Coefficient filter. Both R-ABC and VS-RABC were outperformed by PSO, which for most of the clustering or classification functions returned slightly higher average accuracies, and performed especially well with Agglomerative clustering and Decision Tree.

This thesis concludes with a brief mention of future work on possible methods for improving VS-RABC’s performance.

**Keywords: RABC, Cancer Gene Classification, Reinforcement Learning, Artificial Bee Colony**

**Acknowledgements**

I would like to thank my supervisor Dr. Iman and my co-supervisor Dr. Neoh for their patience and continued interest and contribution of ideas to my project. They very willingly made themselves available, and my discussions and meetings with them were an invaluable source of guidance.

I would be remiss if I neglected to thank my family for their understanding and support on the home front. Their familiar presence and love is the bedrock of all that I am and all that I hope to someday be.

**Introduction**

Though the Artificial Bee Colony algorithm is known for its explorative ability, the original algorithm’s need of improvement is well known in the realm of solution convergence or exploitation [2]–[5]. The Fairee et al (2018) attempted to enhance the exploitative ability of the ABC algorithm by proposing his own variant of R-ABC, or Reinforcement Artificial Bee Colony. His method has proven fairly successful, outperforming all the other variants of ABC he had chosen to compare it to on numeric benchmarks.

However, **to the best of my knowledge, Fairee et al.’s R-ABC has not yet been applied to real-world data as a wrapper for cancer gene classification on Golub Et. Al’s ALL-AML dataset.** There is a clear opportunity here to both test Fairee’s variant of R-ABC on well-known real-world data and at the same time attempt to improve the state of the art by modifying his algorithm to further improve its rate of convergence.

In short, our motivation is clear: Fairee’s variant of R-ABC requires testing on real-world data, and it is a new method in a rapidly-developing field that may have room for improvement.

The problems we seek to address are thus:

1. The absence of real-world data testing for Fairee’s variant of R-ABC
2. Finding out if it is possible to improve R-ABC even further

Our aims will thus be:

1. Writing an R-ABC algorithm from scratch
2. Gathering and linking other algorithms from open-source libraries to compare them with RABC
3. Applying R-ABC to real-world data and comparing it with other filter and wrapper algorithms
4. Modifying R-ABC to enhance its overall performance by proposing our own variant of it (Vector Sampling-RABC)

First, this dissertation will discuss the challenges that my work seeks to address, and then touch briefly on the clustering, classification, and wrapper algorithms that will also be used in the experiment. Then, we will delve into the parts of Fairee et al’s R-ABC algorithm that we are reusing. Next, we will discuss the modifications that VS-RABC will make to the original R-ABC algorithm and lay out the entire algorithm in a pseudocode and a flowchart.

After a brief section regarding the experiment setup and methodology, collated data from the experiment will be shown, and its results discussed. Once the accomplishments of this project have been summarized in the conclusion, a section on the possible further work that could be done with regards to this project is included.

**Literature Review**

Though the Human Geonome project was completed in 2003, it would be a mistake to assume that genetic research is a static field. In 2017, the CHESS gene set was created, adding over 100,000 new genes to existing databases, and in 2018, hundreds of genes were inserted into or removed from the Gencode list [8]. This continuously-evolving data landscape only fuels the need for better, faster, and more accurate techniques to help with processing and organizing the information we have.

One very exciting field of work is gene classification for diseases, which often involves small sample sizes (often only a few dozen genes) and very large dimensionalities (sometimes on the order of thousands of features)[9].

There is a great deal of research and articles regarding cancer and containing the key words “cancer genome”. A quick Google search returned 1.2 million searched items (as of March 8, 2018) with the keywords “cancer genome”. In Google scholar, 3.2 million results where “most of the cited items are cancer genome, proteomics, microarray, machine learning algorithms and others” [10].

A great deal of time and labor can be spent searching for correlations between genes and their expressions. Zhao et. al. [11] puts it most aptly: “The gene selection from gene expression data is challenging because of properties such as small sample size, large dimensions, and high noise. Clinical diagnoses require the selection of a small predictive subset of biologically relevant genes with a high classification accuracy for cancers.”

Filter methods such as the Signal-to-Noise ratio method, correlation coefficient method, Markov blanket filtering and the Redundancy Based Filter(RBF) method [12] have been proposed for the purpose of selecting relevant genes. At a glance, they appear simpler and less computationally expensive compared to wrapper methods, which must repeatedly evaluate their own result for fitness until the stop criteria are met:

Input Data

Filter

Classification/Clustering Algorithm

Output of weights and Accuracy

Accuracy

Input Data

Wrapper

Classification/Clustering Algorithm

Final Accuracy and output weights

Stop Criteria Met?

Yes

No

**Wrapper**

Diagram 1

Though filters can facilitate dimensionality reduction and select more statistically relevant genes, their interactions with each other and the classifier are ignored, and each feature is considered independently, causing interdependencies to be neglected. Overall, wrapper methods offer better feature selection and accuracy overall despite the increased computational cost[13].

Thus, we must also consider wrapper methods, which factor in the prediction accuracy with the selected features. Several examples of widely-used wrapper search strategies are Genetic Algorithms, Particle Swarm Optimization, and Sequential selection strategies for feature selection (SFS and SBS)[14]. R-ABC would fall under this class, as it repeatedly selects a subset of the feature space and tests it with the classifier/clustering algorithm to arrive at a subset with the highest classification accuracy or optimal objective function value.

The wrapper and filter feature selection methods could be used with clustering algorithms like k-means and Self-Organizing Maps (as in [7]), or classification algorithms like Naive Bayes, Decision Tree, and SVM (Linear and RBF). There is no perfect solution, as some algorithms lack the ability to explore combinations and permutations that are too distant from their initial starting points. The ones that do may have poor exploitation abilities [2]–[5].

Clearly, an algorithm that combines both explorative capability and the ability to quickly converge on a global minimum or maxima would be ideal, and at some point, it would have to be tested on real-world data. Thus, the main challenge my work would seek to address would be improving R-ABC’s overall performance, be it exploration or exploitation, on real-world data.

The well-known ALL-AML dataset by Golub et Al. [7] was selected as it would provide an indicator of what filter and wrapper methods would be able to accomplish with regard to a well-known real-world dataset. **Other ABC variants and hybrid approaches have been used for feature selection in cancer gene research**, for example, mRMR-ABC, as proposed by Hala Alshamlan et al. (2015) for cancer classification [15], a combination of a “Support Vector Machine Optimized by Particle Swarm Optimization and Artificial Bee Colony” Lingyun Gao. et al, (2017) [16], and an ABC algorithm that applied the concept of pheromones (from Ant Colony Optimization) by Moosa et al.(2016)[17].

**Clustering Methods**

*K-Means*

The K-Means clustering method was proposed by MacQueen in 1967. This unsupervised method of clustering assigns a centroid to each cluster and not only uses Euclidean distance to measure the differences between points, but may also use other distances such as Manhattan and Minkowski distances. to measure. In general, it operates as below according to J. Yadav et al [18]:

1) Randomly select k items from dataset D as the initial centroids for the clusters;

2) Assign each datapoint to the cluster to which the object is closet according to the mean value   
 of the object in cluster;

3) Calculate the new mean values of the data items for each cluster and update their respective   
 averages

4) Iterate until there are no further changes to the value of the average, or the centroid.

It should be kept in mind that this algorithm can be:

1. Fairly sensitive to the location of the initial selection of centroids, and
2. Easily affected by outliers.

*Agglomerative Clustering*

Agglomerative clustering is a hierarchal clustering algorithm that begins by considering each data object as a single cluster all its own (leaf node), and recursively merges the closest clusters until only one cluster which contains all the other clusters remain (root node). [19]

Like the K-means algorithm, it can be fairly sensitive to noise and outliers. Additionally, an object assigned to a cluster will not be reassigned, which implies that this algorithm is not capable of correcting misclassifications. The computational complexity of Hierarchal Clustering algorithms is generally *O(*), making them an unattractive option for large sets of data. [20]

***Classification Methods***

*Naive Bayes*

The Naive Bayes algorithm applies Bayes’ rule () to the simplistic assumption that the attributes are conditionally independent, given the class. Though this assumption is untrue, naïve Bayes nonetheless frequently delivers reasonably good classification accuracy, making it a method that we should not neglect to explore.

