# An Adaptive Memetic Algorithm for Feature Selection using Proximity Graphs

Amer Abu Zaher, Regina Berretta[1], Nasimul Noman, Pablo Moscato

School of Electrical Engineering & Computing
The University of Newcastle,
University drive, Callaghan, 2300, Australia

Amer.AbuZaher@uon.edu.au,[Regina.Berretta, Nasimul.Noman, Pablo.Moscato]@newcastle.edu.au

[1] Corresponding author

**Abstract**

We propose a multivariate feature selection method that employs proximity graphs for assessing the quality of feature subsets. Initially, a complete graph is built, where nodes are the samples, and edge-weights are calculated considering only the selected features. Next, a proximity graph is constructed based on these weights and different fitness functions, calculated over the proximity graph, evaluate the quality of the selected feature set. We propose an iterative methodology based on a memetic algorithm for exploring the space of possible feature subsets aimed at maximizing a quality score. We designed multiple local search strategies and we employed an adaptive strategy for automatic balancing between the global and local search components of the memetic algorithm. The computational experiments were carried out using four well-known datasets. We investigate the suitability of three different proximity graphs (MST, K-NN and RBG) for the proposed approach. The selected features have been evaluated using a total 49 classification methods from an open source data mining and machine learning package (WEKA). The computational results show that the proposed adaptive memetic algorithm can perform better than traditional genetic algorithms in finding more useful feature sets. Finally, we establish the competitiveness of our approach by comparing it with other well-known feature selection methods.

**Keywords**: feature selection, proximity graph, minimum spanning tree, memetic algorithm, evolutionary algorithm

## 1. Introduction

The exponential growth of available data has placed an increasing demand to computer scientists. More effective and efficient techniques are required for data analysis. One particular challenge in some problems domains is that the number of features, many of which are irrelevant or redundant, significantly exceeds the number of samples [1]. An effective method to handle the 'curse of dimensionality' is to project a high-dimensional data onto a smaller dimension without losing features which are relevant for classification.

The fundamental problem of dimensionality reduction is to identify meaningful low-dimensional spaces of interest from data in high dimensions. Several approaches exist, perhaps the most popular is *feature selection*. The aim, in this case, is to find a subset of the features that may best maintain the "core information" existing in the entire dataset without losing important characteristics [2]. In other words, when samples in the dataset belong to a finite set of distinct "classes" (they are "labelled'), a feature selection approach attempts to identify the best subset of features that can discriminate between any pair of samples that have different "class labels".

Over the last decades, many methods for features selection have been proposed which can be broadly categorized in one of the following classes: *filters*, *wrappers* or *hybrid*, based on their approach for selecting features.

The *filter* approaches, in which the best subset is selected by using some predetermined selection criteria or evaluation function, tend to be simpler to implement, faster in practice, and often proven to be among the algorithmically most efficient methods. These approaches work independently of any learning algorithm that can employ the selected features; therefore, they may fail to select feature subsets that could show good generalisation performances when used in conjunction with a good classification algorithm. Filter approaches can be *univariate* or *multivariate*. *Univariate* filter methods start by individually ranking each feature by using a statistical test or a predefined criterion, then use some ad hoc method to select the best-ranked features [3]. Examples of univariate filter-based feature selection methods can be found in [4, 5]. Understandably, such methods do not consider the pairwise correlation between feature values across the set of samples. On the other hand, *multivariate* approaches work by ranking a subset of features (instead of individual features) by searching for the combination of features with the highest rank. Some examples of multivariate based approaches include correlation-based feature selection (CFS) [6], INTERACT [7], Relief [8] and ReliefF [9], MRMR[10], $M_d$ filter [11] SFS-LW [12], LLE score [13] and an approach based on the ($\alpha$,$\beta$)-*k*-feature set problem [14].

In *wrapper* approaches, the selection process is an iterative combination of search, learning and evaluation phases. This method employs a classifier to evaluate the selected subset of features. The goal is to select the subset of features best suited to the learning algorithm, therefore, the classification performance of wrapper approaches are better than filter approaches. However, wrapper methods are computationally more expensive than filter methods and may overfit the training data especially in small datasets, which is a serious problem from the classification point of view [3]. Recently, several nature inspired algorithms have been proposed for wrapper based feature selection [15-17].

Finally, the *hybrid* approaches are characterized by combining techniques from both filter and wrapper methods allowing an iteration between the feature selection process and the learning algorithm. Basically, hybrid approaches use the filter method to minimise the search space, and the wrapper method to choose the best reduction. For example, Xie et al. [18] used a Fisher score and sequential forward floating search as a filter method, and an SVM classifier from the wrapper method to attain a more accurate and efficient subset. In a recent work, a hybrid approach has been proposed that filters by maximizing relevance, minimizing redundancy and maximizing synergy of the candidate features, then it is combined with a real-coded genetic algorithm [19].

In this work, we propose a novel filter multivariate feature selection approach that employs a proximity graph to evaluate the quality of a subset of features. We first construct a complete weighted graph in which each node represents one of the samples, and the edge weights indicate the distances between samples calculated

using a subset of features. Next, a proximity graph is calculated, and the subset of features is evaluated using a fitness function computed over the proximity graph's topological structure. For instance, the number of edges connecting samples from different classes is one natural quality score that can be used to evaluate the quality of the subset of features. Other fitness functions are also explored in this work.

We developed a memetic algorithm (MA) [20] for searching the subset of features. Several local search heuristics were designed and embedded in the framework of a genetic algorithm (GA). To allow global exploration and local exploitation, an adaptive scheme was incorporated within the proposed memetic framework. Preliminary versions of the current work were presented in [21] and [22], where only the Minimum Spanning Tree (MST) was used as the proximity graph and/or only a GA was used as the search technique, respectively. The effectiveness of our approach is verified using four real-world datasets available on the public domain. The quality of the selected features was evaluated by several classification algorithms and a comparison was carried out with existing feature selection methods.

The structure of this paper is described next. In Section 2, the proposed approach using proximity graphs for feature selection is explained in detail. Section 3 presents the local search algorithms, the proposed memetic framework and the adaptive mechanism responsible to deliver a synergy between the global and local component of the memetic algorithm. Section 4 presents the computational experimental results and comparative studies. Finally, Section 5 presents some discussion on the results and Section 6 concludes the paper.

## 2. Feature selection using proximity graphs

Given a complete graph $G(V,E,W)$, a subgraph $G_{prox}(V, E_{prox}, W_{prox}) \subseteq G$ is considered to be a proximity graph if and only if the relationship between two nodes satisfies particular requirements [23]. The family of proximity graphs includes: the Minimum Spanning Tree (MST), the K-Nearest Neighbours graph (K-NN), the Relative Neighbourhood Graph (RNG), the Gabriel graph (GG) and the Delaunay Triangulation (DT).

According to Carreira-Perpinán et al. [24] and Inostroza-Ponta et al. [25], proximity graphs are useful constructs for clustering and manifold learning because they represent neighbour relationships between objects. If we construct proximity graphs with samples as nodes and edges with a weight that represent the distance between samples then it is reasonable to centre the feature selection procedure around such representation in order to generate more meaningful relationships between these objects when some class membership information is also known.

In this work, we use three proximity graphs: Minimum Spanning Tree (MST), K-Nearest Neighbours (K-NN) and Relative Neighbourhood Graph (RNG) to address the problem of feature selection and then study the effect of their outputs (subsets of features) on classification performance.

First, let's define the proximity graphs used this work. Given $G(V,E,W)$, a spanning tree is a sub-graph of $G$ that links all nodes with $(n–1)$ edges. There are several possible spanning trees for a given weighted complete graph $G(V,E,W)$. The MST, denoted by $G_{MST}(V, E_{MST}, W_{MST})$, is the spanning tree in which the total sum of weights of the $(n–1)$ edges is the lowest among all possible spanning trees. Two well-known algorithms for constructing the MST are Kruskal's and Prim's algorithms. The K-NN graph (denoted as $G_{KNN}(V, E_{KNN}, W_{KNN})$) is a subgraph of $G$ such that each node $v \in V$ is linked with the K of its nearest neighbours. An edge $e_{ij} \in E$ is included in $E_{KNN}$ if the distance between $v_i$ and $v_j$ is among the K-th smallest distances from $v_i$ to other nodes in $V$. The RNG, which we denote as $G_{RNG}(V, E_{RNG}, W_{RNG})$, is constructed as follows: an edge $e_{ij}$ is included in $E_{RNG}$ if and only if: $d_{ij} \leq max\ \{d_{ix}, d_{xj}\}$, $\forall\ x \neq i, j$, where $d_{ij}$ is the distance between objects $i$ and $j$. We refer the reader to more information about these proximity graphs (as well as the Gabriel Graph and the Delaunay Triangulation) in references [23, 25].

Next, we explain how a proximity graph is constructed from a given dataset. Let $H_{m,n}$ be an $m \times n$ matrix where $m$ represents the number of features, $n$ indicates the number of samples and $h_{ij}$ holds the value of feature $i$ in the sample $j$. Let $C_n$ be an $n$ dimensional array where $c_j$ holds the class label of the sample $j$. Assume that a

subset of $k$ features is selected, then an $n \times n$ dimensional dissimilarity matrix $D_{n,n}$ can be calculated using only these $k$ features. For instance, $D=\{d_{ij}\}$ can be computed using the Euclidean distance metric between samples $i$ and $j$ over the selected features. As an example, consider a dataset represented by the matrix $H_{7,5}$ in Figure 1(a) and suppose the subset $\{f_2,f_4,f_6\}$ is selected. Figure 1(b) shows the dataset containing only the selected features $H_{3,5}$. Next, a distance matrix $D_{5,5}$ is calculated (Figure 1(c)) using the dataset from Figure 1(b).

| | | Samples | | | | |
|---|---|---|---|---|---|---|
| | | ***s1*** | ***s2*** | ***s3*** | ***s4*** | ***s5*** |
| Features | $f_1$ | 7.01 | 4.36 | 5.72 | 2.47 | 8.82 |
| | $f_2$ | 3.94 | 0.73 | 5.25 | 6.29 | 6.31 |
| | $f_3$ | 2.30 | 6.42 | 2.80 | 8.43 | 4.92 |
| | $f_4$ | 0.11 | 4.67 | 0.06 | 1.37 | 9.30 |
| | $f_5$ | 9.56 | 5.90 | 8.22 | 5.11 | 5.61 |
| | $f_6$ | 7.17 | 8.69 | 7.20 | 5.56 | 2.28 |
| | $f_7$ | 6.83 | 2.58 | 1.56 | 4.58 | 8.17 |
| Class($C_n$) | | 0 | 0 | 1 | 1 | 1 |

(a)

| | ***s1*** | ***s2*** | ***s3*** | ***s4*** | ***s5*** |
|---|---|---|---|---|---|
| ***f2*** | 3.94 | 0.73 | 5.25 | 6.29 | 6.31 |
| ***f4*** | 0.11 | 4.67 | 0.06 | 1.37 | 9.30 |
| ***f6*** | 7.17 | 8.69 | 7.20 | 5.56 | 2.28 |
| Classes | 0 | 0 | 1 | 1 | 1 |

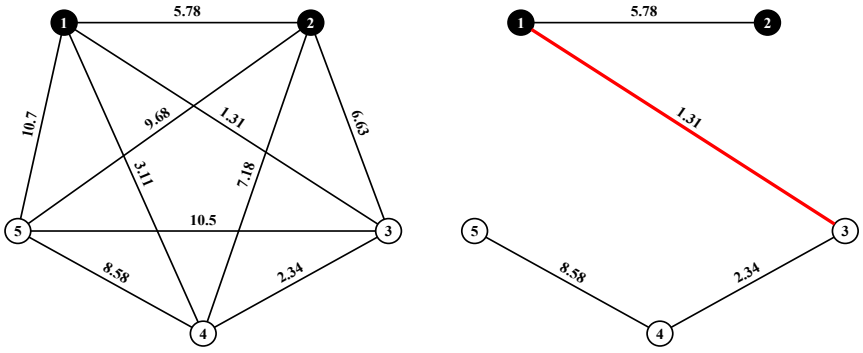
(b)

$$\boldsymbol{D_{ij}} = \begin{bmatrix} 0 & 5.78 & 1.31 & 3.11 & 10.7 \\ 5.78 & 0 & 6.63 & 7.18 & 9.68 \\ 1.31 & 6.63 & 0 & 2.34 & 10.5 \\ 3.11 & 7.18 & 2.34 & 0 & 8.58 \\ 10.7 & 9.68 & 10.5 & 8.58 & 0 \end{bmatrix}$$

(c)

**Figure 1. (a) An example of a dataset $H_{m,n}$ with m = 7 features and n = 5 samples. (b) The reduced dataset $H_{k,n}$ with the selected subset of features $\{f_2,f_4,f_6\}$. (c) A distance matrix $D_{5\times5}$ considering only the reduced dataset $H_{3,5}$.**

Given $D$, we can have a complete, undirected and weighted graph $G(V,E,W)$, where each node in $V$ represents a sample (i.e. $|V|= n$), $E$ is the set of edges between a pair of samples, and $W$ is the set of weights of each edge with $w_{ij} = d_{ij}$. Figure 2(a) shows the graph representation of the dataset $H_{3,5}$ from Figure 1(b) with the weights from the distance matrix $D_{5,5}$ from Figure 1(c). The nodes in the graph are labelled using black for class 0 and white for class 1. We now can compute a proximity graph from $G$. Figure 2(b) shows the MST constructed from the graph in Figure 2(a).




