Feature Selection Optimization using Binary Jaya Algorithm

Objective function: Area Under the Curve (AUC) Score

Table of Contents

Figures and Tables

Abstract

The purpose of this document is to prepare a step by step guide for Ajay. The algorithm includes binary Jaya with the Area Under the Curve (AUC) score as a feature function. These algorithms are used to optimized the feature selection process and improve the overall accuracy of classification results using Naive Bayes (NB), K-Nearest Neighbour (KNN), LDA, and Regression Tree (RT).

Introduction

The document contains the following:

- Datasets Musk and Madelon.
- Classifiers: NB, KNN, LDS, RT.
- Our Approach: Binary Jaya algorithm with AUC score as a function.
- Results

Dataset

Two datasets are used for classification purposes. Both datasets are available at https://archive.ics.uci.edu/ml/datasets

Dataset-1: Musk

Musk dataset statistics are shown in the table: 1.

This dataset is available at https://archive.ics.uci.edu/ml/datasets/Musk+(Version+2).

Dataset-2: Madelon

Madelon dataset statistics are shown in the table: 2:

This dataset is available at https://archive.ics.uci.edu/ml/datasets/Madelon.

Table 1: Musk Dataset Statistics

Content	Training Dataset	Test Dataset
Number of Features [F_{mu}]	168	168
Number of Measurements [M_{mu}^{Tr}]	6598	
Number of Measurements [M_{mu}^{Te}]		476

Table 2: Madelon Dataset Statistics

Content	Training Dataset	Test Dataset
Number of Features [F_{ma}]	500	500
Number of Measurements [M_{ma}^{Tr}]	2000	
Number of Measurements [M_{ma}^{Te}]		1800

List of Symbols

Table 3: Abbreviations

AUC Score	Area Under the ROC Curve Score
KNN	K-Nearest Neighbour
RT	Regression Tree
LDA	Linear Discriminant Analysis
FS	Feature Selection

Table 4: Nomenclature

P	Population Size
D	Minimum Number of Features
Random D	Length of a feature set
Fx	Objective Function
MaxIter	Maximum number of Iterations
S	Number of iterations
P	One population
X	Binary Population Set
F_{mu}	Features of Musk Dataset
F_{ma}	Features of Madelon Dataset
$M - Tr_{mu}$	Number of training measurements of Musk dataset
$M - Tr_{ma}$	Number of training measurements of Madelon dataset
$M - Te_{ma}$	Number of test measurements of Musk dataset
$M - Te_{mu}$	Number of test measurements of Madelon dataset

Classifiers

Four classifiers are used in this project with their default parameters are shown in the table below:

Table 5: Classifiers

Classifiers	Parameters
KNN	Algorithm = 'auto', leaf_size = 30, metric = 'minkowski' N_neighbors = 10 , p = 2, weights = 'uniform'
RT	Criterion = 'mse' , max_depth = None, min_samples_leaf = 1 , random_state = None, splitter = 'best'
NB	Priors = None, var_smoothing = $1e^{-09}$, tol = 0.0001, store_covariance = False
LDA	n_components = None, solver = svd, tol = 0.0001, store_covariance = False

Algorithm

This section will explain the proposed algorithm step by step, along with the code.

Step-1: Initialize Jaya Algorithm Parameters

Four parameters need to be initialized.

1. Population Size **(P)**: For Jaya algorithm execution, a specific number of population is required. The population size was *10*.
2. Number of Executions **(E)**: For result-averaging, $E = 1,2,\dots,20$. (the number of times you want to run the complete process).
3. Maximum Number of Iteration **(MaxIter)**: For the Jaya Algorithm number of iterations are required. $MaxIter = 10,\dots,100$.
4. The minimum number of features **(D)**: Features are dimensions. In my opinion, it is vital to consider a certain number of dimensions that accommodate 80-90 % of the variance.

Step-2: Define Content

Once parameters are initialized, it is essential to define the following:

Datasets

For training purposes, two sample datasets are used. Musk training dataset below:

$$X_{mu}^{Tr} = \begin{bmatrix} (1,1) & \dots & (1, F_{mu}) \\ \vdots & \ddots & \vdots \\ (M_{mu}^{Tr}, 1) & \dots & (M_{mu}^{Tr}, F_{mu}) \end{bmatrix} \quad 1$$

And Musk test dataset below:

$$X_{mu}^{Te} = \begin{bmatrix} (1,1) & \dots & (1, F_{mu}) \\ \vdots & \ddots & \vdots \\ (M_{mu}^{Te}, 1) & \dots & (M_{mu}^{Te}, F_{mu}) \end{bmatrix} \quad 2$$

Where $M_{mu}^{Tr} = 0,1, \dots, 6598$ and $M_{mu}^{Te} = 0,1, \dots, 476$ as shown in Table 1.