For categorical attributes like the data we possess, P() and P() are usually calculated by referring to frequency counts of features and outcomes stored in arrays, which are calculated from the training data to give the probability of outcomes given a certain set of conditions.

Even though it has low variance, and is fairly insensitive to noise due to its using all attributes in every prediction, caution must be exercised as it can have high bias. [21]

*Decision Tree*

Decision trees partition or split the input data based on a discrete function of the input data’s features. This process is repeated until a decision tree with one root node and a set of leaf nodes which only contain samples from one class. [22] For example, a decision tree might split the data based on the feature or attribute that provides the greatest increase in entropy, and then repeat this process with subsequent features that provide the greatest increase in entropy until its leaf nodes are pure.

Decision trees perform fairly well when using a few highly pertinent attributes, but will not do so well if there are many complex interdependencies or interactions.

Additionally, the greedy tendency of decision trees causes it to become overly-sensitive to noise and irrelevant attributes in the training set. [23]

*Linear- Kernel Support Vector Machine (SVM-L) and Radial Basis Function- Kernel Support Vector Machine (SVM-RBF)*

Support vector machines are a type of supervised type of machine learning algorithm that construct hyperplanes (n-1 dimensional planes in n-dimensional data) which partition data with n features or dimensions. With the labelled training data it is provided with, an SVM builds a model which can categorize subsequent data [24]. They are fairly robust classifiers too, being capable of outperforming MLPs (Multilayer Perceptrons) with most kernels [25]. Like most machine learning models, care must be taken not to overfit them.

It was necessary that both linear and radial basis function support vector machines were included in this assessment as the data might not have been linearly separable.

**Optimization Methods**

*PSO*

First introduced in 1995 by Eberhart & Kennedy, PSO creates a swarm of particles that moves through multiple dimensions to discover optima. Each particle represents a potential solution, and is also influenced by information that its neighbors discover. There has been much work done on applying, refining, and creating new variants of this method, which, like ABC, is a metaheuristic swarm optimization algorithm. [26]

As a metaheuristic swarm algorithm, it would be a suitable benchmark with which to compare R-ABC and VS-RABC.

*ABC Algorithm*

The basic ABC algorithm was first proposed by Karaboga & Akay (2005). Subsequently, numerous proposals have been made to augment its exploitative ability. Gao et. al. (2012)[27], Akay and Karaboga (2012)[28], Gao et al. (2015)[29], Kiran et al. (2015)[30], and Alshamlan et al. (2015) [31] are just a few examples which illustrate the amount of attention this field has received.

Saeed and Niknafs (2015)[32] and Ma and Zhang (2016)[33] have already worked on adding Reinforcement learning to ABC’s solution selection and updating strategies, but both papers were focused on Q-Learning-based Reinforcement learning.

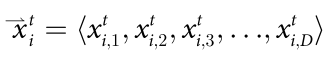
*R-ABC Algorithm*

The method proposed by Fairee S et al.(2018), which they have called the Reinforcement-Artificial Bee Colony method (**R-ABC**) [6] applies a classic Reinforcement Learning algorithm (the linear reward-penalty scheme, also known as the Bush-Mosteller scheme) to the Onlooker bees’ update equation, creating a new variant of the vanilla ABC algorithm. Since VS-RABC is a modification of R-ABC, and the employed and onlooker bee phases have been kept the same, the rough outlines of their work shall be given:

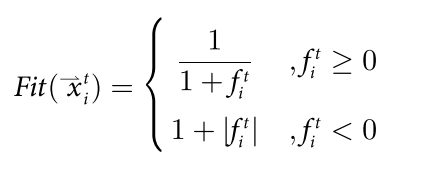
The RABC algorithm is constructed on the basic principles that the ABC is built on. Let there be a set of food sources , consisting of *SN* food sources.

The food source of will thus be , where :

(Equation 1)

­Each source of food will have D optimization parameters in a D-feature dataset. The parameter of will thus be , such that :

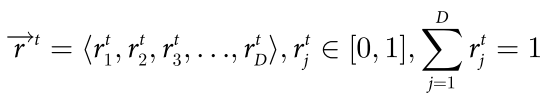
(Equation 2)

The algorithm will attempt to discover the optimization parameters that give the maximum value of the largest objective function value of all food sources, or, in other cases, the minimum value of its inverse:

(Equation 3)

Where is the objective function value of , and Fit() is the fitness value of x.

Since this paper is attempting to maximize the accuracy of the classifier, we may simply proceed with calculating and attempting to maximize .

Fairee S et al. added a reinforcement vector for members of the dataset into the R-ABC algorithm. For a D-Dimensional set of values:

(Equation 4)

will be used as the reinforcement vector for parameters of all food sources.

Food sources are instantiated at random as shown below, where is the lower-bound and is the upper-bound of the optimization parameter:



(Equation 5)

And the reinforcement vector for the *0th* iteration is given as below:



(Equation 6)

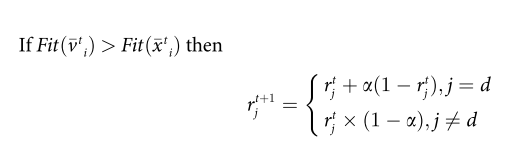
Employed bees discover neighboring food sources  of a current food source as shown below:



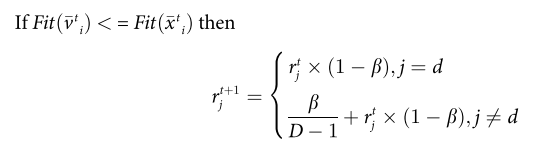
for all . (Equation 7)

An integer j is stochastically selected in the range [1, D]. The index of food source k is stochastically selected in the range of [1, SN], such that k≠i so that .

If a newly-discovered source of food returns a better yield or objective function value than the current food source that has an employed bee assigned to it, the current food source is forgotten and a new one is adopted.

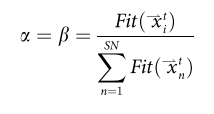
The reinforcement vector, for the following iteration is changed according to the linear reward-penalty scheme in Equations (8) & (9):

(Equation 8)



(Equation 9)

If the new food source provides a better fitness value, the employed bee will not only replace the current food source, but also increase the reinforcement value to the selected dimension while the reinforcement values of other dimensions are set to be smaller as in Eq (8). Conversely, if the current food source is still better, the new food source is discarded and the selected dimension is penalized by decreasing its reinforcement value as shown in Eq (9). Note that the sum of the reinforcement vector must always be 1.

α is the degree of reward, and β is the degree of penalty. They are given by:

Or, in the case where we are attempting to maximize the objective function values:

(Equation 10)

In the onlooker bee phase, each onlooker bee then proceeds to select a food source provided by the employed bee according to a probability based on the fitness values calculated as in Equation (3). An onlooker bee generates or locates a new food source using Equation (11), and replacing the current food source with a new food source if the new food source gives a higher return or reward:



(Equation 11)

Where ∊[-1,1] is a stochastically-generated real number and is the optimization parameter for the feature *k* of the global best food source.

**Proposed Modifications to Existing Algorithm**

Building on the work of Fairee S et al.(2018), I would like to propose a modification to the algorithm’s scout bee that might further facilitate its ability to explore and cover new ground.

In the R-ABC algorithm, the scout bee will randomly attempt to discover a food source if the number of tries by the employed and onlooker bee have exceeded a user-determined limit. There seemed to be a clear need for a more systematic approach, as the scout bee could actually end up discovering a food source that was worse than the initial one. Additionally, there was also the chance that the random point could be selected in an area with many other existing points with a low return, which would not be a sensible strategy if there was unexplored space.

To reduce the possibility of that happening, the following steps are performed. Diagrams 1 and 2, provide visual representation in 2D:

1. Calculate the Euclidean midpoint M of the cluster for all existing points occupied by employed bees.
2. Randomly instantiate a new point, using the scout bee

Center of existing cluster of food sources

Newly generated food source

Food source generated by Scout Bee

M

Diagram 2

1. Treat as a vectorpointing from M to a random direction.
2. Get the magnitude of , by
3. Scale the vector to a fixed, user-determined search distance (SD) and apply the reinforcement vector to it:

Newly Generated Food Source, =

(Equation 12)

1. Traverse the space between and the other existing points, sampling the space at a number of fixed points,

Center of existing cluster of food sources

Diagram 3

This would allow not only for the scout bee to escape local minima as in the original algorithm, but it would allow the bee to traverse and inspect previously uninspected space if the search distance was set to be far enough.