(a) (b)

**Figure 2. (a) The graph $G(V,E,W)$ represents the distance matrix $D$ where the samples from two classes are labelled using black and white nodes respectively. (b) A MST is constructed from G where only one edge connects samples from different classes (red edge).**

Note that a different subset of features will give a different dissimilarity matrix leading to a different proximity graph. The evaluation of the subset of features related to a specific proximity graph is described next.

### 2.1. Evaluating feature sets

Now, the set of features used to construct the proximity graph can be evaluated. In this work, we suggest evaluating the quality of the subset of features based on its size, which we will denote as $k$ (not to be confused

the number of neighbours considered in a K-NN graph) and/or the number of edges connecting nodes (samples) from different classes (denoted by *e*) in the built proximity graph. We proposed four evaluation criteria:

***Mine***: In this criteria, the objective is to choose the subset of features that minimises the number of edges, *e*, connecting samples from different classes in the constructed proximity graph.

$$\textbf{Mine}: \textit{Minimise (e)} \tag{1}$$

***MineMink:*** Since it is important to reduce the size of the feature set, this criterion aims to minimise the sum of the normalised *e* and normalised size of feature set (*k)*.

$$\textbf{MineMink}: \textit{Minimise}\left(\frac{e}{n-1}+\frac{k}{m}\right) \tag{2}$$

The previous two evaluation criteria do not take into account the weight of the edges. It is possible that two different solutions could produce two different proximity graphs with the same value of *e,* but the weights of the interclass edges could be different, which would indicate a different quality of features. To better discriminate samples from different classes we should also try to maximise distances between samples of different classes. For that reason, we define a new score, named proximity graph score (*PGscore*), for a given proximity graph $G_{prox}(V, E_{prox}, W_{prox})$ and use it evaluating features. Assuming we have a bipartition of $V$ in $V_A$ and $V_B$ (where $V_A$ and $V_B$ are the sets of samples that belong to class *A* and *B,* respectively*),* we can define a score for $G_{prox}$ as follows:

$$PGscore(A,B)=\frac{\frac{1}{|V_A|}\sum_{\substack{i\in V_A\\ j\in V_A}} w_{ij}-\frac{1}{|V_B|}\sum_{\substack{i\in V_B\\ j\in V_B}} w_{ij}}{1+\max_{\substack{i\in V_B\\ j\in V_B}}\{w_{ij}\}-\min_{\substack{i\in V_B\\ j\in V_B}}\{w_{ij}\}} \tag{3}$$

We can then define the other two evaluation criteria by using *PGscore* in Equation (3).

***MineMaxPGscore***: In this criterion, we favour the feature sets that bring the samples from the same class closer and keep the samples of different classes apart. In this case, the objective is to minimise the ratio between the number of edges, *e*, and the *PGscore(A,B)*

$$\textbf{MineMaxPGscore}: \textit{Minimise}\left(\frac{e}{PGscore(A,B)}\right) \tag{4}$$

***MineMinkMaxPGscore***: Like MineMink, in this criterion we consider the number of interclass edges (*e*) and the feature set size (*k*) as well as *PGScore*. In this case, the objective is to minimise the ratio between the sum of the normalised *e* and normalised *k* values and the *PGscore(A,B)*

$$\textbf{MineMinkMaxPGscore}: \textit{Minimise}\left(\frac{\frac{e}{n-1}+\frac{k}{m}}{PGscore(A,B)}\right) \tag{5}$$

The four evaluation criteria defined above are used as the fitness function in the proposed MA for exploring the feature space as explained in the next section.

## 3. Memetic algorithms for searching the feature space

Memetic Algorithms (MA), first introduced by Moscato in 1989 [26], is a population-based metaheuristic that has proven to be very efficient tackling practical optimisation problems in a variety of applications [38]. The key strategy behind MAs is to hybridise the recognised strength of a population-based method with the intensification capability of a local search to take advantage of both paradigms. Studies on MAs have confirmed that these algorithms are better capable of exploring the search space for complex problems than the traditional evolutionary algorithms (EAs) [20]. Moreover, it also has been shown that MAs can successfully

produce efficient and effective results for many NP-hard combinatorial optimisation problems, including the feature selection problem [27, 28].

We adopted one of the most commonly used MA frameworks, one in which a basic genetic algorithm (GA) provides the global exploration capability and some local search (LS) algorithm contributes achieving the exploitation of the search space. Furthermore, we also proposed an adaptation scheme for balancing the global search and the local search capabilities. The MA searches for the optimal feature sets which are evaluated by constructing proximity graphs as explained in the previous section. Therefore, we call this method **F**eature **S**election with **P**roximity graph using **M**emetic **A**lgorithm (FSPMA). In the following subsections, we explain different components of our algorithm.

### 3.1. Memetic Framework

The memetic framework used in this work is shown in Algorithm 1. As specified before, the global search capability is provided by a standard GA. In our previous work [22], we carried out extensive computational experiments to evaluate several GAs implemented for feature selection using MST. We have implemented two strategies: Steady State GA (SSGA) and Generational GA (GGA) using different combinations of operators and parameters. The performance of these algorithms was evaluated using four datasets, and one of the Generational GAs exhibited significantly better performance over the others, which is the one chosen as the global search algorithm in our memetic framework. Next, we describe each component of the algorithm.

**Population:** The population is a collection of individuals where an individual is a set of parameters that describe a single solution to the target problem, which in our case is a subset of features. We represent each individual using a binary array (***s***) of size $m$ (total number of features), where $s_j=1$ if feature $j$ ($j=1,..,m$) is selected, $s_j=0$, otherwise.

First, the population $P$, structured as a list of $p$ individuals and their corresponding fitness values, is created by ***initialisePop*()**. The method randomly initialises each of the $p$ individuals in the population. In order to increase the diversity of the population as well as to control the size of the feature sets being selected we use an *initialisation bias* inspired by [29]. First, the number of selected features in the $i$-th individual, denoted as $size(P_i)$, is determined using a linear function defined in Equation (6).

$$size(P_i) = \left\lfloor \frac{x_{max}-x_{min}}{p} * i + x_{min} \right\rfloor ; \text{where } 1 \leq i \leq p \quad (6)$$

Accordingly, the size of the selected subset of features will range between two parameters: $x_{min}$ (the minimum number of selected features) and $x_{max}$ (the maximum number of selected features). In our implementation, the value of $x_{max}$ and $x_{min}$ is set to 50% and 5% of $m$, respectively. After calculating $size(P_i)$, the $i$-th individual is created by randomly selecting $size(P_i)$ number of features in $P_i$. ***initialisePop*()** guarantees that each individual in the population is unique and assigns a fitness value, which is one of the four evaluation criteria (i.e. *Mine*, *MineMink*, *MineMaxPGscore*, *MineMinkMaxPGscore*) which have been introduced in Section 2.1.

**Algorithm 1: The proposed memetic algorithm (FSPMA).**

*p: number of individuals in the population*
*P: population of individuals and their fitness values* (*|P|=p*)
*q: number of offspring in each generation*
*Q: the set of offspring and their fitness values* (*|Q|=q*)
$P_{LS}$*: local search probability* ( $P_{LS}$ *= 0.50*)

```
01    P ← initialisePop()
02    for j ← 1 to 10%*p do
03        Ind ← randomly select an individual from P
04        Ind ← localSearch (Ind)
05        P ← updatePop(P, Ind)
06    Repeat
07        Q ←{}
08        Repeat
09            selectParents(Parent1, Parent2)
10            Offspring ← crossover(Parent1, Parent2, r_crossover)
11            Offspring ← mutation(Offspring, r_mutation)
12            fitness(Offspring)
13            if a rand(0,1) > P_LS then
14                offspring ← localSearch (offspring)
15            if offspring is not in Q then
16                Q ← add(Q,offspring)
17                i++
18        until i = q
19        P ← updatePop(P, Q)
20        updateP_LS()
21    until Stop1 or Stop2
```

**Genetic Operators:** Initially, ***selectParents****(Parent1, Parent2)* select the parents from the current population using tournament selection. The tournament strategy selects two different sets of individuals from the current population: *set1* and *set2*. In each set, five different individuals are chosen randomly such that *set1* ∩ *set2* = Ø. The fittest individual in *set1* and *set2* becomes *Parent1* and *Parent2*, respectively.

Next, the ***crossover****(Parent1, Parent2)* employs the uniform crossover operator with a crossover rate of $r_{crossover}$ = 40% to create a new individual *Offspring*. The generated *Offspring* then undergoes mutation operation, ***mutation****(Offspring)*, which is implemented using 1-flip mutation with a mutation rate of $r_{mutation}$ = 5%.

In each generation, a group of *q* offspring, denoted by *Q*, is created from potentially *q* different parent pairs through crossover and mutation. Next, the generated set of offspring, *Q* compete with the individuals in the current population, *P*, for survival. The ***updatePop****(P, Q)* selects the fittest *p* individuals from *p*+*q* candidates (i.e. the *p* individuals in the current population and the *q* new offspring created in each generation) for constituting the population of the next generation.

The iterative search continues until either of the two stopping criteria has been met. The first criterion, *Stop1,* is to terminate if a predetermined maximum number of fitness evaluations $Eval_{max}$ =10000 has elapsed; and the second criterion, *Stop2*, is to terminate if the best individual is unchanged for a maximum number of fitness evaluations $BEval_{max}$ = 1000.

### 3.2. Local Search

The other key component in the MA framework is the Local Search (LS) algorithm which performs an intense search in the neighbourhood of an individual. The purpose of a LS algorithm is to improve the current solution to the local optimum by exploiting the local neighbourhood of a solution. The success of a LS algorithm depends on many things: the aptitude of the LS algorithm to exploit the neighbourhood, the frequency and the length of the local search, the quality of the individual undergoing a local search, etc. [30].

We designed five different LS search algorithms (LS1, LS2, LS3, LS4 and LS5). The general structure of these LS algorithms is shown in Algorithm 2. The LS scheme continues the search as long as it can improve the

current solution and stops after a maximum number of failed attempts (*maxNumberAttempts* = 10) to improve the current solution. All five LS algorithms work within this framework, but employ different neighbourhood definitions. Next, we describe the neighbourhood definition employed in each LS.

**Algorithm 2: Local Search algorithm structure.**

```
localsearch(currentSolution)
01    failure ← 0
02    while failure < maxNumberAttempts do
03        failure ← failure + 1
04        newSolution ← create a new neighbouring solution from currentSolution
05        if newSolution.fitness < currentSolution.fitness then
06            currentSolution ← newSolution
07            failure ← 0
08    return(currentSolution)
```

**LS1:** A neighbouring solution (*newSolution*) is created by randomly flipping one position in the current solution (*currentSolution*).

**LS2:** First, a position (feature) x in the *currentSolution* is chosen randomly and its nearest neighbouring y, using Euclidean distance, is identified. If both x and y are the same (i.e. they both are 0 or both are 1), then create the *newSolution* by randomly flipping one of the two positions (x or y).

**LS3:** Same as in LS2, two neighbouring positions x and y in the *currentSolution* are identified. If x and y are different (i.e. one is 0 and the other is 1) then create the *newSolution* by swapping the values of x and y.