Similarly, Madelon training and test datasets are shown below:

$$X_{ma}^{Tr} = \begin{bmatrix} (1,1) & \dots & (1, F_{ma}) \\ \vdots & \ddots & \vdots \\ (M_{ma}^{Tr}, 1) & \dots & (M_{ma}^{Tr}, F_{ma}) \end{bmatrix} \quad 3$$

And Madelon test dataset above:

$$X_{ma}^{Te} = \begin{bmatrix} (1,1) & \dots & (1, F_{ma}) \\ \vdots & \ddots & \vdots \\ (M_{ma}^{Te}, 1) & \dots & (M_{ma}^{Te}, F_{ma}) \end{bmatrix} \quad 4$$

Where $M_{ma}^{Tr} = 0,1, \dots, 2000$ and $M_{ma}^{Te} = 0,1, \dots, 1800$ as shown in Table 2

Objective Function

For analysis purposes, two objective functions are in place.

Error Function

This objective function aims to select the optimal subset of the features that minimize the fitness function of the selected classification models. The error is computed as the difference between the actual output, and the selected model estimation is presented by eq.5, where $k = 1,2, \dots m$ and m is the number of testing observations $m = M_{ma}^{Te}$ for Madelon dataset and $m = M_{mu}^{Te}$ for Musk dataset. The fitness function values of the models are computed by dividing the summation of errors by the number of observations is presented by the eq.6

$$Error(k) = (\hat{y}(k) \neq y(k)) \quad 5$$

For each population p where, $p = 1,2, \dots, P$, the objective function (eq.6) is calculated using eq.5.

$$Fitness(p) = \frac{\sum_{k=1}^{m} Error(k)}{m} \quad 6$$

Area Under the ROC Curve (AUC) Score

It is essential to understand the ROC curve before applying the AUC score. A ROC curve (receiver operating characteristic curve) shows a classification model's performance at all classification thresholds. This curve plots two parameters:

- True Positive Rate (TPR)

$$TPR = \frac{TP}{TP+FN} \quad 7$$

- False Positive Rate (FPR)

$$FPR = \frac{FP}{FP+TN} \qquad 8$$

AUC measures the entire two-dimensional area underneath the entire ROC curve. AUC is desirable for the following two reasons:

- AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values as done in eq.5.
- AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

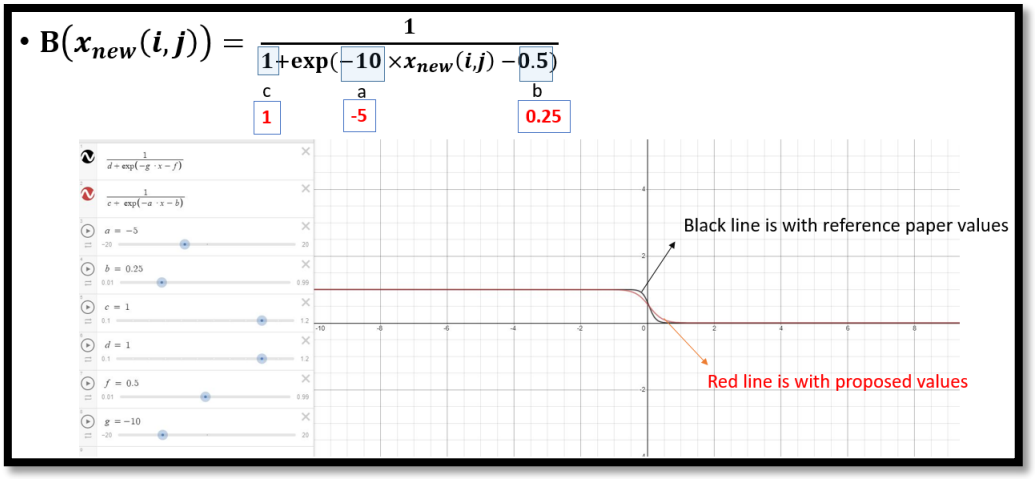
Moreover, the error function is based on the classifier's accuracy only, leading to an accuracy paradox. Secondly, when there is data imbalance, accuracy might increase due to only one class's influence. For comparison purposes, both objective functions are used.