Keeping this in mind, the following two steps were added:

1. The search distance (SD) was set to increase every time the scout bee failed to find a better value when traversing this space, and to decrease every time the scout bee found a better food source
2. The failure or success in finding a better food source would also be fed back into the reinforcement learning reward scheme as in equations (8) and (9).

Doing so once might not be enough to discover a new and better source of food for the above process, especially for spaces with very large dimensionality. Therefore, the above process would be repeated several times, before the best value could be taken. Thus, we arrive at the pseudocode on the following page:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pseudocode:**  Algorithm 1: The Vector-Sampling R ABC Algorithm: | | | | | |  |
| Initialization: | |  |  |  |  |  |
| 1: | Randomly create a set of food sources | | | | |  |
| 2: | Initialize the Reinforcement vector | | | | |  |
| 3: | FOR the specified Number of Iterations: | | | | |  |
| 4: |  | BREAK if Best food source Accuracy/Return > 99.99% | | | | |
| Employed Bee: | | | |  |  |  |
| 5: |  | FOR all food sources: | | |  |  |
| 6: |  |  | Evaluate fitness of current position | | | |
| 7: |  |  | Update position | |  |  |
| 8: |  |  | Get Fitness value of new position | | | |
| 9: |  |  | IF fitness of new position > Current: | | | |
| 10: |  |  |  | Reset TRIAL | | |
| 11: |  |  |  | Replace current with new food source | | |
| 12: |  |  |  | Apply positive reinforcement to the reinforcement weights | | |
| 13: |  |  | ELSE |  |  |  |
| 14: |  |  |  | Increment TRIAL | | |
| 15: |  |  |  | Apply negative reinforcement to reinforcement weights | | |
| Onlooker Bee: | |  |  |  |  |  |
| 16: |  | FOR all food sources: | | |  |  |
| 17: |  |  | Select existing food source stochastically | | | |
| 18: |  |  | Update Employee bee position | | |  |
| 19: |  |  | Evaluate new position | | |  |
| 20: |  |  | IF Fitness value of new position > current: | | | |
| 21: |  |  |  | Reset TRIAL | | |
| 22: |  |  |  | Replace food source with the newly-discovered one | | |
| 23: |  |  | ELSE: |  |  |  |
| 24: |  |  |  | Increment TRIAL | | |
| 25: | |  |  | Apply negative reinforcement to reinforcement weights | | |
| Scout Bee: | |  |  |  |  |  |
| 26: |  | FOR each food source: | | |  |  |
| 27: |  |  | IF TRIAL >TRIAL Limit: | | | |
| 28: |  |  |  | FOR number of attempts=SCOUT\_MAXTRY: | | |
| 29: |  |  |  |  | Take Euclidean midpoint of all food sources, M | |
| 30: |  |  |  |  | Randomly select a new point | |
| 31: |  |  |  |  | Treat as a vector from the Midpoint M | |
| 32: | | | | | Scale the vector so that it has the same magnitude as the user-set distance | |
| 33: | | | | | Calculate the vectors from the end of to the other existing points and sample it at fixed intervals | |
| 34: |  |  |  |  | Get all the fitness values of all the points sampled | |
| 35: |  |  |  |  | IF fitness of any on vector point > fitness of current point | |
| 36: |  |  |  |  |  | Discard old point, replace with new one |
| 37: |  |  |  |  |  | Apply Positive Reinforcement |
| 38: |  |  |  |  |  | Increase the scaling distance, SD |
| 39: |  |  |  |  | Else: |  |
| 40: |  |  |  |  |  | Apply Negative Reinforcement |
| 41: |  |  |  |  |  | Decrease the scaling distance, SD |

The diagram on the next page provides a graphical summary of the above algorithm:

**Figure 1: Flowchart of the VS-RABC algorithm\***

YES

Retain current food source

Update Employee Bee position

New Fitness value better?

Generate Random Food sources and initial ranges

START

Generate Random Food sources and initial ranges

Stop Criteria Satisfied?

YES

END; return accuracy, features

NO

Negative reinforcement to chosen feature

Positive reinforcement to chosen feature

Reset TRIAL

Increment TRIAL

\*Reference: Fairee et. al [6]

**Employee Bee Phase**

NO

New Fitness value better?

**Onlooker Bee Phase**

NO

TRIAL++

Keep current food

Replace current food source with new one

Update positions of onlooker bees

Reset TRIAL

YES

NO

TRIAL> LIMIT?

**Scout Bee Phase**

YES

Generate a new point and treat it as a vector from the midpoint

Take midpoint of all food sources

Sample vectors at fixed intervals

Calculate vectors from to existing points

Scale the vector to the user-set distance

Fitness value on Any point >current?

Increase scaling distance, SD

Positive reinforcement to chosen feature

Discard current, replace with new

YES

NO

NO

YES

SCOUT\_MAXTRY>LIMIT?

Decrease scaling distance, SD

Negative reinforcement to chosen feature

**Approach to work and Methodology**

The project was written and implemented in Python, a widely-used object-oriented programming language. The clustering and classification algorithms, filter algorithms, and PSO program were all taken from widely-available libraries online, and linked to functions that I wrote myself.

The main benefit of this approach is that I don’t have to code well-known and open-sourced algorithms from scratch. I coded the R-ABC algorithm myself so that I could modify it whenever the need arose. In this approach, I prioritized functionality and ease of development for myself over ease of portability to other developers; at the moment, functionality is the only thing that matters.

The experiment was run on an Intel Core I5 8300H. A fixed random kernel for the k-fold partition was set. Though this does not guarantee that every experiment will turn out exactly the same, as the algorithms being tested rely on random searching, the main benefit of this approach is that it provides some assurance that variances caused by differences in testing and training datasets will be minimized or eliminated.

The cancer cell data was randomly partitioned into 2 folds to avoid overtraining as the dataset being examined is fairly small (only 72 samples). This process was repeated 10 times with the entire set of data to create 10 partitions of data with 2 folds each, and different data points in each fold.

An average was then taken for the final accuracies achieved by the wrapper and filter algorithms for each fold. The VS-RABC and RABC algorithms had their fitness values recorded at the beginning of every employee bee phase for each fold, and an average was calculated for each of those phases. The standard deviation of the averages of the results was also calculated.

Table 1 provides an overview of the algorithms and feature selection methods being used, and Table 2 provides a list of the parameters set for each of the feature selection methods:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Algorithms | | | | | |
|  |  |  | Clustering | | Classification | | | |
|  |  |  | K-means | Agglomerative Clustering | Naïve Bayes | Decision tree | SVM (Linear) | SVM (RBF) |
| Feature Selection Methods | Filter | Signal-to-noise | Signal-to-noise + K-means | Signal-to-noise + Agglomerative Clustering | Signal-to-noise + Naïve Bayes | Signal-to-noise + Decision tree | Signal-to-noise + SVM (Linear) | Signal-to-noise + SVM (RBF) |
| Correlation coefficient | Correlation coefficient + K-means | Correlation coefficient + Agglomerative Clustering | Correlation coefficient + Naïve Bayes | Correlation coefficient + Decision tree | Correlation coefficient + SVM (Linear) | Correlation coefficient + SVM (RBF) |
| Wrapper | PSO | PSO + K-means | PSO + Agglomerative Clustering | PSO + Naïve Bayes | PSO + Decision tree | PSO + SVM (Linear) | PSO + SVM (RBF) |
| R-ABC | R-ABC + K-means | R-ABC + Agglomerative Clustering | R-ABC + Naïve Bayes | R-ABC + Decision tree | R-ABC + SVM (Linear) | R-ABC + SVM (RBF) |
| VS-RABC | VS-RABC + K-means | VS-RABC + Agglomerative Clustering | VS-RABC + Naïve Bayes | VS-RABC + Decision tree | VS-RABC + SVM (Linear) | VS-RABC + SVM (RBF) |

Table 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Parameters | Symbol | Feature Selection Method | | | | |
| Wrapper | | | Filter | |
| VS-RABC | R-ABC | PSO | Correlation Coefficient | Signal-to-noise |
| Population size | N | 3 | 50 | 100 | - | - |
| Trial Limit | LIMIT | 1 | 20 | - | - | - |
| Initial Search Distance | SD | 85 | - | - | - | - |
| Traversal sampling interval |  | 12 | - | - | - | - |
| Max number of iterations | T | 4 | 30 | 77 | - | - |
| Individual Scaling Factor | Phip | - | - | 0.5 | - | - |
| Swarm Scaling Factor | Phig | - | - | 0.87 | - | - |

Table 2

All the algorithms were run until they reached the maximum value of 1.0, or until they reached their respective maximum thresholds, which were set to be as close to each other as possible. R-ABC was run up to 7,750 calls to the fitness function and PSO was run for up to 7,700 calls to the fitness function (100 particles x 77 iterations). In the case of VS-ABC, it was run for up to 7443 fitness calls, which was judged to be sufficiently close to the other algorithms’ fitness values to allow us to compare them fairly.