**LS4:** LS4 applies both LS2 and LS3. In other words, after selecting the neighbouring positions x and y, if both x and y are the same (i.e. they both are 0 or both are 1) then the *newSolution* is created by randomly flipping one of them. If x and y are different (i.e. one is 0 and the other is 1) then the *newSolution* is created by swapping their values.

**LS5:** The LS5 algorithm begins by identifying the nearest neighbour of each feature (using Euclidean distance). Next, for all selected features in the *currentSolution* (for all positions with value 1), the neighbour features are identified. The *newSolution* is created by selecting these neighbour features, and we call this *newSolution* reflection of *currentSolution*. If the *newSolution* is not better, then the *currentSolution* is changed using a modified version of LS4 where x is chosen randomly, and y is chosen as the reflection of x.

### 3.3. Adaptive balance between global search and local search

The synergy between the global and local searches is an essential aspect for the success of MAs. In order to design an effective searching mechanism, by taking advantage of both paradigms, we need to combine the exploration capability of GA and the exploitation capability of LS in a well-balanced manner [31]. Heavy exploitation coupled with little exploration may result in premature convergence of the search. On the other hand, shallow exploitation paired with too much exploration may slow-down the convergence of the search. Additionally, the best length and the frequency of the applicability of the local search is often problem dependent [32]. Therefore, an excellent and suitable strategy would be to balance the extent of both search paradigm adaptively taking feedback from the search itself [31].

In this work, we propose to distribute the search time between the LS and GA adaptively depending on how each search component performed in recent history – an idea originally proposed by Ishibuchi et al. [31]. We used a parameter called probability of LS ($P_{LS}$), similar to [31], to find this balance. Initially, the $P_{LS}$ = 0.5 which means we select 50% offspring to undergo the LS. During the search, we keep track of the performance of the LS. For all the offspring that undergo LS, we count how many of those were improved ($nLS_{Improved}$ ), how many of those were not improved ($nLS_{notImproved}$), the average fitness of the offspring improved by the LS ($FitLS_{Improved}$), the average fitness of the offspring not improved by the LS ($FitLS_{notImproved}$). At the end of each generation, we update the $P_{LS}$ value using the records of LS performances in the last generation. We give more

emphasis to the LS (increasing the value of $P_{LS}$) if it continuously improves the offspring, otherwise, the probability of the global search is increased (i.e. the value of $P_{LS}$ is decreased). In this way, the computational time (number of fitness evaluations) is adaptively allocated between the global search and the LS based on their relative performance. Algorithm 3 depicts the proposed ***updatePLS()*** procedure. The procedure considers only the subset of the offspring that are subject to LS, $P_{LS}$ is increased by 0.03 or decreased by 0.02 at a time and kept between the values 0.1 and 0.75.

**Algorithm 3: Adjusting the $P_{LS}$ value.**

***update*$P_{LS}$*()***
*$FitLS_{notImproved}$: the average fitness value of the solutions not improved by the LS*
*$FitLS_{Improved}$: the average fitness value of the solutions improved by the LS*
*$nLS_{notImproved}$: number of the solutions not improved by the LS*
*$nLS_{Improved}$: number of the solutions improved by the LS*
*if ($nLS_{Improved} = 0$) or ($FitLS_{Improved} > FitLS_{notImproved}$) then*
    *$P_{LS} = max(0.1, P_{LS} - 0.02)$*
*else if ($nLS_{notImproved} = 0$) or ($FitLS_{Improved} \leq FitLS_{notImproved}$) then*
    *$P_{LS} = min(0.75, P_{LS} + 0.03)$*
*return ($P_{LS}$)*

## 4. Computational results

This section presents the computational experiments we performed to establish the effectiveness of the proposed method. The experiments performed over multiple datasets exhibit the superiority of the memetic algorithm over the genetic algorithm. The suitability of different proximity graphs for the proposed method has also been investigated. Furthermore, the contribution of different components in the proposed memetic algorithm such as the local search and the adaptive mechanism are also examined.

### 4.1. Datasets

We used four well-known binary classification datasets that are available in the public domain for future reproducibility and comparability with past studies. The characteristics of these four datasets used are summarised in (Table 1) and described next.

**Shakespearean era plays and poems dataset:** This dataset contains 256 literary works from the Shakespearean era. The two binary classes in this dataset are plays (202) and poems (54). The "frequency of use" of 220 functional words have been extracted from a cohort of 66907 words previously analysed by Arefin et al. in [4]. The observed frequencies in the different works of these 220 functional words are used as features. The goal is to identify 'a subset of functional words' that can group the works into the two classes: plays and poems.

**Alzheimer's disease datasets:** These datasets were introduced by Ray et al. [33] and they pose several classification problems. We work with the two first datasets in which the objective is to identify a subset of relative abundances of 120 proteins that could distinguish clinical symptoms of Alzheimer's Disease (AD) five years in advance in comparison against the Non-Demented Control (NDC) group [34]. The training dataset contains the relative abundances (z-scores to be used as features) measured on 83 people belonging to two classes: 43 AD and 40 NDC samples. The test dataset contains 120 protein measures of 81 patients classified into two classes – 42 AD samples and 39 NDC samples. We note that the original test dataset also contains 11 samples labelled as 'Other Dementia' (OD) samples, but these are excluded from our analysis to keep it a binary classification problem.

**Table 1 Datasets used for evaluating the FSPMA method**

| Dataset Name | | Features | Samples | Classes | |
|---|---|---|---|---|---|
| Shakespearean era plays and poems [35] | | 220 | 256 | Play | 202 |
| | | | | Poem | 54 |
| Alzheimer's disease [33] | *training* | 120 | 83 | AD | 43 |
| | | | | NDC | 40 |
| | *test* | 120 | 81 | AD | 42 |
| | | | | NDC | 39 |
| Colon dataset [36] | | 2000 | 62 | Tumour tissue | 40 |
| | | | | Normal tissue | 22 |
| Embryonal tumour [37] | | 7129 | 59 | Survivors children | 20 |
| | | | | Failures children | 39 |

**Colon dataset:** The colon dataset [36] consists of the expression of 2000 genes (originally 6000 gene expression levels were measured; 2000 were selected based on the confidence in the measurement) with highest minimal intensity across the 62 samples of colon epithelial cells collected from colon-cancer patients: 22 Normal tissues and 40 Tumour tissues. The goal is to find a subset of genes that can classify Normal tissues and Tumour tissues For more information about the Colon dataset please refer to [36] and it is available on the public domain from the http://genomics-pubs.princeton.edu/oncology/ website.

**Embryonal tumour dataset:** The data originates from the study of 59 human tissue samples obtained from children with different brain tumours with medulloblastomas [37] associated with the Committee for Clinical Investigation of Boston Children's Hospital. The Dataset C from this study is used as an input instance in this work and is downloaded from http://portals.broadinstitute.org/cgi-bin/cancer/datasets.cgi website. These samples were labelled into two classes: "survivors" (20) and "failures" (39) (labels taken verbatim from [37]). From the frozen tumour samples RNA was isolated and hybridised to an array containing 7129 probes. The goal is to select a subset of those 7129 probes that can define the best separation of "survivors" and "failures".

### 4.2. Experimental setup

As mentioned in Section 3.1, the setup of the best performing GA from [22] was selected (it is called GGA4 in [32]) as the global search algorithm in our experiments. Therefore, our proposed memetic algorithm (FSPMA) works with a population of 100 individuals, uses tournament strategy for parent selection and applies uniform crossover and 1-flip mutation for offspring generation as described in Section 3.1. All the algorithms were coded in Python 2.7 and executed under the Unix operating system in a machine with Dual Xeon 2.67 GHz, eight cores and 32 GB RAM.

### 4.3. Performance of the LS algorithms

Before embedding the LS algorithms in the memetic framework, we evaluated the standalone performance of the designed LS algorithms. We assessed the performance of each LS algorithm, regarding fitness score and time requirement, in improving the quality of a random solution. The study uses all the datasets described in Section 4.1, but only the *Mine* fitness criterion. For each dataset, 50 random solutions are generated and each LS strategy is applied to them. Table 2 shows the results of this experiment regarding average fitness value (Fit) of these 50 solutions before and after applying different LS strategies and the average time in seconds taken by each tested LS strategy. The best performing strategy (in terms of average fitness score) is highlighted for each dataset.

**Table 2: Performance of LS schemes in *Mine* fitness criterion on different datasets.**

| | Shakespeare | | Alzheimer's disease | | Colon cancer | | Embryonal tumour | |
|---|---|---|---|---|---|---|---|---|
| **Local search** | ***Fit*** | ***Time*** | ***Fit*** | ***Time*** | ***Fit*** | ***Time*** | ***Fit*** | ***Time*** |
| Initial solution | 13.84 | | 20.30 | | 14.20 | | 24.92 | |
| *LS1* | 12.84 | 1.63 | **17.96** | 0.18 | 13.98 | 0.41 | 24.86 | 1.63 |
| *LS2* | 13.08 | 0.69 | 18.64 | 0.06 | 14.10 | 0.25 | 24.88 | 0.90 |
| *LS3* | 12.96 | 0.51 | 18.66 | 0.05 | 14.08 | 0.17 | 24.82 | 0.53 |
| *LS4* | **12.52** | 1.29 | 18.30 | 0.13 | 14.04 | 0.39 | 24.80 | 1.40 |
| *LS5* | 13.60 | 0.93 | 18.16 | 0.10 | **13.00** | 0.25 | **22.66** | 0.40 |

According to the results in Table 2, all the proposed LS strategies have the ability to improve the quality of the solution, but it is not conclusive which LS has the best performance in regard to quality of solution and required time. Therefore, we incorporated each of the proposed LS strategies with the global search algorithm and designed five MAs .

### 4.4. Benefit of adaptive balancing mechanism

All the MAs studied in this work were designed by embedding one of the five LS strategies in the MA framework of Algorithm 1. We denoted the MAs as MA1 (LS1), MA2 (LS2), MA3 (LS3), MA4 (LS4) and MA5 (LS5) where the LS strategy incorporated in them are shown in the parentheses.

Next, we performed some specific experiments to show that the proposed adaptive scheme is useful to determine a right balance between local and global searches in the MAs and improve their search performance thereby. As described earlier, the balance between local and global searching is controlled by the $P_{LS}$ parameter in Algorithm 1. Therefore, we experimented by creating two variants of each MA one of which keeps $P_{LS}$=50% fixed (denoted by MA1-F, MA2-F, MA3-F, MA4-F and MA5-F) and the other changes it adaptively according to Algorithm 3 (denoted by MA1, MA2, MA3, MA4 and MA5). In this experiment, we use only the AD training dataset and the *Mine* fitness function. Each MA is repeated 50 times. Each row of Table 3 shows the average of 50 executions in terms of the fitness value (*Fit*), number of selected features (*k*), running time in seconds (*Time*), number of fitness evaluations in the global search (*EvGS*), number of fitness evaluations in the local search (*EvLS*), time spent in the local search (*LSTime*) and the average $P_{LS}$ values obtained by the *update*$P_{LS}$*()* method.

**Table 3: Comparison between MAs with fixed $P_{LS}$ and adaptive $P_{LS}$ in the Alzheimer's disease dataset with *Mine* fitness criterion.**

| | ***Fit*** | ***k*** | ***Time*** | ***EvGS*** | ***EvLS*** | ***LSTime*** | ***Average*$P_{LS}$** |
|---|---|---|---|---|---|---|---|
| MA1-F | 4.60 | 40.54 | 325 | 4275 | 23556 | 260 | |
| MA1 | 4.60 | 41.36 | 264 | 4062 | 18740 | 207 | 0.41 |
| MA2-F | 4.62 | 40.58 | 190 | 3745 | 11344 | 125 | |
| MA2 | 4.44 | 41.26 | 236 | 3927 | 14889 | 164 | 0.62 |
| MA3-F | 4.41 | 40.76 | 173 | 4338 | 9411 | 102 | |
| **MA3** | **4.34** | **40.12** | **241** | **4317** | **14830** | **160** | **0.70** |
| MA4-F | 4.62 | 39.72 | 322 | 4048 | 24487 | 257 | |
| MA4 | 4.60 | 41.96 | 414 | 4041 | 32795 | 344 | 0.64 |
| MA5-F | 4.61 | 40.66 | 530 | 3940 | 41770 | 424 | |
| MA5 | 4.64 | 41.28 | 117 | 4498 | 5720 | 58 | 0.11 |

The results in Table 3 show that the adaptive $P_{LS}$ was helpful in achieving a better fitness score in MA2, MA3 and MA4. In these three MAs the local search schemes were successful in improving the fitness scores of the individuals; therefore, the average $P_{LS}$ score was higher than 0.50. On the other hand in MA1 and MA5 the adaptive strategy was performing similar and worse to fixed $P_{LS}$, respectively using a much smaller number of fitness evaluations in local search. Looking at the average $P_{LS}$ score we can see that the local search scheme was not useful to improve individual's fitness score and therefore the average $P_{LS}$ was less than 0.5. In terms of computational time, the adaptive LS strategy was more efficient than the fixed LS strategy in the number of fitness evaluation used in local search. Based on these observations, we argue that the adaptive $P_{LS}$ was useful to balance between global and local searches and helped to explore the search space effectively to find a better solution. Therefore, we adopted the adaptive balance in the MAs. Next, we present the extensive computational tests of the MAs using all datasets.