S-Function (Sigmoid Function)

The binary JAYA population is reconstructed using the JAYA algorithm using an objective function and selected features. However, the resulting values of the solutions are continuous. The sinusoidal transfer function (or S-shape transfer function as drawn in Figure 1 is used to convert them to binary values (J & Eberhart, 1997). In practice, the new solution vector $X[d]$ will be entered into eq.9 and eq.10 to calculate its binary vector.

$$B\left(X_{new}(p,d)\right) = \frac{1}{1+ exp^{(-5 \times X_{new(p,d)} - 0.25)}} \qquad 9$$

Figure 1: S-shaped function for converting objective function output as a binary number



$$X_{new} = \begin{cases} 1, & B\left(X_{new}(p,d)\right) > rand() \\ 0 & Otherwise \end{cases} \qquad 10$$

Here, rand() is the random number distributed between the range of 0 and 1.

Classifier

For objective function, we need to define a classifier. Any classifier can be selected from **Table 5**.)

Step-3: Execute Binary Jaya Algorithm

Step-A: Generate Binary Population

Construct the initial binary populations. In this step, each binary population is generated using a random selection of features. Later on, the fitness function cost of each generated population is calculated.

Step-B: Determine Best and Worse Solutions

In this step, the best and the worse solutions are selected based on the objective function's value.

- **Objective function = error function:** The minimum value of the objective function will lead towards the best solution. The maximum value of the objective function will determine the worse value.

- **Objective function = AUC score:** The maximum value of the objective function will lead towards the best solution. The minimum value of the objective function will determine the worse value.

Step-C: Improvement Process

In this step, every decision variable of every solution at each iteration is stochastically modified using the JAYA operator formulated in eq.11 (Awadallah, Al-Betar, Hammouri, & Alomari, 2020).

$$X_{new}(p,d) = x(p,d) + r1 \times (bestX - |x(p,d)|) - r2(worseX - |x(p,d)|) \quad 11$$

Step-D: Selection Process

When the new solution X i is completely generated, it compares to the current one Xp stored in the population's pth position. Suppose the fitness value of Xnew is better than the fitness value of Xp. In that case, the new solution replaces the current one in the population.

This new fitness is compared with the old fitness.

- **Objective function = error function:** If the new fitness value is less than the old fitness value, its corresponding population is selected for the next iteration and otherwise dropped.
- **Objective function = AUC score:** If the new fitness value is more than the old fitness value, its corresponding population is selected for the next iteration and otherwise dropped.

Step-E: Stop Criteria

Stop criterion. Steps B to D are repeated until either one of the following conditions is satisfied:

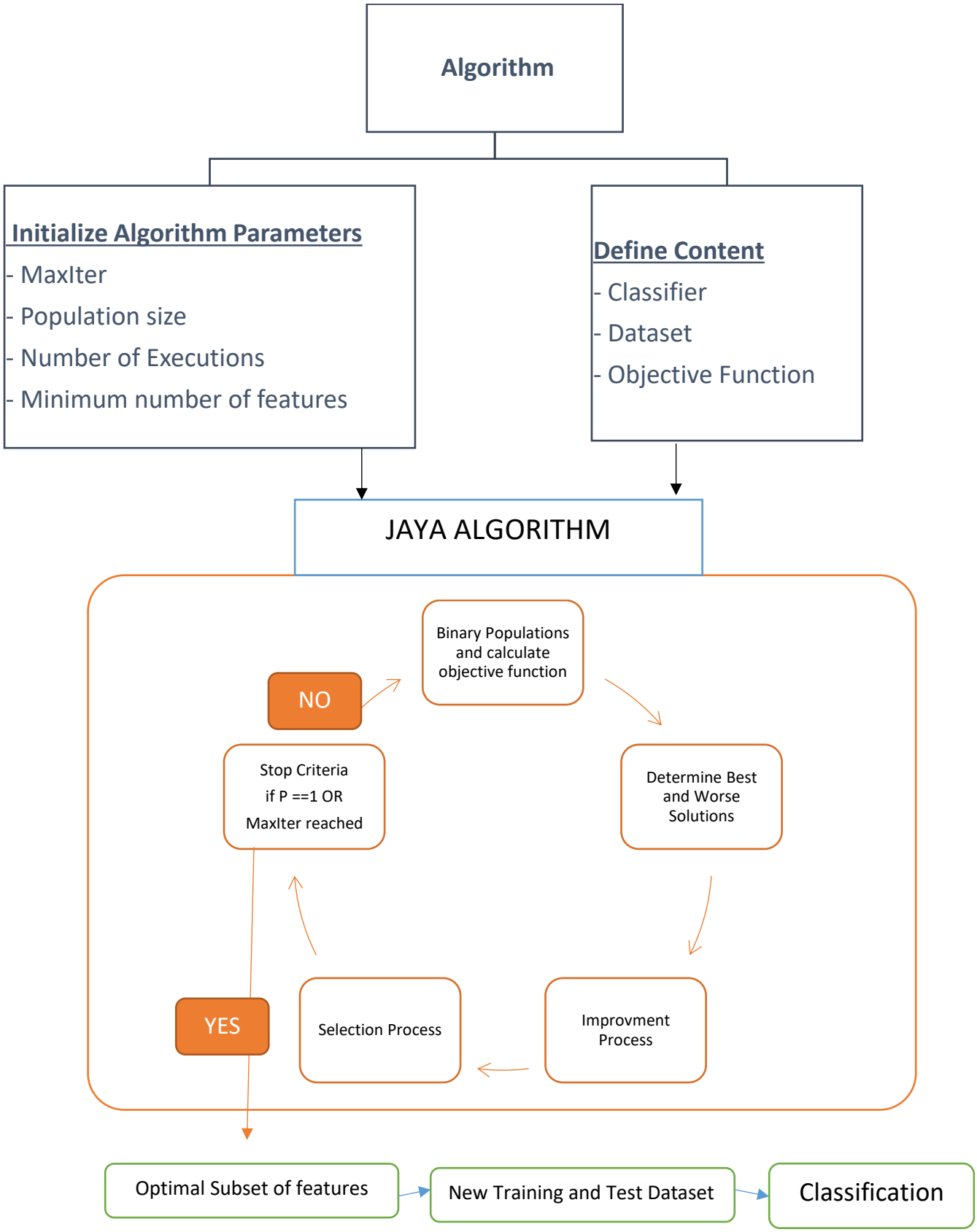
- the maximum number of iterations (i.e., Max_itr) are satisfied.
- Only one population is left.

Algorithm

Algorithm: The pseudo-code of the Proposed Algorithm

1. ---------- Initialize parameters and content ----------
2. Initialize Jaya algorithm parameters. Population Size (**P**), Number of Executions (**E**), Maximum Iterations (**MaxIter**), Minimum number of Features (**D**).
3. Load Datasets Train (Tr), Test (Te)
4. Initialize content. Objective Function, S-Function, Classifier
5. ---------- Initialize the binary population ----------
6. Generate binary population $X_{(i,j)}$ $\forall\ p = 1,2,\dots,P$ and $\forall\ d = 1,2,\dots,D$ features.
7. Calculate $f(X)$
8. ---------- Execution of Jaya Algorithm ----------
9. **For** $a = 1, 2, \dots, E$ **do**
10. **For** m $= 1, 2, \dots, MaxIter$ **do**
11. **if** $P > 1$ **do**
12. Determine the best solution in the population (X_{best})
13. Determine the worst solution in the population (X_{worst})
14. **For** $p = 1,2,\dots,P$ **do**
15. **For** $d = 1,2,\dots,D$ **do**
16. $r1 = rand(0,1)$
17. $r2 = rand(0,1)$
18. $\text{Jaya}\ (p,d) = \text{x}(p,d) + \text{r1}\ \times (\text{bestX} - |\text{x}(p,d)|) - \text{r2}(\text{worseX} - |\text{x}(p,d)|)$
19. ------ binary conversion using S-Function ------
20. $\text{B}\big(\text{X}_{\text{new}}(\text{p},\text{d})\big) = \frac{1}{1+\exp^{(-5\times \text{X}_{\text{new(p,d)}}-0.25)}}$
21. **End for**
22. Calculate $f'(X_{new}(p,d))$
23. **End for**
24. --------- population update ----------
25. **If** $f'(X_{new})) > f(X))$ **do**
26. $f(X) = f'(X_{new})$ // update objective function
27. **Else** remove $X(p,d)$ // remove population
28. $P = len(X)$ // *update* ***P***
29. **Else** STOP **for** // for m
30. **End for**
31. **End for**
32. ---------- Optimal Subset of Features ----------
33. Re-arrange Train and Test datasets according to optimal features
34. Classify