VS-RABC was given a population of only 3 bees over 4 iterations. Despite the relatively small number of iterations, those bees would perform many more fitness evaluations per iteration due to having to traversing the space multiple times and examining it at set intervals. Its trial limit was set to just 1 because most of the discoveries of new food sources on this dataset take place during the scout bee phase for this algorithm. The initial search distance for the VS-RABC was set to 85, approximately the unit Euclidean distance for the 7129-feature solution space:

(=.)

PSO was given a population size of 100 as well as 77 iterations – this would mean that it called the fitness evaluation 7,700 times. Phip and Phig were set after several trial-and-error attempts at determining the best combination for this dataset.

The accuracy values with respect to number of function calls will be compared for VS-RABC and R-ABC. However, the same comparison will not be made for PSO due to limitations in the third-party software library and the time limitations of this project.

The clustering algorithms were run with parameter weights set to unity passed directly to the clustering/classification algorithms as training weights **without** passing them through a filter or a wrapper. This was done to gauge the clustering algorithms’ baseline accuracy, as a completely ignorant user or system with no experience would simply feed the full set of features to those clustering algorithms.

The below values were averages obtained for 2-fold test-train data repeated 10 times over:

|  |  |
| --- | --- |
| Clustering Algorithms | Baseline accuracies (%) |
| K means | 67.21236 |
| Agglomerative Clustering | 69.34363 |
| Naïve Bayes | 98.06178 |
| Decision Tree | 87.27413 |
| SVM\_RBF | 93.04247 |
| SVM\_L | 95.58301 |

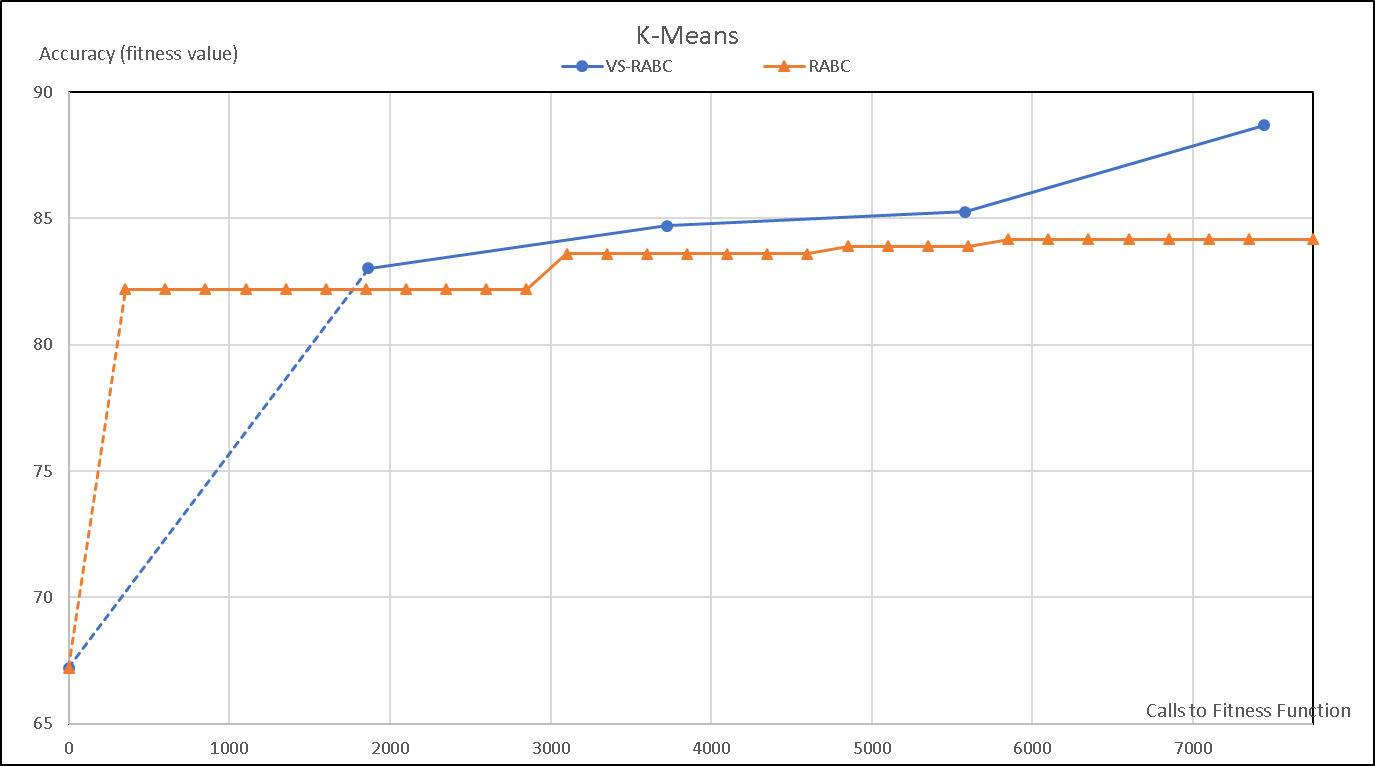
Table 3

These values will be used to complete graphs of accuracy against fitness for the algorithms when 0 fitness evaluations have been run, and they will serve as a baseline against which performance improvement can be measured.

**Results**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Clustering/Classification Algorithm | %Accuracy | Feature Selection Method | | | | |
| Wrapper Methods | | | Filter Methods | |
| VS-RABC | RABC | PSO | Signal-to-Noise Ratio | Correlation Coefficient |
| K-Means | Mean | 88.68726 | 84.1776 | 91.11969 | 65.2896 | 85.9073 |
| Minimum | 75.6757 | 65.7143 | 65.71429 | 64.8649 | 78.3784 |
| Maximum | 100.0 | 97.2973 | 100 | 65.7143 | 97.1429 |
| Standard Deviation | 6.6863 | 8.9790 | 10.86325 | 0.4247 | 6.8522 |
| Agglomerative Clustering | Mean | 91.5985 | 91.2973 | 96.36293 | 66.6409 | 84.0000 |
| Minimum | 71.4286 | 74.2857 | 88.57143 | 64.8649 | 75.6757 |
| Maximum | 97.2973 | 97.2973 | 100 | 78.3784 | 94.2857 |
| Standard Deviation | 8.2318 | 6.8504 | 2.865572 | 3.9329 | 5.4998 |
| Naive Bayes | Mean | 98.0618 | 99.1583 | 99.1583 | 65.8610 | 95.8224 |
| Minimum | 94.5946 | 97.1429 | 97.1429 | 54.2857 | 89.1892 |
| Maximum | 100.0 | 100.0 | 100 | 78.3784 | 100.0000 |
| Standard Deviation | 1.7536 | 1.2863 | 1.2863 | 7.5932 | 3.0522 |
| Decision Tree | Mean | 87.2741 | 87.2741 | 93.86873 | 61.0193 | 86.7490 |
| Minimum | 72.9729 | 72.9730 | 88.57143 | 42.8571 | 72.9730 |
| Maximum | 97.1429 | 97.1429 | 100 | 72.9730 | 94.2857 |
| Standard Deviation | 5.6725 | 5.6725 | 4.441222 | 8.3559 | 6.0343 |
| SVM-RBF | Mean | 97.7761 | 96.9653 | 98.85714 | 53.8610 | 90.6100 |
| Minimum | 94.2857 | 89.1892 | 94.28571 | 34.2857 | 78.3784 |
| Maximum | 100.0 | 100.0 | 100 | 77.1429 | 97.2973 |
| Standard Deviation | 2.0820 | 3.3208 | 1.895214 | 16.3714 | 5.5618 |
| SVM-L | Mean | 98.3629 | 99.1737 | 99.4440 | 73.8224 | 91.4054 |
| Minimum | 94.5946 | 97.1429 | 97.14286 | 60.0000 | 85.7143 |
| Maximum | 100.0 | 100.0 | 100 | 82.8571 | 97.2973 |
| Standard Deviation | 2.1704 | 1.2628 | 1.112505 | 7.3547 | 3.9646 |