### 4.5. Performance of the memetic algorithms

After we have ascertained the benefit of the local search schemes and the benefit of the adaptive balancing strategy, we compared the five MAs in all four datasets, using the four fitness criteria. We run each MA 50 times for each fitness criterion on each dataset using the same experimental setup. Tables 4–7 present the results obtained for each dataset, where each row shows the average of 50 runs. The performance of the MAs was compared with their common parent genetic algorithm GA. We also run GA allowing to use the same

number of fitness evaluations used by the best-performing MA (MA3), which is shown in the tables by GA+. The aim is to show that the hybridisation between the LS and the global search can produce better results than GA even with the same total number of fitness evaluations. Each table shows the average of best fitness values (Fit), the average number of inter-class edges (e), the size of selected subset of features (k), the average execution time in seconds (Time), the average total number of fitness evaluations for global search (EvGen), the average number of fitness evaluations in the LS (EvLS) and the average total local search running time (LSTime). The tables also show the *p*-value result of a Wilcoxon test between the best-performing MA (MA3) and each method.

For the Shakespeare dataset (Table 4), MA3 achieved the best or equal best average fitness scores when compared to other algorithms. Almost half of the Wilcoxon *p*-values (11 out of 24) were significant (≤0.05). MA2 and MA3 gave the same results using *Mine* criterion. In addition, MA3 did not exhibit notably different behaviour using the *MineMink* fitness criterion. In the case of the AD training dataset (Table 5), the MA3 performed significantly better than the other methods in most cases (83% of Wilcoxon *p*-values were less than or equal to 0.05). In addition, MA3 required less time in most cases. For the colon cancer dataset (Table 6), MA3 outperformed other methods significantly as 92% of Wilcoxon p-values indicated that the MA3 results were significantly better than the other methods. Another advantage of MA3 is that it achieved the best results in less time than the other MAs. Finally, the results obtained when we used the embryonal tumour dataset (Table 7) show 71% of Wilcoxon *p*-values were less than or equal to 0.05, indicating a significantly better performance by MA3. Using *Mine*, MA2 obtained better results than MA3, but the difference was not statistically significant ($p = 0.12$) and MA2 required more computational time. Using *MineMink*, MA3 and MA5 achieved the same result, but again, MA3 required less computational time. It is also worth mentioning that MA3 achieved significantly better results in most of the cases when compared with GA+ in all four datasets which establish the superiority of memetic search over pure genetic search. Based on the results from Tables 4–7, it can be concluded that MA3 outperformed the other algorithms where MA3 was significantly better ($p \leq 0.05$) than the other methods in 77% of Wilcoxon tests.

**Table 4: Performance of different algorithms on Shakespeare dataset using different fitness criteria**

| | | *Fit* | *k* | *e* | *Time* | *EvGen* | *EvLS* | *LSTime* | *p-value* |
|---|---|---|---|---|---|---|---|---|---|
| *Mine* | MA1 | 1.44 | 60.84 | 1.44 | 2277 | 2477 | 17403 | 1904 | 0.06 |
| | **MA2** | **1.22** | **61.16** | **1.22** | **1497** | **2441** | **9192** | **1054** | **0.43** |
| | **MA3** | **1.22** | **58.00** | **1.22** | **1256** | **2531** | **6962** | **800** | **--** |
| | MA4 | 1.68 | 61.02 | 1.68 | 2303 | 2334 | 17851 | 1908 | 0.05 |
| | MA5 | 1.58 | 59.86 | 1.58 | 685 | 2496 | 3887 | 365 | 0.04 |
| | GA | 1.70 | 59.00 | 1.70 | 541 | 2560 | -- | -- | 0.02 |
| | GA4+ | 1.50 | 32.19 | 1.50 | 1700 | 9500 | -- | -- | 0.01 |
| *MineMink* | MA1 | 0.08 | 13.78 | 3.48 | 1844 | 9968 | 12371 | 774 | 0.08 |
| | **MA2** | **0.07** | **13.34** | **3.62** | **1880** | **10000** | **10648** | **673** | **0.21** |
| | **MA3** | **0.07** | **12.96** | **3.62** | **1759** | **10000** | **12403** | **740** | **--** |
| | MA4 | 0.08 | 13.66 | 3.72 | 1834 | 10000 | 19055 | 1140 | 0.06 |
| | MA5 | 0.08 | 14.42 | 3.88 | 1745 | 10000 | 14357 | 1090 | 0.09 |
| | GA | 0.08 | 15.28 | 4.26 | 713 | 10000 | -- | -- | 0.07 |
| | **GA+** | **0.07** | **11.92** | **3.98** | **1770** | **22400** | **--** | **--** | **0.34** |
| *MineMaxPGscore* | MA1 | 6.07 | 112.42 | 2.80 | 2358 | 2447 | 19577 | 2030 | 0.03 |
| | MA2 | 5.39 | 113.72 | 2.46 | 1631 | 2454 | 10072 | 1177 | 0.07 |
| | **MA3** | **5.35** | **115.68** | **3.00** | **1638** | **2464** | **9881** | **1165** | **--** |
| | MA4 | 5.61 | 114.26 | 2.60 | 2662 | 2523 | 22156 | 2273 | 0.01 |
| | MA5 | 6.08 | 111.00 | 3.00 | 983 | 2432 | 6881 | 648 | 0.00 |
| | GA | 6.11 | 111.14 | 2.82 | 515 | 2475 | -- | -- | 0.00 |
| | GA+ | 5.44 | 116.40 | 3.70 | 1820 | 12300 | -- | -- | 0.04 |
| *MineMinkMaxPGscore* | MA1 | 0.30 | 29.30 | 12.78 | 2574 | 10000 | 18502 | 1627 | 0.08 |
| | MA2 | 0.30 | 28.32 | 12.88 | 2460 | 10000 | 16215 | 1423 | 0.12 |
| | **MA3** | **0.29** | **27.34** | **12.82** | **2916** | **9913** | **18625** | **1549** | **--** |
| | MA4 | 0.30 | 28.20 | 13.18 | 3589 | 9968 | 33912 | 2620 | 0.05 |
| | MA5 | 0.30 | 28.56 | 13.40 | 2546 | 10000 | 23154 | 1686 | 0.06 |
| | GA | 0.34 | 33.24 | 13.76 | 761 | 10000 | -- | -- | 0.00 |
| | GA+ | 0.30 | 34.28 | 12.40 | 2922 | 28500 | -- | -- | 0.08 |

**Table 5: Performance of different algorithms on Alzheimer dataset using different fitness criteria**

| | | *Fit* | *k* | *e* | *Time* | *EvGen* | *EvLS* | *LSTime* | *p-value* |
|---|---|---|---|---|---|---|---|---|---|
| *Mine* | MA1 | 4.60 | 41.36 | 4.60 | 264 | 4062 | 18740 | 207 | 0.02 |
| | MA2 | 4.44 | 41.26 | 4.44 | 236 | 3927 | 14889 | 164 | 0.05 |
| | **MA3** | **4.34** | **40.12** | **4.34** | **240** | **4317** | **14830** | **160** | **--** |
| | MA4 | 4.60 | 41.96 | 4.60 | 414 | 4041 | 32795 | 344 | 0.02 |
| | MA5 | 4.64 | 41.28 | 4.64 | 117 | 4498 | 5720 | 58 | 0.01 |
| | GA | 4.70 | 41.00 | 4.70 | 74 | 4160 | -- | -- | 0.02 |
| | GA+ | 4.43 | 30.72 | 4.43 | 285 | 19100 | -- | -- | 0.05 |
| *MineMink* | MA1 | 0.22 | 17.86 | 6.20 | 338 | 9948 | 16781 | 205 | 0.06 |
| | MA2 | 0.22 | 17.10 | 6.70 | 507 | 9924 | 37006 | 359 | 0.07 |
| | **MA3** | **0.20** | **15.74** | **6.24** | **419** | **9903** | **22650** | **238** | **--** |
| | MA4 | 0.22 | 16.68 | 6.58 | 743 | 9930 | 68021 | 602 | 0.07 |
| | MA5 | 0.22 | 17.34 | 6.31 | 704 | 9984 | 43126 | 503 | 0.12 |
| | GA | 0.23 | 17.04 | 7.68 | 49 | 10000 | -- | -- | 0.04 |
| | GA+ | 0.22 | 16.68 | 6.18 | 398 | 32500 | -- | -- | 0.05 |
| *MineMaxPGscore* | MA1 | 16.71 | 47.86 | 8.98 | 577 | 9552 | 27515 | 411 | 0.00 |
| | MA2 | 16.36 | 48.58 | 8.94 | 812 | 9648 | 38131 | 574 | 0.02 |
| | **MA3** | **15.65** | **44.32** | **7.96** | **805** | **9675** | **38444** | **562** | **--** |
| | MA4 | 16.38 | 47.82 | 8.86 | 1303 | 9348 | 74820 | 1087 | 0.03 |
| | MA5 | 16.31 | 44.86 | 9.10 | 429 | 9520 | 21867 | 273 | 0.04 |
| | GA | 18.70 | 46.04 | 9.62 | 134 | 10000 | -- | -- | 0.00 |
| | GA+ | 16.36 | 41.70 | 8.72 | 809 | 48100 | -- | -- | 0.00 |
| *MineMinkMaxPGscore* | MA1 | 0.46 | 24.72 | 19.68 | 418 | 10000 | 22555 | 281 | 0.00 |
| | MA2 | 0.46 | 25.60 | 20.94 | 635 | 10000 | 36034 | 448 | 0.00 |
| | **MA3** | **0.42** | **21.44** | **18.96** | **557** | **10000** | **29265** | **350** | **--** |
| | MA4 | 0.46 | 25.14 | 20.86 | 1076 | 10000 | 71801 | 886 | 0.00 |
| | MA5 | 0.47 | 24.94 | 20.94 | 419 | 10000 | 24602 | 282 | 0.00 |
| | GA | 0.51 | 25.48 | 21.08 | 109 | 10000 | -- | -- | 0.00 |
| | GA+ | 0.47 | 21.72 | 19.06 | 664 | 39220 | -- | -- | 0.00 |