Flow Diagram



Results and Conclusion

SL. No	Datasets	FS Algorithms	NB			KNN			LDA			Regression Tree		
			PR	Re	F1	PR	Re	F1	PR	Re	F1	PR	Re	F1
1	Madelon	Without FS	0.496	0.496	0.495	0.504	0.503	0.496	0.513	0.513	0.513	0.498	0.498	0.498
		FS-JAYA [Error Function]	0.501	0.501	0.501	0.4988	0.498	0.4935	0.501	0.501	0.501	0.506	0.506	0.506
		FS-JAYA [AUC Score]	**0.5078**	**0.5078**	**0.5075**	0.5012	0.5011	0.495	0.5011	0.5011	0.5009	**0.5094**	**0.5094**	**0.5094**
2	Musk	Without FS	0.704	0.660	0.657	0.909	0.902	0.905	0.784	0.783	0.784	0.683	0.686	0.678
		FS-JAYA [Error Function]	**0.720**	**0.700**	**0.7032**	0.892	0.877	0.882	0.71	0.75	0.755	**0.724**	**0.723**	**0.723**
		FS-JAYA [AUC Score]	**0.73**	**0.707**	**0.710**	0.876	0.859	0.8649	0.70	0.69	0.69	**0.792**	**0.794**	**0.7933**

Results show improvement with the proposed algorithm. As mentioned earlier, the AUC score as an objective function can add logical value to the results, especially if the classes' data division is not balanced. Each error function rationale and advantage has been shared already.

Algorithm Rationale

This algorithm's ultimate purpose is to select the best features without compromising the classification results in terms of overall accuracy and individual class detection accuracy. Jaya algorithm is an optimization algorithm. The selection of its parameters and objective function plays a vital role in a classification problem. In the proposed algorithm, parameters and objective functions are selected based on maintaining the classifier's performance for overall accuracy and individual classes.

- **The minimum number of features:** it is essential to select at least a specific feature set, which covers 80-90% of data variance in a hyperspace. This is based upon the dimensionality reduction algorithm.
- **Area Under the Curve:** This objective function focused on a balanced ratio between True Positive Class detection versus True Negative Class detection. This is important as this function makes the complete feature selection process independent of any one-class impact. The value of the AUC score is higher when both classes detection rate is comparable. AUC score is a better option when the datasets are not balanced.
- **S-Function (Sigmoid Function): S-**function values are finalized using exhaustive search.

The result table above shows a definite improvement in overall classifiers performance with the above parameters and AUC score as an objective function compared to the classifier's performance with Error rate.

Moreover, parameter tweak to the binary conversion function (S-function) does impact the overall result. For future work, this function can be replaced with any of the following:

- Logistic function

$$f(x) = \frac{1}{1+e^{-x}}$$

- Hyperbolic tangent (shifted and scaled version of the logistic function, above)

$$f(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Arctangent function

$$f(x) = \arctan x$$

- Gudermannian function

$$f(x) = \operatorname{gd}(x) = \int_0^x \frac{1}{\cosh t}\,dt = 2\arctan\left(\tanh\left(\frac{x}{2}\right)\right)$$

- Error function

$$f(x) = \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt$$

- Generalised logistic function

$$f(x) = (1+e^{-x})^{-\alpha}, \quad \alpha > 0$$

- Smoothstep function

$$f(x) = \begin{cases} \left(\int_0^1 (1-u^2)^N\,du\right)^{-1} \int_0^x (1-u^2)^N\,du, & |x| \le 1 \\ \operatorname{sgn}(x) & |x| \ge 1 \end{cases} \qquad N \ge 1$$

- Some algebraic functions, for example

$$f(x) = \frac{x}{\sqrt{1+x^2}}$$

References

Awadallah, M. A., Al-Betar, M. A., Hammouri, A. I., & Alomari, O. A. (2020). Binary JAYA Algorithm with Adaptive Mutation for Feature Selection. *Arabian Journal for Science and Engineering*.

Das, H., Naik, B., & Behera, H. (2020). A Jaya algorithm based wrapper method for optimal feature selection in supervised classification. *Journal of King Saud University - Computer and Information Sciences*.

J, K., & Eberhart. (1997). A discrete binary version of the particle swarm algorithm. *IEEE International Conference on Systems, Man and Cybernetics. Computational and Cybernetics and Simulations, 5*, 4104-4108.