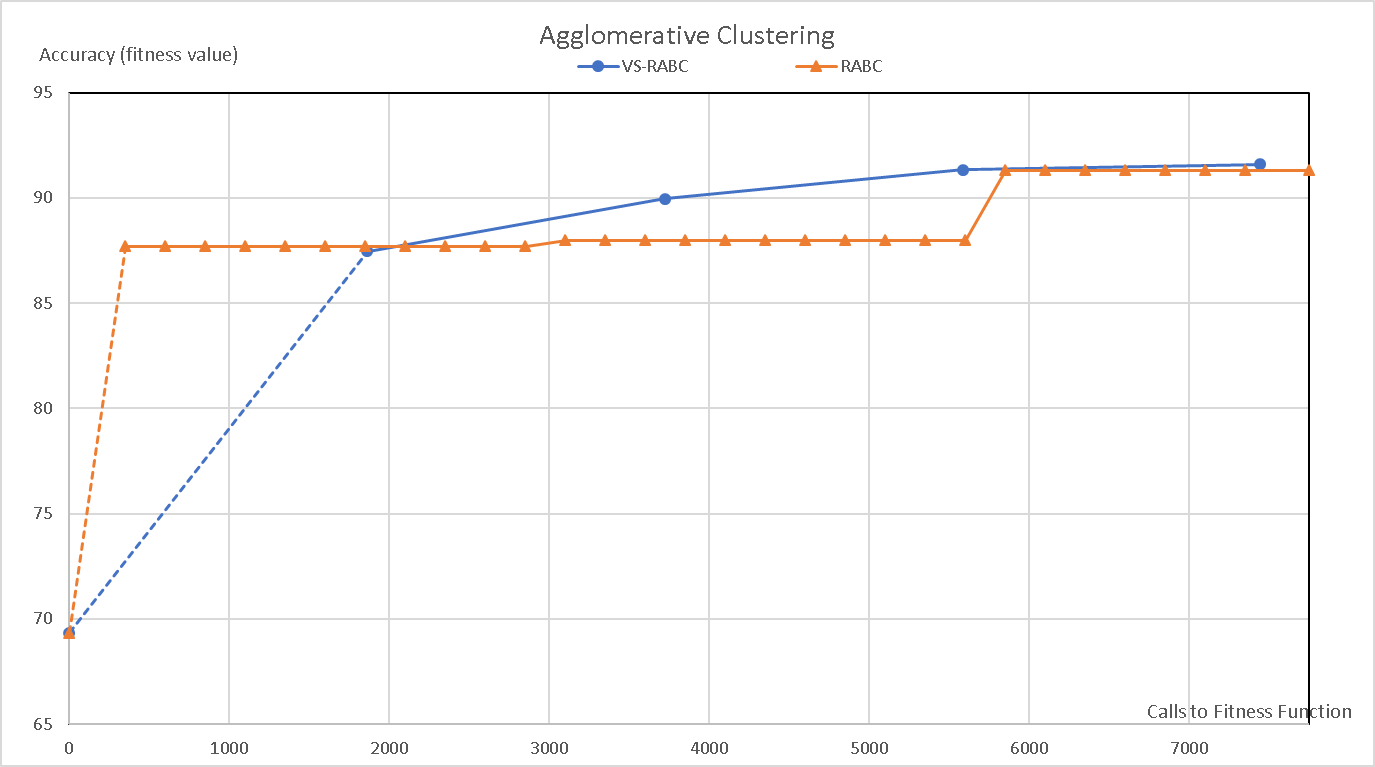
Table 4

Graph 1

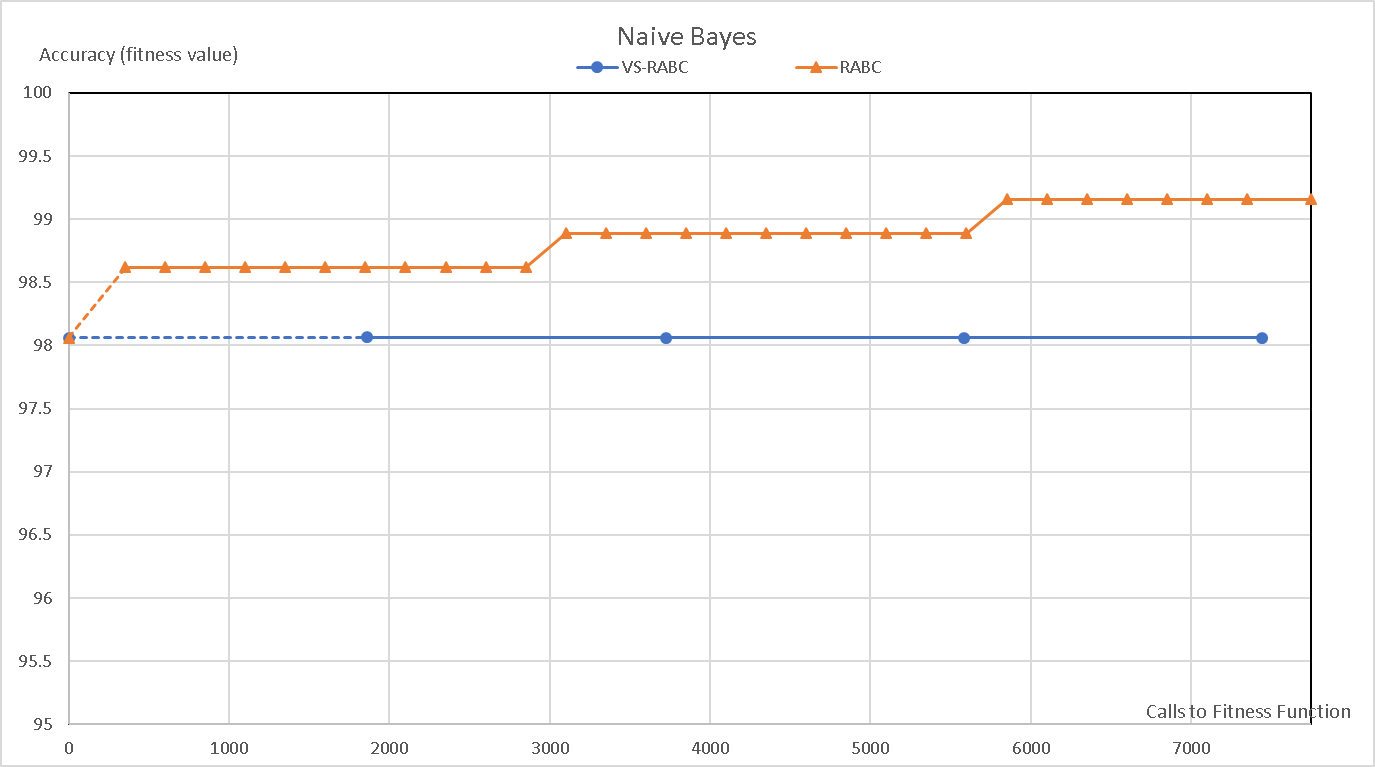
The dotted segments on the graphs of accuracy against calls to fitness function represent an extrapolation from the baseline accuracy for the clustering and classification algorithms. The points after that represent real data.

The above results clearly show that VS-RABC converges much more quickly than the normal R-ABC when used with the K-Means algorithm.