**Table 6: Performance of different algorithms on colon cancer dataset using different fitness criteria**

| | | *Fit* | *k* | *E* | *Time* | *EvGen* | *EvLS* | *LSTime* | *p-value* |
|---|---|---|---|---|---|---|---|---|---|
| *Mine* | MA1 | 5.22 | 389.32 | 5.40 | 360 | 1812 | 7410 | 275 | 0.04 |
| | **MA2** | **5.04** | **386.12** | **5.04** | **234** | **1780** | **3845** | **145** | **0.33** |
| | **MA3** | **5.04** | **408.62** | **5.04** | **135** | **1747** | **1131** | **51** | **--** |
| | MA4 | 5.28 | 381.68 | 5.28 | 344 | 1856 | 7131 | 258 | 0.04 |
| | MA5 | 5.40 | 342.70 | 5.50 | 187 | 1860 | 4669 | 112 | 0.00 |
| | GA | 5.40 | 403.00 | 5.40 | 67 | 1840 | -- | -- | 0.02 |
| | GA+ | 5.24 | 278.30 | 5.24 | 155 | 5600 | -- | -- | 0.04 |
| *MineMink* | MA1 | 0.1654 | 172.50 | 4.76 | 236 | 1904 | 9060 | 188 | 0.01 |
| | MA2 | 0.1602 | 173.62 | 4.32 | 225 | 2008 | 8987 | 170 | 0.05 |
| | **MA3** | **0.1575** | **166.82** | **4.32** | **83** | **2055** | **706** | **16** | **--** |
| | MA4 | 0.1672 | 173.60 | 4.84 | 282 | 2071 | 11921 | 227 | 0.00 |
| | MA5 | 0.1614 | 141.30 | 5.48 | 400 | 1933 | 22423 | 356 | 0.04 |
| | GA | 0.1664 | 167.32 | 5.08 | 74 | 1978 | -- | -- | 0.01 |
| | GA+ | 0.1584 | 155.10 | 4.88 | 86 | 3000 | -- | -- | 0.05 |
| *MineMaxPGscore* | MA1 | 30.38 | 613.42 | 7.24 | 1696 | 1983 | 25576 | 1518 | 0.00 |
| | MA2 | 30.72 | 597.76 | 7.54 | 828 | 1972 | 10314 | 632 | 0.00 |
| | **MA3** | **28.74** | **557.20** | **7.50** | **550** | **2051** | **6624** | **370** | **--** |
| | MA4 | 29.13 | 599.18 | 7.00 | 1182 | 2029 | 18624 | 1013 | 0.05 |
| | MA5 | 32.04 | 418.92 | 8.44 | 508 | 2256 | 15431 | 405 | 0.00 |
| | GA | 32.07 | 587.58 | 7.58 | 227 | 2073 | -- | -- | 0.00 |
| | GA+ | 29.00 | 643.90 | 7.54 | 667 | 8640 | -- | -- | 0.17 |
| *MineMinkMaxPGscore* | MA1 | 0.95 | 282.60 | 9.20 | 535 | 3334 | 14135 | 420 | 0.00 |
| | MA2 | 0.96 | 283.10 | 9.52 | 519 | 3265 | 12337 | 376 | 0.00 |
| | **MA3** | **0.84** | **270.10** | **9.20** | **320** | **3523** | **5107** | **160** | **--** |
| | MA4 | 0.97 | 281.78 | 9.50 | 719 | 3303 | 18836 | 575 | 0.00 |
| | MA5 | 0.90 | 217.02 | 10.78 | 710 | 3549 | 29819 | 609 | 0.02 |
| | GA | 0.96 | 272.80 | 10.10 | 161 | 3007 | -- | -- | 0.00 |
| | GA+ | 0.88 | 231.80 | 9.92 | 362 | 8600 | -- | -- | 0.03 |

**Table 7: Performance of different algorithms on embryo dataset using different fitness criteria**

| | | *Fit* | *k* | *E* | *Time* | *EvGen* | *EvLS* | *LSTime* | *p-value* |
|---|---|---|---|---|---|---|---|---|---|
| *Mine* | MA1 | 8.80 | 1216.38 | 8.80 | 1144 | 2028 | 6002 | 827 | 0.05 |
| | **MA2** | **8.52** | **1116.28** | **8.52** | **802** | **2069** | **3586** | **441** | **0.12** |
| | MA3 | 8.55 | 1251.10 | 8.50 | 564 | 2024 | 995 | 189 | -- |
| | MA4 | 9.45 | 1134.08 | 9.45 | 985 | 2000 | 5399 | 701 | 0.02 |
| | MA5 | 9.08 | 702.94 | 9.08 | 391 | 2233 | 5251 | 200 | 0.00 |
| | GA | 9.86 | 1268.00 | 9.86 | 263 | 2000 | -- | -- | 0.00 |
| | GA+ | 9.72 | 1222.22 | 9.72 | 560 | 3000 | -- | -- | 0.00 |
| *MineMink* | MA1 | 0.22 | 589.42 | 7.80 | 590 | 2184 | 8253 | 468 | 0.03 |
| | MA2 | 0.23 | 609.54 | 8.38 | 615 | 2280 | 6203 | 430 | 0.00 |
| | **MA3** | **0.20** | **599.80** | **8.70** | **233** | **2097** | **572** | **57** | **--** |
| | MA4 | 0.23 | 599.22 | 8.24 | 743 | 2198 | 8037 | 584 | 0.00 |
| | **MA5** | **0.20** | **345.14** | **8.44** | **801** | **2364** | **19974** | **682** | **0.37** |
| | GA | 0.24 | 628.96 | 8.76 | 254 | 2183 | -- | -- | 0.00 |
| | GA+ | 0.24 | 618.60 | 8.76 | 285 | 2650 | -- | -- | 0.00 |
| *MineMaxPGscore* | MA1 | 35.50 | 2251.73 | 15.41 | 998 | 2224 | 29353 | 8919 | 0.04 |
| | MA2 | 35.90 | 2197.02 | 15.36 | 477 | 2284 | 11246 | 3573 | 0.05 |
| | **MA3** | **33.43** | **2096.40** | **16.02** | **339** | **2141** | **6463** | **2190** | **--** |
| | MA4 | 34.62 | 2117.86 | 14.92 | 688 | 2282 | 20374 | 5855 | 0.05 |
| | MA5 | 35.72 | 2176.86 | 15.94 | 137 | 2064 | 6678 | 492 | 0.04 |
| | GA | 36.15 | 2151.56 | 15.44 | 146 | 2327 | -- | -- | 0.00 |
| | GA+ | 33.77 | 2252.95 | 14.94 | 350 | 8600 | -- | -- | 0.06 |
| *MineMinkMaxPGscore* | MA1 | 0.84 | 1049.94 | 17.46 | 2031 | 2886 | 13467 | 1607 | 0.42 |
| | MA2 | 0.90 | 1099.08 | 17.90 | 1649 | 2867 | 9099 | 1137 | 0.01 |
| | **MA3** | **0.83** | **1120.98** | **17.38** | **661** | **2684** | **1310** | **219** | **--** |
| | MA4 | 0.88 | 1093.58 | 17.46 | 2210 | 3092 | 14009 | 1704 | 0.03 |
| | MA5 | 0.86 | 824.16 | 17.20 | 539 | 3001 | 10552 | 318 | 0.06 |
| | GA | 0.88 | 1086.08 | 16.74 | 670 | 2963 | -- | -- | 0.09 |
| | GA+ | 0.85 | 1072.60 | 16.58 | 811 | 4000 | -- | -- | 0.08 |

### 4.6. Classification Results

In this section, we assess the practical use of the set of features obtained by FSPMA (using MA3) for classification. In order to evaluate the classification performance of the different selected feature sets, we adopted the methodology used in [17]. We used a total of 49 machine learning algorithms from the well-known WEKA software package [50] (version 3.6.4). The list of classifiers used is shown in Table 8 along with their respective types as categorised in WEKA.

**Table 8: List of classifiers categorised by their types from WEKA [38] used in this work**

| Type | Classifier |
|---|---|
| Bayes | BayesNet, NaiveBayes, NaiveBayesUpdatable |
| Function | Logistic, SimpleLogistic, RBFNetwork, SMO, SPegasos, VotedPerceptron |
| Lazy | IB1, Kstar, LWL |
| Rules | ConjunctiveRule, DecisionTable, Jrib, NNge, OneR, Part, Ridor |
| Tree | ADTree, BFTree, FT, LADTree, LMT, DecisionStump, J48, J48graft, RepTree, NBTree, Random_Forest, RandomTree |
| Mesc | HyperPipes, VFI |
| Meta | AdaBoost, AttributeSelectedClassifier, Bagging, ClassificationViaRegression, Dagging, Decorate, END, FilteredClassifier, LogitBoost, MultiBoostAB, MultiClassClassifier, OrdinalClass, RandomCommittee, RandomSub_Space, RotationForest, ThresholdSelector |

For each of the MA3 results from Tables 4–7, the subset of features (out of 50 solutions) with the best fitness value (Fit) was selected for evaluating the classification performance of each algorithm using 10-fold cross validation. We then calculated the average accuracy (ACC) and the average Mathew's correlation coefficient (MCC) of the 49 WEKA classifiers for comparison.

As described in Section 3.1, the solutions in the initial population are created by setting the parameters as $x_{max} = m*50\%$ and $x_{min} = m*5\%$ (Equation (6)), which has a direct influence on the number of features selected by the algorithms. This effect of initialisation bias can be noticed by the number of selected features in Table 4-7 by different MAs. Since the number of selected feature also influences the classification accuracy, we re-ran the MA3 algorithm with a different initial bias, specifically with $x_{max} = m*10\%$ and $x_{min} = m*1\%$, in order to obtain a smaller subset of features. To differentiate between these two different initialization biases we use the term *wide initialisation* for $x_{max} = m*50\%$ and $x_{min} = m*5\%$ and *narrow initialisation* for $x_{max} = m*10\%$ and $x_{min} = m*1\%$. We compared the effect of these two initialization biases in Table 9 in terms of the fitness value (Fit), the size of the subset of features (k), the number of edges that connect samples between different classes in the MST (e), the required time in seconds (Time), total number of required fitness evaluations (Ev), average MCC and average ACC for different fitness criteria and datasets. For each dataset and fitness function, the best ACC and MCC results are highlighted.

The results presented in Table 9 show that initialisation using the narrow initialisation not only reduced the size of the selected subset of features, but also had a positive effect by improving classification performance in terms of MCC and ACC. No single fitness function was found to be consistently the best for all datasets. However, *MineMink* achieved the best performance in two datasets and each of *Mine* and *MineMinkMaxPGscore* achieved the best performance in one dataset.

**Table 9: Classification performance of MA3 algorithm using different initialization bias for different dataset and fitness criteria.**

| Dataset | | Initialisation | *Fit* | *e* | *k* | *Time* | *Ev* | MCC | ACC |
|---|---|---|---|---|---|---|---|---|---|
| *Shakespeare Dataset* | ***Mine*** | Wide | 1 | 1 | 36 | 1108 | 8874 | 0.889 | 0.963 |
| | | **Narrow** | **1** | **1** | **17** | **1403** | **11080** | **0.903** | **0.967** |
| | *MineMink* | Wide | 0.08 | 1 | 18 | 2083 | 24021 | 0.889 | 0.963 |
| | | Narrow | 0.07 | 1 | 17 | 2126 | 24219 | 0.900 | 0.966 |
| | *MineMaxPGscore* | Wide | 2.29 | 1 | 35 | 1813 | 13066 | 0.882 | 0.961 |
| | | Narrow | 1.89 | 1 | 19 | 1615 | 14112 | 0.895 | 0.965 |
| | *MineMinkMaxPGscore* | Wide | 0.22 | 4 | 73 | 2741 | 24948 | 0.885 | 0.962 |
| | | Narrow | 0.16 | 3 | 13 | 2667 | 24006 | 0.888 | 0.962 |
| *Alzheimer' s dataset* | *Mine* | Wide | 3 | 3 | 38 | 213 | 22017 | 0.766 | 0.882 |
| | | Narrow | 2 | 2 | 12` | 209 | 19113 | 0.786 | 0.892 |
| | ***MineMink*** | Wide | 0.16 | 3 | 15 | 378 | 30819 | 0.766 | 0.883 |
| | | **Narrow** | **0.1** | **3** | **8** | **319** | **28864** | **0.790** | **0.895** |
| | *MineMaxPGscore* | Wide | 12.34 | 3 | 34 | 775 | 40884 | 0.779 | 0.889 |
| | | Narrow | 10.09 | 2 | 15 | 789 | 44313 | 0.784 | 0.891 |
| | *MineMinkMaxPGscore* | Wide | 0.63 | 3 | 14 | 488 | 33498 | 0.774 | 0.887 |
| | | Narrow | 0.41 | 3 | 12 | 507 | 35204 | 0.785 | 0.892 |
| *Colon Dataset* | *Mine* | Wide | 3 | 3 | 231 | 172 | 3092 | 0.606 | 0.821 |
| | | Narrow | 2 | 2 | 66 | 136 | 2832 | 0.622 | 0.827 |
| | *MineMink* | Wide | 0.12 | 4 | 109 | 77 | 3013 | 0.601 | 0.817 |
| | | Narrow | 0.06 | 2 | 69 | 70 | 3114 | 0.630 | 0.831 |
| | *MineMaxPGscore* | Wide | 15.64 | 5 | 402 | 566 | 9091 | 0.599 | 0.818 |
| | | Narrow | 6.63 | 4 | 71 | 487 | 8917 | 0.629 | 0.830 |
| | ***MineMinkMaxPGscore*** | Wide | 0.43 | 12 | 178 | 288 | 7891 | 0.610 | 0.822 |
| | | **Narrow** | **0.11** | **4** | **50** | **254** | **7402** | **0.653** | **0.842** |
| *Embryonal Dataset* | *Mine* | Wide | 7 | 7 | 654 | 747 | 3314 | 0.257 | 0.683 |
| | | Narrow | 2 | 2 | 68 | 289 | 2988 | 0.456 | 0.756 |
| | ***MineMink*** | Wide | 0.41 | 4 | 436 | 205 | 2413 | 0.366 | 0.720 |
| | | **Narrow** | **0.08** | **2** | **70** | **140** | **2841** | **0.509** | **0.782** |
| | *MineMaxPGscore* | Wide | 23.42 | 11 | 594 | 986 | 8241 | 0.334 | 0.708 |
| | | Narrow | 2.16 | 6 | 73 | 310 | 8452 | 0.465 | 0.764 |
| | *MineMinkMaxPGscore* | Wide | 0.43 | 10 | 598 | 882 | 5082 | 0.385 | 0.730 |
| | | Narrow | 0.16 | 5 | 59 | 343 | 4894 | 0.465 | 0.764 |

The performance of the 49 classifiers in WEKA varies widely. Therefore, next, we calculated the average MCC and ACC for the top five best performing classifiers. Table 10 depicts the results, where each row contains the average from the five top performing classifiers (out of 49 classifiers listed in Table 8) for the feature sets generated by FSPMA with narrow initialization.