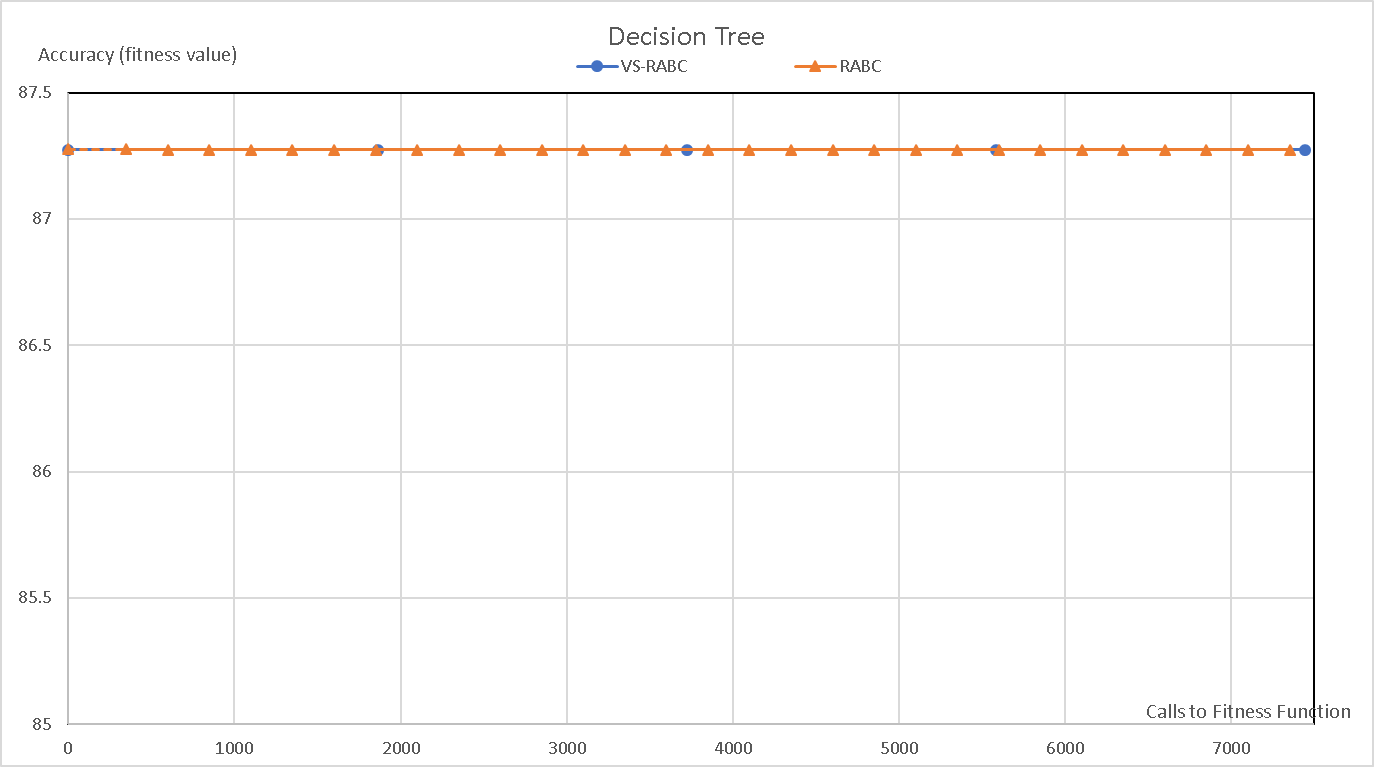
As mentioned in the literature review, the K-Means algorithm is particularly susceptible to the location of initialization of its centroid and to outliers in the data; thus, it comes as no surprise that out of all the clustering and classification algorithms, the K-Means algorithm has the highest standard deviation for RABC and PSO at 8.9790 and 10.86325 respectively. Of the three wrapper algorithms, VS-RABC had the lowest standard deviation in average accuracy at 6.6863 for this algorithm, which is still relatively high compared to the standard deviation values obtained for other clustering/classification methods.

Graph 2

The results for agglomerative clustering were fairly inconclusive. Though VS-RABC was initially quicker to converge in terms of accuracy, the two functions eventually reached roughly the same accuracy level. It must be noted, however, that VS-RABC has a higher standard deviation, and that it has a maximum value that is only slightly larger than R-ABC’s as well as a minimum value that is significantly smaller. VS-RABC might not perform as well with Agglomerative clustering as RABC does.

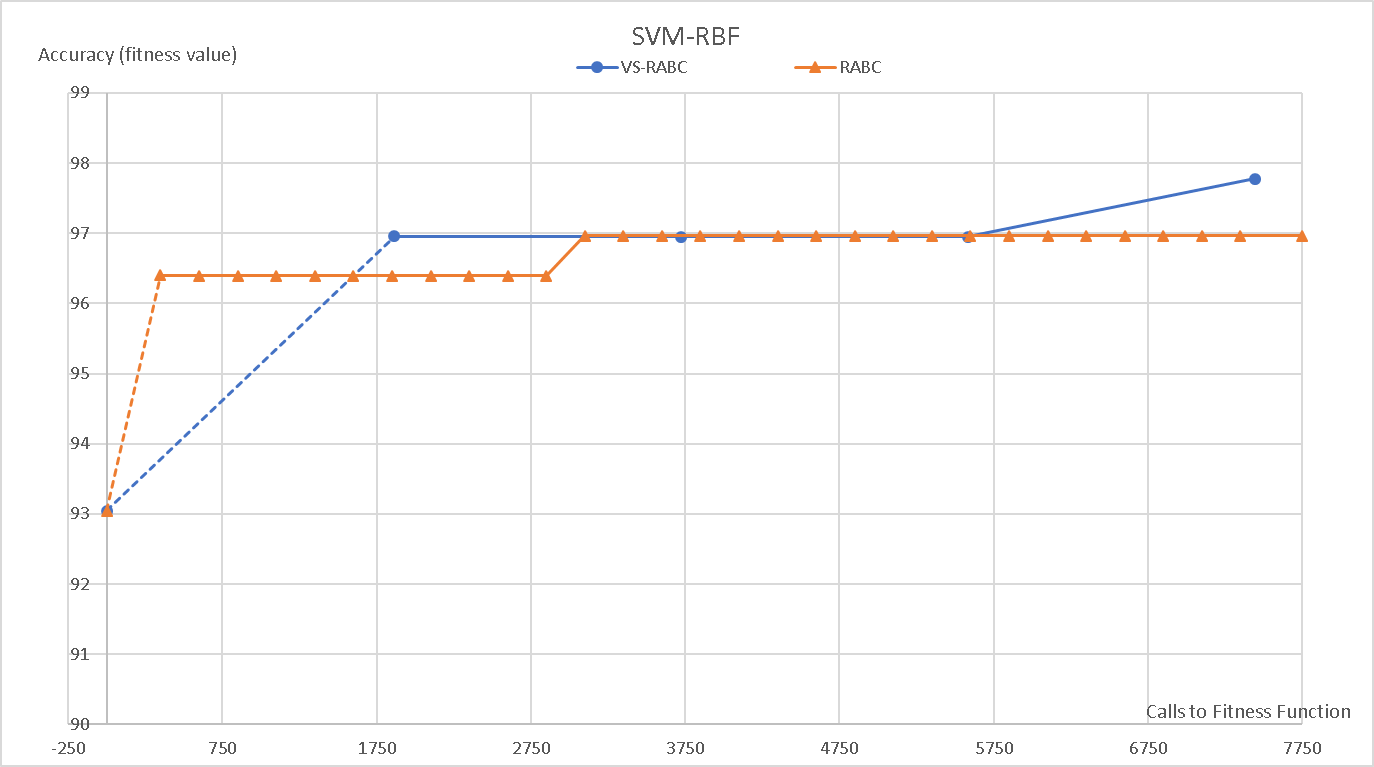
Graph 3

VS-RABC completely failed to demonstrate any improvement for Naive Bayes. It is suspected that this is due to the already high accuracy of the Naive Bayes algorithm, which, as shown in Table 3, has a baseline accuracy of 98.06% when supplied with unity weights without having any features removed by filters or wrappers.