**Table 10: The average MCC and ACC for the pool of classifiers created by the best five performing classifiers in different fitness functions.**

| | **Dataset** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Shakespeare** | | **Alzheimer's disease** | | **Colon cancer** | | **Embryonal tumour** | |
| | ACC | MCC | ACC | MCC | ACC | MCC | ACC | MCC |
| *All* | 0.971 | 0.914 | 0.864 | 0.728 | 0.800 | 0.552 | 0.631 | 0.141 |
| *Mine* | 0.979 | 0.937 | 0.925 | 0.851 | 0.846 | 0.663 | 0.798 | 0.539 |
| *MineMink* | **0.983** | **0.950** | **0.928** | **0.857** | **0.865** | **0.703** | **0.831** | **0.615** |
| *MineMaxPGscore* | 0.979 | 0.938 | 0.926 | 0.852 | 0.845 | 0.662 | 0.790 | 0.523 |
| *MineMinkMaxPGscore* | 0.975 | 0.925 | 0.925 | 0.851 | 0.859 | 0.692 | 0.787 | 0.518 |

The Table 10 evidently shows that the *MineMink* fitness function is always superior. The results of Table 9 showed that *MineMink* fitness function achieved the best average MCC and ACC results in two datasets (Alzheimer's and Embryo datasets). However, after selecting the five best-performing classifiers, it attained the best results in all datasets.

Figure 3 illustrates the MST generated by FSPMA using the *MineMink* fitness function in each dataset. We can see that in case of Shakespeare dataset, an ideal solution (single interclass edge) has been achieved where the 202 samples in the 'plays' class represented as black nodes are perfectly separated from the 54 samples in the 'poems' class represented as white nodes. The constructed MSTs have achieved the quasi-ideal solutions

(few inter-class edges) in the other datasets. From these observations, we can see that FSPMA with *MineMink* fitness criteria generates the highest quality feature sets that can separate the samples into different classes very well.

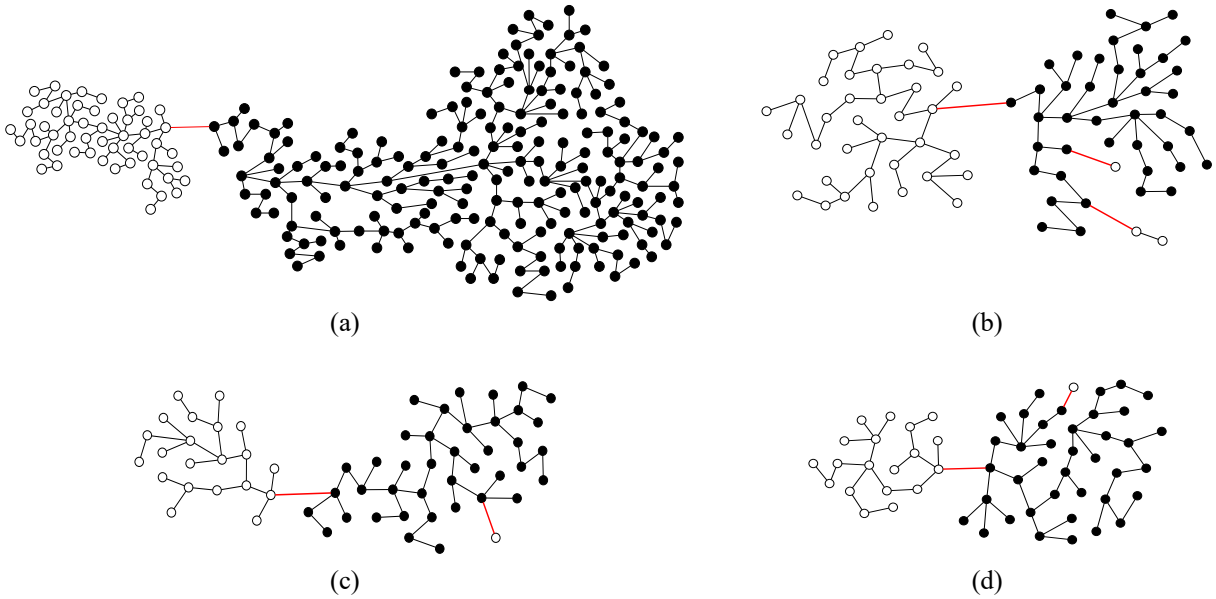



**Figure 3: The MST constructed for the feature set selected by MA3 with *MineMink* fitness function for (a) Shakespeare (b) Alzheimer (c) colon and (d) embryo dataset. Inter-class edges are shown in red.**

### 4.7. Feature Selection using other proximity graphs

In the previous sections, we presented extensive computational results of our approach (FSPMA) using the MST as the proximity graph. In this section, for the generalisation of the hypothesis, we replace the MST in the FSPMA with two other proximity graphs: the K-Nearest Neighbours (K-NN) and the Relative Neighbourhood Graph (RNG). The aim is to examine the usefulness of these proximity graphs in the FSPMA approach for feature selection and classification performance.

From the definition of K-NN graph presented in Section 2, it is clear that for different values of K, different K-NN graphs can be defined. Since it is impractical to test all values for K, we have chosen the value of 'K' in the K-NN graph from K=1 to K=$\lfloor \log10(m) + 0.5 \rfloor$. For the Shakespeare and Alzheimer's disease datasets, only 1-NN and 2-NN are studied. Similarly, 1-NN, 2-NN, and 3-NN are used for the Colon dataset and 1-NN, 2-NN, 3-NN, and 4-NN are used for the Embryo dataset. We repeated the same approach as we applied with MST. First, we run FSPMA (MA3) method 50 times for each proximity graph. The subset of features selected by the individual that presented the best fitness value (Fit), out of the 50 runs, is selected. We then evaluated the performance of the chosen feature subset using the 49 Weka classifiers with 10-fold cross validation. The average ACC and MCC values obtained by these 49 classifiers is calculated and compared using different proximity graphs. This process is repeated for each fitness criterion in each dataset.

The outcomes of this experiment are presented in Tables 11-14, where each table contains the results for each dataset. The proximity graph that achieved the best classification accuracy (ACC and MCC) is shown in bold for each fitness function.

**Table 11: Average ACC and MCC of 49 classifiers for the best subset of features generated by the MA3 method with different proximity graphs and fitness function in Shakespeare dataset.**

| | *PG* | *Fit* | *e* | *k* | *Time* | *Ev* | MCC | ACC |
|---|---|---|---|---|---|---|---|---|
| ***Mine*** | **MST** | **1** | **1** | **17** | **1403** | **11080** | **0.903** | **0.967** |
| | 1NN | 0 | 0 | 37 | 339 | 7733 | 0.891 | 0.963 |
| | 2NN | 0 | 0 | 55 | 629 | 8749 | 0.868 | 0.956 |
| | RNG | 2 | 2 | 48 | 40223 | 13418 | 0.876 | 0.958 |
| *MineMink* | **MST** | **0.07** | **1** | **17** | **2126** | **24219** | **0.900** | **0.966** |
| | 1NN | 0.05 | 0 | 14 | 462.2 | 13557 | 0.893 | 0.964 |
| | 2NN | 0.07 | 2 | 15 | 1063 | 21460 | 0.875 | 0.958 |
| | RNG | 0.08 | 10 | 14 | 67228 | 21300 | 0.858 | 0.953 |
| *MineMaxPGscore* | **MST** | **1.89** | **1** | **19** | **1615** | **14112** | **0.895** | **0.965** |
| | 1NN | 0 | 0 | 67 | 429 | 7447 | 0.875 | 0.958 |
| | 2NN | 2.82 | 1 | 89 | 1518 | 17494 | 0.844 | 0.949 |
| | RNG | 10.8 | 5 | 103 | 60875 | 21454 | 0.835 | 0.945 |
| *MineMinkMaxPGscore* | **MST** | **0.16** | **3** | **13** | **2667** | **24006** | **0.888** | **0.962** |
| | 1NN | 0.25 | 6 | 26 | 1420 | 29080 | 0.836 | 0.945 |
| | 2NN | 0.4 | 6 | 36 | 2210 | 32519 | 0.873 | 0.958 |
| | RNG | 0.33 | 15 | 23 | 80777 | 28066 | 0.821 | 0.940 |

**Table 12: Average ACC and MCC of 49 classifiers for the best subset of features generated by the MA3 method with different proximity graphs and fitness function in Alzheimer's dataset**

| | *PG* | *Fit* | *e* | *k* | *Time* | *Ev* | MCC | ACC |
|---|---|---|---|---|---|---|---|---|
| *Mine* | **MST** | **2** | **2** | **12** | **209** | **19113** | **0.786** | **0.892** |
| | 1NN | 1 | 1 | 36 | 92 | 8832 | 0.751 | 0.875 |
| | 2NN | 9 | 9 | 39 | 277 | 19568 | 0.746 | 0.872 |
| | RNG | 5 | 5 | 46 | 1497 | 16334 | 0.727 | 0.862 |
| ***MineMink*** | **MST** | **0.1** | **3** | **8** | **318** | **28864** | **0.790** | **0.895** |
| | 1NN | 0.15 | 5 | 10 | 47 | 5644 | 0.764 | 0.881 |
| | 2NN | 0.3 | 12 | 19 | 89 | 7628 | 0.769 | 0.883 |
| | RNG | 0.24 | 6 | 19 | 2979 | 30250 | 0.773 | 0.886 |
| *MineMaxPGscore* | **MST** | **10.09** | **2** | **15** | **789** | **44313** | **0.784** | **0.891** |
| | 1NN | 3.23 | 1 | 37 | 216 | 19265 | 0.744 | 0.871 |
| | 2NN | 25.85 | 14 | 48 | 519 | 38620 | 0.731 | 0.865 |
| | RNG | 16.25 | 10 | 35 | 2166 | 24651 | 0.722 | 0.860 |
| *MineMinkMaxPGscore* | **MST** | **0.41** | **3** | **12** | **506** | **35204** | **0.785** | **0.892** |
| | 1NN | 0.34 | 11 | 23 | 93 | 10742 | 0.732 | 0.865 |
| | 2NN | 0.56 | 21 | 22 | 459 | 37949 | 0.740 | 0.869 |
| | RNG | 0.42 | 24 | 26 | 2932 | 34656 | 0.754 | 0.876 |