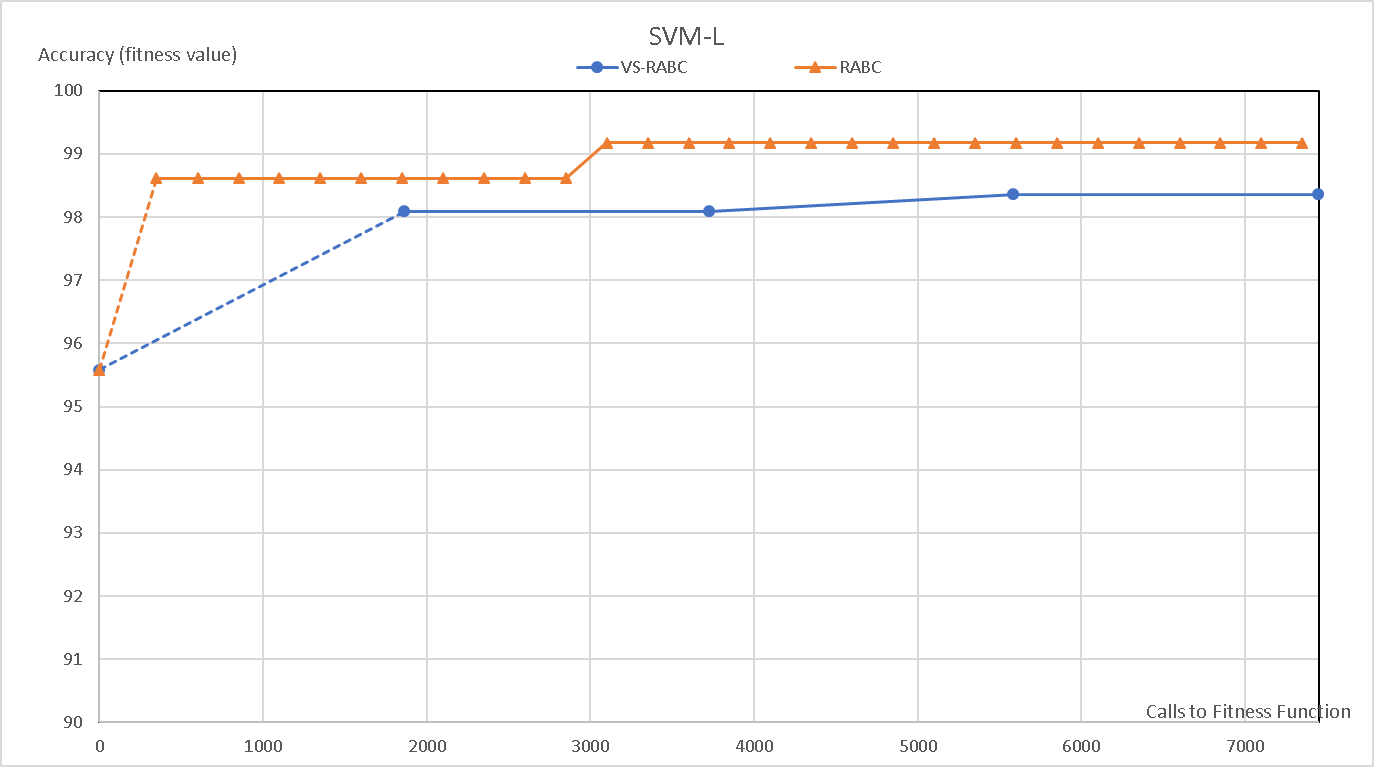
Graph 4

There is no change or improvement at all in the average accuracy for the Decision Tree. When inspected more closely, it was found that the accuracy of both VS-RABC and RABC did not vary with each iteration.

As mentioned earlier in the literature review, the Decision Tree can be fairly sensitive to noise and outliers in a dataset. Though the dataset is small, there are 7129 features that could contain a lot of irrelevant information, which could have the effect of masking any changes made to the selected data by the weights that the algorithms output.

Graph 5

VS-RABC appears to be performing very slightly better than RABC with SVM-RBF. VS-RABC’s higher minimum accuracy value and slightly smaller standard deviation appears to provide confirmation that SVM-RBF might be one of the methods that would work slightly better with VS-RABC than RABC.

Graph 6

RABC has higher mean, minimum values, and maximum values of average accuracy than VS-ABC here in addition to a smaller standard deviation.

**Conclusion**

Initially, the objectives of this project were:

1. To write an ABC algorithm from scratch
2. Gather and link other algorithms from open-source libraries to compare them to RABC
3. Apply R-ABC to real world data and compare it with other filter and wrapper algorithms
4. Modifying R-ABC to enhance its overall performance by proposing our own variant of it (Vector Sampling-RABC)

As of now, this project has:

1. Written an ABC algorithm
2. Gathered and linked other well-known and widely-used algorithms to compare them to RABC
3. Applied RABC to Golub Et Al’s (1999) cancer dataset, and compared its performance with other filter and wrapper algorithms. However, more work needs to be done to properly compare PSO with RABC and VS-RABC.
4. Created a variant of RABC (VS-ABC), which, though it shows promise in some areas, may need further work before it can be described as a properly-developed algorithm.

*Discussion of results and their significance:*

Though VS-RABC shows a clear advantage in performance over RABC when used with K-Means, demonstrates a very small advantage over RABC when used with SVM-RBF, and performs better than the correlation coefficient or signal-to-noise ratio filter methods across the board, the results are either inconclusive or very slightly in favor of R-ABC for the other methods. We will need to do further work on VS-RABC before we can say for sure whether or not the method can be significantly improved by our proposals.

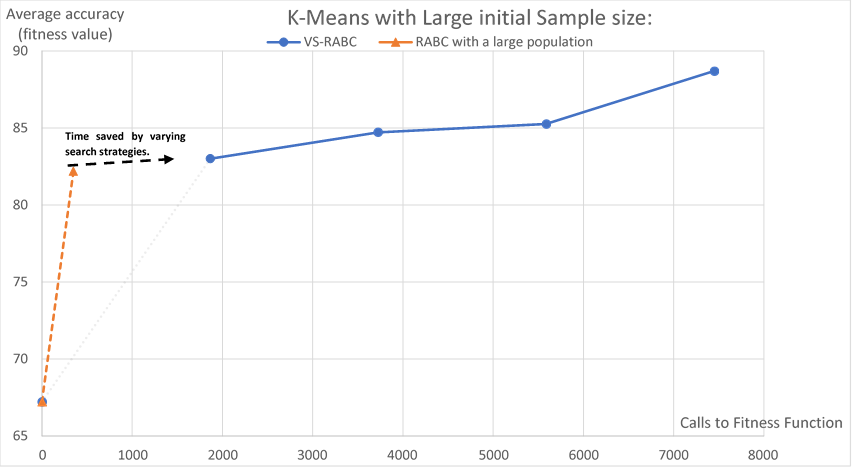
Both algorithms are outperformed by PSO on most of the other clustering/classification algorithms, with the exception of Naive Bayes and SVM-L, where R-ABC performs just as well as PSO in terms of overall accuracy when used with Naive Bayes, and only underperforms PSO by less than 0.3% when used with SVM-L. On the rest of the tests, PSO outperformed VS-RABC in terms of overall accuracy and had a smaller standard deviation with the exception of the K-Means algorithm, where PSO had the largest standard deviation. We need to focus on doing more work to test the VS-RABC and RABC algorithm against PSO for this particular dataset.

We began this project with two problems, namely the lack of real-world data testing for Fairee’s version of R-ABC, since it was only tested on numerical benchmarks, and a lack of knowledge as to how to improve R-ABC. The first problem has been resolved for filter methods – we now have a baseline with which we can compare Fairee et al’s R-ABC with other filter methods, at least, and a rough idea of its performance on real-world data. R-ABC outperforms filter methods on the dataset collected by Golub Et al (1999).

The second issue is a work in progress. Some headway has been made in finding out ways to tweak the algorithm’s behavior and improve its performance slightly, but when it comes to the issue of advancing the state of the art by improving on R-ABC, we can say that we remain tentatively optimistic on VS-RABC, but further work needs to be done to be done to make certain that the resulting modified algorithm is robust and well-suited for all types of optimization problems before any conclusive statements can be made.

**Possible Future Work**

Further comparisons need to be made between the average accuracy achieved with the number of calls to fitness function between wrapper algorithms such as PSO and other variants of the wrapper algorithms to the VS-RABC algorithm. Comparing them on other numerical datasets would go a long way towards settling the issue of this proposal’s feasibility conclusively.

From the data collected, it can be seen that though our method offers a very slight advantage when it comes to overall convergence with some clustering algorithms, initializing the algorithm with a large number of food sources appears to increase the best fitness value found for a comparatively small cost (measured in terms of calls to the fitness function). It is thus proposed that the search strategy should be varied by randomly generating either a fixed number of food sources at first (a hundred or so), or randomly generating food sources until the rate of increase in accuracy reaches a minimum level. The best points would then be selected for further evaluation, while the rest would be discarded or randomly retained to facilitate further searching.

Another obvious potential refinement to the proposed method needs to be mentioned: the scout bee generates a new point at a random distance from the center of the cluster formed by existing food sources, *which may still be unacceptably close to an area that is already densely populated by existing food sources.* Below is a case in point in 2 dimensions:

Center of existing cluster of food sources

Newly generated food source

Instead of randomly generating a point at a known Euclidean distance from the cluster’s center, we may obtain better results by generating a point that is not only at a known distance from the cluster’s center as proposed, but that is also as far away as possible from the other known food sources.