**Table 13: Average ACC and MCC of 49 classifiers for the best subset of features generated by the MA3 method with different proximity graphs and fitness function in colon cancer dataset**

| | *PG* | *Fit* | *e* | *k* | *Time* | *Ev* | MCC | ACC |
|---|---|---|---|---|---|---|---|---|
| *Mine* | **MST** | **2** | **2** | **66** | **136** | **2832** | **0.622** | **0.827** |
| | 1NN | 0 | 0 | 117 | 27 | 1750 | 0.551 | 0.794 |
| | 2NN | 6 | 6 | 101 | 34 | 1981 | 0.551 | 0.794 |
| | 3NN | 3 | 3 | 146 | 71 | 2692 | 0.530 | 0.786 |
| | RNG | 5 | 5 | 117 | 124 | 2803 | 0.524 | 0.781 |
| *MineMink* | **MST** | **0.06** | **2** | **69** | **69** | **3114** | **0.630** | **0.831** |
| | 1NN | 0.07 | 1 | 114 | 37 | 2287 | 0.513 | 0.776 |
| | 2NN | 0.12 | 4 | 107 | 49 | 3154 | 0.553 | 0.797 |
| | 3NN | 0.10 | 2 | 137 | 75 | 3012 | 0.547 | 0.793 |
| | RNG | 0.12 | 4 | 93 | 226 | 5357 | 0.538 | 0.787 |
| *MineMaxPGscore* | **MST** | **6.63** | **4** | **71** | **487** | **8917** | **0.629** | **0.830** |
| | 1NN | 5.12 | 5 | 123 | 142 | 7996 | 0.513 | 0.781 |
| | 2NN | 9.48 | 11 | 124 | 111 | 5631 | 0.489 | 0.765 |
| | 3NN | 8.93 | 3 | 212 | 178 | 5434 | 0.524 | 0.785 |
| | RNG | 5.98 | 12 | 159 | 461 | 9456 | 0.516 | 0.779 |
| ***MineMinkMaxPGscore*** | **MST** | **0.11** | **4** | **50** | **254** | **7402** | **0.653** | **0.842** |
| | 1NN | 0.33 | 2 | 130 | 40 | 2478 | 0.514 | 0.780 |
| | 2NN | 0.17 | 12 | 126 | 111 | 6045 | 0.525 | 0.785 |
| | 3NN | 0.54 | 29 | 186 | 191 | 6486 | 0.503 | 0.774 |
| | RNG | 0.11 | 10 | 109 | 438 | 8810 | 0.530 | 0.788 |

**Table 14: Average ACC and MCC of 49 classifiers for the best subset of features generated by the MA3 method with different proximity graphs and fitness function in embryonal tumour dataset**

| | ***PG*** | ***Fit*** | ***e*** | ***k*** | ***Time*** | ***Ev*** | **MCC** | **ACC** |
|---|---|---|---|---|---|---|---|---|
| *Mine* | MST | 2 | 2 | 68 | 289 | 2988 | 0.456 | 0.756 |
| | **1NN** | **1** | **1** | **56** | **12** | **3120** | **0.473** | **0.768** |
| | 2NN | 1 | 1 | 88 | 56 | 5245 | 0.453 | 0.761 |
| | 3NN | 0 | 0 | 73 | 64 | 4497 | 0.462 | 0.765 |
| | 4NN | 0 | 0 | 81 | 45 | 2657 | 0.461 | 0.765 |
| | RNG | 4 | 4 | 54 | 487 | 10302 | 0.460 | 0.766 |
| ***MineMink*** | **MST** | **0.08** | **2** | **70** | **140** | **2841** | **0.509** | **0.782** |
| | 1NN | 0.09 | 1 | 45 | 11 | 2800 | 0.488 | 0.777 |
| | 2NN | 0.10 | 0 | 52 | 71 | 6226 | 0.437 | 0.754 |
| | 3NN | 0.15 | 0 | 76 | 47 | 3290 | 0.435 | 0.749 |
| | 4NN | 0.11 | 0 | 57 | 118 | 6136 | 0.469 | 0.768 |
| | RNG | 0.18 | 5 | 57 | 168 | 3901 | 0.447 | 0.760 |
| *MineMaxPGscore* | **MST** | **2.16** | **6** | **73** | **509** | **8452** | **0.465** | **0.764** |
| | 1NN | 0.14 | 19 | 67 | 77 | 17717 | 0.194 | 0.653 |
| | 2NN | 0.41 | 41 | 107 | 280 | 21340 | 0.222 | 0.670 |
| | 3NN | 0.71 | 65 | 56 | 400 | 28408 | 0.210 | 0.664 |
| | 4NN | 0.64 | 83 | 111 | 574 | 29067 | 0.217 | 0.673 |
| | RNG | 0.19 | 21 | 71 | 770 | 16914 | 0.337 | 0.719 |
| *MineMinkMaxPGscore* | **MST** | **0.16** | **5** | **59** | **343** | **4894** | **0.465** | **0.764** |
| | 1NN | 0.02 | 11 | 48 | 10 | 2743 | 0.405 | 0.743 |
| | 2NN | 0.01 | 40 | 65 | 159 | 16207 | 0.362 | 0.726 |
| | 3NN | 0.01 | 60 | 52 | 211 | 15976 | 0.347 | 0.717 |
| | 4NN | 0.01 | 88 | 49 | 463 | 26115 | 0.278 | 0.694 |
| | RNG | 0.00 | 20 | 76 | 519 | 12884 | 0.366 | 0.729 |

In all four datasets, MST demonstrated superiority in terms of MCC and ACC over other proximity graphs. The classification performance of the FSPMA using MST generated feature sets with the best fitness over different datasets. The next best-performing proximity graph was the 1-NN, which also attained the best results in the case of the embryonal tumour dataset using the *Mine* fitness function. Moreover, as expected, 1-NN was the best-performing proximity graph in terms of execution time (i.e., the time required to run the MA3 code using the corresponding proximity graph) and 2-NN came next. On the other hand, the execution time required by RNG was often ten times or even more than that required by 1-NN. As reported earlier, FSPMA achieved the best fitness value using *Mine* and *MineMink* fitness functions.

Next, as we have done in the previous section, for each fitness function and dataset, we selected the five top-performing classifiers (out of 49 classifiers) for the feature sets generated by the different proximity graphs (MST, K-NN and RNG) and we calculated the average MCC and ACC obtained by the top five classifiers, which are summarized in Table 15. When the selected feature sets were evaluated using high-quality classifiers, the superiority of the *MineMink* fitness function with MST is unanimously established over other fitness functions. The results in Table 15, once again highlight that the feature sets selected using MST and *MineMink* fitness function are superior to the other proximity graphs and other fitness functions.

**Table 15: Average ACC and MCC for the pool of classifiers created by accumulating the best five performing classifiers in different proximity graphs.**

| | | *Mine* | | *MineMink* | | *MineMaxMSTscore* | | *MineMinkMaxMSTscore* | |
|---|---|---|---|---|---|---|---|---|---|
| | | **ACC** | **MCC** | **ACC** | **MCC** | **ACC** | **MCC** | **ACC** | **MCC** |
| Shakespeare | MST | **0.982** | **0.948** | **0.983** | **0.949** | **0.980** | **0.941** | **0.982** | **0.948** |
| | 1NN | 0.980 | 0.941 | 0.973 | 0.921 | 0.978 | 0.936 | 0.973 | 0.921 |
| | 2NN | 0.976 | 0.929 | 0.969 | 0.907 | 0.969 | 0.907 | 0.981 | 0.945 |
| | RNG | 0.975 | 0.926 | 0.969 | 0.908 | 0.976 | 0.929 | 0.959 | 0.876 |
| Alzheimer's disease | MST | **0.936** | **0.874** | **0.942** | **0.883** | **0.933** | **0.866** | **0.930** | **0.861** |
| | 1NN | 0.906 | 0.813 | 0.881 | 0.762 | 0.906 | 0.814 | 0.898 | 0.798 |
| | 2NN | 0.911 | 0.823 | 0.915 | 0.830 | 0.900 | 0.802 | 0.898 | 0.796 |
| | RNG | 0.901 | 0.803 | 0.890 | 0.781 | 0.901 | 0.804 | 0.903 | 0.808 |
| Colon cancer | MST | **0.861** | **0.696** | **0.867** | **0.709** | **0.858** | **0.692** | **0.865** | **0.707** |
| | 1NN | 0.834 | 0.629 | 0.811 | 0.582 | 0.812 | 0.586 | 0.815 | 0.587 |
| | 2NN | 0.817 | 0.595 | 0.829 | 0.626 | 0.799 | 0.565 | 0.822 | 0.606 |
| | 3NN | 0.809 | 0.577 | 0.751 | 0.459 | 0.829 | 0.617 | 0.819 | 0.597 |
| | RNG | 0.814 | 0.584 | 0.820 | 0.602 | 0.817 | 0.593 | 0.810 | 0.579 |
| Embryonal tumour | MST | **0.814** | **0.573** | **0.851** | **0.669** | **0.800** | **0.549** | **0.807** | **0.558** |
| | 1NN | 0.795 | 0.535 | 0.804 | 0.557 | 0.675 | 0.281 | 0.794 | 0.525 |
| | 2NN | 0.784 | 0.501 | 0.747 | 0.443 | 0.708 | 0.326 | 0.745 | 0.392 |
| | 3NN | 0.794 | 0.528 | 0.749 | 0.465 | 0.663 | 0.253 | 0.746 | 0.404 |
| | 4NN | 0.789 | 0.517 | 0.765 | 0.472 | 0.726 | 0.368 | 0.716 | 0.328 |
| | RNG | 0.791 | 0.516 | 0.764 | 0.470 | 0.687 | 0.303 | 0.701 | 0.279 |

### 4.8. Comparison with Other Feature Selection Methods

In this section, we establish the competitiveness of the proposed FSPMA method by comparing it with other well-known feature selection techniques. We compared the performance of the most successful setup of FSPMA (MA3 algorithm with *MineMink* and MST) with nine univariate and multivariate feature selection methods. Among the univariate supervised methods chosen are Chi-square feature selection (Chi), Information gain ratio (GainRatio), Information gain (IG), ReliefF, Symmetrical Uncertainty (SU) from WEKA and CM1 score [39]. Among the multivariate supervised methods, we selected Correlation based Feature Selection (CFS), Consistency subset feature selection (Consistency) from WEKA and *(α,β)-k*Feature Set [40]. We implemented the (*α*,*β*)-*k*-feature set and CM1 score methods and for all other methods, we adopted the WEKA implementation using their default configurations. For CM1, following the same approach of [39], we selected the 20 highest and 20 lowest CM1 markers for all datasets. In order to be consistent, we chose the same number of features for other univariate feature selection methods.

Table 16 depicts the results for *k*, ACC and MCC, where in this case *k* is the number of features finally selected by the method ('*All*' indicates the total number of features in each dataset). FSPMA obtained the best results in regards to ACC and MCC in every dataset. In addition, the FSPMA achieved the lowest value of *k* in the Shakespeare and AD training datasets, and the value of *k* did not increase significantly with the total number of features. The CFS and the Consistency method selected the largest and the second largest feature set, respectively, for the colon cancer and embryonal tumour datasets. Moreover, the size of feature sets for these methods increased dramatically with the increase in a total number of features. Although the (*α*,*β*)-*k*-feature set method selected the largest feature set for the Shakespeare datasets, its performance was not affected by the total number of features.

In order to evaluate the classification performance of the different selected feature sets, we used the same 49 WEKA classifiers as before. The average MCC and ACC scores of 49 WEKA classifiers are compared in Table 16. FSPMA's *MineMink* fitness function outperformed all the feature selection methods in terms of average MCC and average ACC in all datasets. Clearly, all feature selection methods reduced the dimensionality of the colon cancer dataset, but few improved classification performance as suggested by the average MCC and ACC results for all features (All). In contrast, FSPMA's *MineMink* fitness function reduced the dimensionality of all datasets as well as achieved impressive results compared with all features (*All*). The

second best results in terms of MCC and ACC were achieved by CFS, *(α,β)-k*-feature set, GainRation and SU in Shakespeare, AD, colon and embryo datasets respectively.