This would be accomplished by calculating the vectors from the known points to the center, and then calculating another vector that is as orthogonal to each of the vectors as possible. In a 2-dimensional case, this would be implemented by choosing one vector, calculating the dot product between that vector and each of the vectors, and repeating that process with the vectors produced until only one vector remains:

V1

Newly generated food source

V2

V3

Vn

The main aim of this is to increase the likelihood of new areas that were not previously investigated being investigated.

In the case of there being multiple areas with feasible or high-value food sources, it would make more sense to treat the areas as different clusters, and compute the new food source for them separately. A quick clustering separation algorithm like SVM - RBF will be run at regular intervals to check if any clusters of food sources or possible solutions exist in the current solution space. They could then each be treated as a separate cluster for the purposes of generating a new food source.

Center of existing cluster of food sources

Newly generated food source

Center of existing cluster of food sources

Newly generated food source

Of course, this treatment creates a question as to whether or not there should be a directed effort to explore the area between the clusters of known food sources, and if so, how.

Additionally, in the algorithm proposed in this paper, several parameters were user-determined and set to vary with each iteration or iteration outcome. It is hoped that in the future, these parameters can be varied adaptively by the algorithm for faster convergence or greater exploratory capabilities.

Finally, it must be noted that only Euclidean distance has been proposed in the modifications and further possible work to be done. More work may have to be done in exploring the viability of other types of distances (Manhattan, Minkowski, Cosine, etc.), and tests should also be run on standard optimization benchmarks like the Rastrigin and Griewank function.

**References**

[1] D. Karaboga, “An Idea Based on Honey Bee Swarm for Numerical Optimization,” Kayseri/Türkiye, 2005.

[2] S. Babaeizadeh and R. Ahmad, “Constrained artificial bee colony algorithm for optimization problems,” in *ADVANCES IN INDUSTRIAL AND APPLIED MATHEMATICS: Proceedings of 23rd Malaysian National Symposium of Mathematical Sciences (SKSM23)*, 2016.

[3] A. Singh and K. Deep, “Exploration–exploitation balance in Artificial Bee Colony algorithm: a critical analysis,” *Soft Comput.*, Sep. 2018.

[4] N. Sulaiman, J. Mohamad-Saleh, and A. G. Abro, “New Enhanced Artificial Bee Colony (JA-ABC5) Algorithm with Application for Reactive Power Optimization,” *Sci. World J.*, vol. 2015, pp. 1–11, 2015.

[5] W. Gao and S. Liu, “Improved artificial bee colony algorithm for global optimization,” *Inf. Process. Lett.*, vol. 111, no. 17, pp. 871–882, Sep. 2011.

[6] S. Fairee, S. Prom-On, and B. Sirinaovakul, “Reinforcement learning for solution updating in Artificial Bee Colony,” *PLoS One*, 2018.

[7] T. R. Golub *et al.*, “Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring,” *Science (80-. ).*, vol. 286, no. 5439, pp. 531–537, 1999.

[8] S. L. Salzberg, “Open questions: How many genes do we have?,” *BMC Biol.*, vol. 16, no. 1, p. 94, Dec. 2018.

[9] H. Hijazi and C. Chan, “A classification framework applied to cancer gene expression profiles.,” *J. Healthc. Eng.*, vol. 4, no. 2, pp. 255–83, 2013.

[10] M. D. Joseph M. De Guia, “Analysis of Cancer Classification of Gene Expression Data A Scientometric Review,” *Int. J. Pure Appl. Math.*, vol. 119, no. 12, pp. 12505–12513, 2018.

[11] GuodongZhao & YanWu, “Feature Subset Selection for Cancer Classification Using Weight Local Modularity,” Shanghai, 2016.

[12] H. Hu, J. Li, H. Wang, and G. Daggard, “Combined Gene Selection Methods for Microarray Data Analysis,” pp. 976–983, 2006.

[13] B. Kumari and T. Swarnkar, “Filter versus Wrapper Feature Subset Selection in Large Dimensionality Micro array: A Review.”

[14] N. El Aboudi and L. Benhlima, “Review on wrapper feature selection approaches,” *Proc. - 2016 Int. Conf. Eng. MIS, ICEMIS 2016*, pp. 1–5, Sep. 2016.

[15] H. Alshamlan, G. Badr, and Y. Alohali, “mRMR-ABC: A Hybrid Gene Selection Algorithm for Cancer Classification Using Microarray Gene Expression Profiling.,” *Biomed Res. Int.*, vol. 2015, p. 604910, Apr. 2015.

[16] L. Gao, M. Ye, and C. Wu, “Cancer Classification Based on Support Vector Machine Optimized by Particle Swarm Optimization and Artificial Bee Colony,” *Molecules*, vol. 22, no. 12, p. 2086, Nov. 2017.

[17] J. M. Moosa, R. Shakur, M. Kaykobad, and M. S. Rahman, “Gene selection for cancer classification with the help of bees,” *BMC Med. Genomics*, vol. 9, no. S2, Aug. 2016.

[18] J. Yadav, M. Sharma, and M. S. Jyoti Yadav, “A Review of K-mean Algorithm,” *Int. J. Eng. Trends Technol.*, vol. 4, no. 7, pp. 2972–2976, 2013.

[19] D. P. Dabhi, M. R. Patel, and M. R. P. Dipak P Dabhi, “Extensive Survey on Hierarchical Clustering Methods in Data Mining,” *Int. Res. J. Eng. Technol.*, vol. 03, no. 11, pp. 659–665, 2016.

[20] D. W. I. Rui Xu, “Survey of Clustering Algorithms,” *IEEE Trans. NEURAL NETWORKS*, vol. 16, no. 3, pp. 645–678, 2005.

[21] G. I. Webb, E. Keogh, R. Miikkulainen, R. Miikkulainen, and M. Sebag, “Naïve Bayes,” in *Encyclopedia of Machine Learning*, Boston, MA: Springer US, 2011, pp. 713–714.

[22] Anuradha and G. Gupta, “A self explanatory review of decision tree classifiers,” *Int. Conf. Recent Adv. Innov. Eng. ICRAIE 2014*, no. 3, pp. 1–7, May 2014.

[23] L. Rokach and O. Maimon, “Classification Trees,” in *Data Mining and Knowledge Discovery Handbook*, Boston, MA: Springer US, 2009, pp. 149–174.

[24] Ashis Pradhan, “SUPPORT VECTOR MACHINE-A Survey,” *Int. J. Emerg. Technol. Adv. Eng.*, vol. 2, no. 8, pp. 82–85, 2012.

[25] E. A. Zanaty, “Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification,” *Egypt. Informatics J.*, vol. 13, no. 3, pp. 177–183, Nov. 2012.

[26] R. A. P. I. Saptarshi Sengupta, Sanchita Basak, “Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives,” 2019.

[27] W. Gao, S. Liu, and L. Huang, “A global best artificial bee colony algorithm for global optimization,” *J. Comput. Appl. Math.*, vol. 236, no. 11, pp. 2741–2753, May 2012.

[28] B. Akay and D. Karaboga, “A modified Artificial Bee Colony algorithm for real-parameter optimization,” *Inf. Sci. (Ny).*, vol. 192, pp. 120–142, Jun. 2012.

[29] W. Gao, F. T. S. Chan, L. Huang, and S. Liu, “Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood,” *Inf. Sci. (Ny).*, vol. 316, pp. 180–200, Sep. 2015.

[30] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, “Artificial bee colony algorithm with variable search strategy for continuous optimization,” *Inf. Sci. (Ny).*, vol. 300, no. 1, pp. 140–157, Apr. 2015.

[31] H. M. Alshamlan, G. H. Badr, and Y. A. Alohali, “Genetic Bee Colony (GBC) algorithm: A new gene selection method for microarray cancer classification,” *Comput. Biol. Chem.*, vol. 56, pp. 49–60, Jun. 2015.

[32] S. Saeed and A. Niknafs, “Artificial Bee Colony-Fuzzy Q Learning for Reinforcement Fuzzy Control (Truck Backer-Upper Control Problem),” *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 24, no. 01, pp. 123–136, Feb. 2016.

[33] P. Ma and H.-L. Zhang, “Improved Artificial Bee Colony Algorithm Based on Reinforcement Learning,” in *International Conference on Intelligent Computing*, 2016, pp. 721–732.