**Table 16: Comparison of different feature selection methods in terms of feature-subset size (k), MCC and ACC**

| | | Dataset | | | |
|---|---|---|---|---|---|
| | | **Shakespeare** | **Alzheimer's disease** | **Colon cancer** | **Embryonal tumour** |
| *All* | *k* | 220 | 120 | 2000 | 7129 |
| | ACC | 0.949 | 0.848 | 0.744 | 0.619 |
| | MCC | 0.849 | 0.698 | 0.443 | 0.111 |
| ***FSPMA*** | *k* | 17 | 8 | 69 | 70 |
| | ACC | **0.966** | **0.895** | **0.831** | **0.782** |
| | MCC | **0.900** | **0.791** | **0.630** | **0.509** |
| *CFS* | *k* | 88 | 27 | 642 | 1762 |
| | ACC | 0.957 | 0.872 | 0.770 | 0.648 |
| | MCC | 0.870 | 0.746 | 0.503 | 0.187 |
| *Consistency* | *k* | 18 | 33 | 236 | 1402 |
| | ACC | 0.904 | 0.814 | 0.740 | 0.570 |
| | MCC | 0.716 | 0.630 | 0.448 | 0.007 |
| *(α,β)-k*-feature set | *k* | 140 | 10 | 60 | 30 |
| | ACC | 0.952 | 0.891 | 0.821 | 0.756 |
| | MCC | 0.856 | 0.784 | 0.606 | 0.450 |
| *Chi* | *k* | 40 | 40 | 40 | 40 |
| | ACC | 0.946 | 0.871 | 0.804 | 0.765 |
| | MCC | 0.839 | 0.744 | 0.571 | 0.460 |
| *GainRatio* | *k* | 40 | 40 | 40 | 40 |
| | ACC | 0.945 | 0.877 | 0.826 | 0.775 |
| | MCC | 0.835 | 0.755 | 0.621 | 0.487 |
| *IG* | *k* | 40 | 40 | 40 | 40 |
| | ACC | 0.953 | 0.872 | 0.808 | 0.760 |
| | MCC | 0.861 | 0.746 | 0.579 | 0.459 |
| *ReliefF* | *k* | 40 | 40 | 40 | 40 |
| | ACC | 0.953 | 0.872 | 0.808 | 0.760 |
| | MCC | 0.861 | 0.746 | 0.579 | 0.459 |
| *SU* | *k* | 40 | 40 | 40 | 40 |
| | ACC | 0.949 | 0.872 | 0.803 | 0.777 |
| | MCC | 0.848 | 0.745 | 0.567 | 0.494 |
| *CM1* | *k* | 40 | 40 | 40 | 40 |
| | ACC | 0.941 | 0.860 | 0.813 | 0.594 |
| | MCC | 0.824 | 0.722 | 0.587 | 0.067 |

Once again we evaluated the selected feature set obtained by FSPMA using the top performing classifiers as in previous sections. For each dataset, we condensed the results from the top five performing classifiers (out of 49 classifiers) for the feature sets generated by different feature selection methods. Then we evaluated each of these feature sets using those top classifiers and the average MCC and ACC are presented in Table 17. We observe that the FSPMA continues to exhibit superior performance compared to other feature selection methods in all datasets. The average performance of all feature selection methods was improved when high-quality classifiers were used but FSPMA outperformed all of those in each dataset.

**Table 17: Average ACC and MCC for the pool of classifiers created by accumulating the best five performing classifiers in different feature selection methods.**

| | | Shakespeare | Alzheimer's disease | Colon cancer | Embryonal tumour |
|---|---|---|---|---|---|
| ***FSPMA*** | ACC | **0.983** | **0.925** | **0.867** | **0.834** |
| | MCC | **0.951** | **0.851** | **0.709** | **0.617** |
| *CFS* | ACC | 0.976 | 0.885 | 0.796 | 0.685 |
| | MCC | 0.928 | 0.771 | 0.561 | 0.250 |
| *Consistency* | ACC | 0.929 | 0.842 | 0.752 | 0.565 |
| | MCC | 0.791 | 0.687 | 0.471 | -0.002 |
| *(α,β)-k-feature set* | ACC | 0.968 | 0.912 | 0.845 | 0.775 |
| | MCC | 0.905 | 0.827 | 0.664 | 0.494 |
| *Chi* | ACC | 0.959 | 0.885 | 0.815 | 0.786 |
| | MCC | 0.876 | 0.770 | 0.597 | 0.515 |
| *GainRatio* | ACC | 0.963 | 0.892 | 0.841 | 0.807 |
| | MCC | 0.889 | 0.785 | 0.658 | 0.566 |
| *IG* | ACC | 0.966 | 0.887 | 0.818 | 0.771 |
| | MCC | 0.896 | 0.775 | 0.604 | 0.497 |
| *ReliefF* | ACC | 0.955 | 0.869 | 0.820 | 0.729 |
| | MCC | 0.867 | 0.738 | 0.600 | 0.389 |
| *SU* | ACC | 0.965 | 0.882 | 0.804 | 0.806 |
| | MCC | 0.897 | 0.764 | 0.574 | 0.564 |
| *CM1* | ACC | 0.954 | 0.878 | 0.833 | 0.755 |
| | MCC | 0.863 | 0.757 | 0.631 | 0.446 |

## 5. Discussion

We proposed a novel multivariate filter feature selection method employing proximity graphs by considering the distance relationships among samples, given a subset of features. The essence of the approach relies on the hypothesis that for an appropriate set of discriminatory features, the nodes from the same class will cluster in the constructed proximity graph. Therefore, the objective is to find the subset of features which can minimise the number of edges between the samples belonging to different classes.

Based on the above proposition, we designed several scoring functions to measure the quality of a feature set from the topological makeup of the constructed proximity graph. We proposed to quantify the quality of a feature subset based on its size, the number of edges connecting nodes (samples) from different classes in the built proximity graph, and a score calculated from each proximity graph. Based on these variables, we presented four evaluation criteria to evaluate the selected subset of features: *Mine*, *MineMink*, *MineMaxPGscore* and *MineMinkMaxPGscore*.

We proposed a memetic algorithm called FSPMA ('Feature Selection with Proximity graph using Memetic Algorithm'). We designed five different local search algorithms and tested their standalone search effectiveness in terms of fitness improvement and time requirement. After ascertaining that all the designed LS strategies have the ability to improve quality of a feature set, we embedded these local search algorithms in a generational genetic algorithm and designed five memetic algorithms. The experimental results on four real-world datasets showed that the designed hybrid algorithms can explore the search space more effectively than the global search algorithm. We also incorporated an adaptation strategy to balance exploration and exploitation of the search space. The adaptive memetic algorithms were found not only effective but also efficient in optimizing the fitness functions. Among the five memetic algorithms, MA3 outperformed all others significantly when tested with different fitness functions and datasets. Based on this empirical study, we presume that the reason behind the superior performance of MA3 over other MAs is the synergy of the high quality local search operator with the other genetic operators in the common MA framework.

In our study, we used 49 machine learning algorithms from the WEKA software package to evaluate the set of features obtained by the best-performing FSPMA method (MA3) with four proposed fitness functions. We also evaluated the feature-sets using a pool of best-performing classifiers in order to test their classification performance with high-quality classifiers. These analyses show that the FSPMA can reduce the dimensionality and increase classification performance of the algorithms. Due to tractability and practicability, the majority

of analysis in this work was done by using MST as the proximity graph. After establishing the validity of the method we tested the generalization of the approach by testing it on other types of proximity graphs – specifically on K-NN and RNG. This study showed that the method generalized well with other proximity graphs in terms of dimensionality reduction and classification improvement. However, comparing among different types of proximity graphs it was clear that MST can deliver the maximum classification performance when used in the FSPMA framework.

In order to establish the competitiveness of the proposed approach, we compared the results obtained by FSPMA with nine well-known feature selection methods. FSPMA again demonstrated its superiority over the nine methods using the 49 classifiers and using selecting high-quality classifiers. All the empirical study undertaken in this work established the validity and usefulness of the proposed approach for feature selection.

The results of the study suggest that the method is indeed highly scalable since the results with the MST have shown to be very satisfactory in comparison with other choices of proximity graphs. We note that the computational complexity of FSPMA depends on the number of samples, the used fitness criteria and, most importantly, by the type of proximity graph used (i.e. it takes significantly higher time when RNG is used instead of using either the MST or the 1NN graphs). It has been interesting to observe that in this study the best performing FSPMA algorithm uses the MST which scales well for datasets with a larger number of samples. In addition, the MST worked well with the MineMink fitness criteria hence this pair of proximity graph and fitness criteria encourages their joint use for larger datasets.

## 6. Conclusion

We presented a multivariate filter feature selection method that uses proximity graphs for evaluating the quality of a feature set using proposed fitness scores. In order to search for the optimal feature subset, we implemented an adaptive memetic algorithm that works by optimizing the fitness scores. An extensive study was undertaken using four real-world datasets where we investigate the effect of different types of proximity graphs, fitness scores and memetic algorithmic setups. The effectiveness of the proposed method was established by evaluating the selected features with well known classifiers and comparing with other established feature selection methods.

The current work deserves further investigation in a few important directions. First of all, the only distance metric used in this work was Euclidean distance whereas there are many other metrics exist. It would be interesting to apply a few other important distance metrics as Mahalanobis, Minkowski and Spearman's rank-based distances. Secondly, in the proposed fitness functions, multiple criteria (e.g. feature set size, number of inter-class edges) were combined to formulate a single objective optimisation problem. It would be worth to investigate these multiple criteria as a multi-objective optimisation problem. Finally, the methodology could be investigated in multi-class datasets.

**Acknowledgements**

Pablo Moscato acknowledges support from the Australian Research Council Future Fellowship FT120100060. Pablo Moscato and Regina Berretta acknowledge support from the Australian Research Council Discovery Projects DP120102576.

## References

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research,* vol. 3, pp. 1157-1182, 2003.

[2] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, no. 4, pp. 491-502, 2005.

[3] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis,* vol. 1, no. 3, pp. 131-156, 1997.

[4] A. S. Arefin, R. Vimieiro, C. Riveros, H. Craig, and P. Moscato, "An Information Theoretic Clustering Approach for Unveiling Authorship Affinities in Shakespearean Era Plays and Poems," *PLoS ONE,* vol. 9, no. 10, p. e111445, 2014.
[5] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications,* vol. 24, no. 1, pp. 175-186, 2014.
[6] M. A. Hall, "Correlation-based feature selection for machine learning," The University of Waikato, 1999.
[7] Z. Zhao and H. Liu, "Searching for interacting features," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, vol. 7, pp. 1156-1161.
[8] I. Kononenko, "Estimating attributes: analysis and extensions of RELIEF," in *Proceedings of the European Conference on Machine Learning (ECML-94)*, 1994, pp. 171-182: Springer.
[9] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of ReliefF and RReliefF," *Machine Learning,* vol. 53, no. 1-2, pp. 23-69, 2003.
[10] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 27, no. 8, pp. 1226-1238, 2005.
[11] S. Seth and J. C. Principe, "Variable selection: A statistical dependence perspective," in *Proceedings of the 9th International Conference on Machine Learning and Applications (ICMLA)*, 2010, pp. 931-936: IEEE.
[12] C. Liu, W. Y. Wang, Q. Zhao, X. M. Shen, and M. Konan, "A new feature selection method based on a validity index of feature subset," (in English), *Pattern Recognition Letters,* vol. 92, pp. 1-8, Jun 1 2017.
[13] Y. Chao, L. Ya-Feng, J. Bo, H. Jungong, and H. Junwei, "LLE Score: A New Filter-Based Unsupervised Feature Selection Method Based on Nonlinear Manifold Embedding and Its Application to Image Recognition," *IEEE Trans Image Process,* vol. 26, no. 11, pp. 5257-5269, Nov 2017.
[14] C. Cotta, C. Sloper, and P. Moscato, "Evolutionary search of thresholds for robust feature set selection: application to the analysis of microarray data," in *Applications of Evolutionary Computing*: Springer, 2004, pp. 21-30.
[15] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," (in English), *Applied Soft Computing,* vol. 62, pp. 441-453, Jan 2018.
[16] M. M. Mafarja, D. Eleyan, I. Jaber, S. Mirjalili, and A. Hammouri, "Binary Dragonfly Algorithm for Feature Selection," (in English), *2017 International Conference on New Trends in Computing Sciences (Ictcs),* pp. 12-17, 2017.
[17] S. K. Gu, R. Cheng, and Y. C. Jin, "Feature selection for high-dimensional classification using a competitive swarm optimizer," (in English), *Soft Computing,* vol. 22, no. 3, pp. 811-822, Feb 2018.
[18] J. Xie, W. Xie, C. Wang, and X. Gao, "A novel hybrid feature selection method based on IFSFFS and SVM for the diagnosis of erythemato-squamous diseases," *Journal of Machine Learning Research-Proceedings Track,* vol. 11, pp. 142-151, 2010.
[19] O. Abedinia, N. Amjady, and H. Zareipour, "A New Feature Selection Technique for Load and Price Forecast of Electrical Power Systems," (in English), *Ieee Transactions on Power Systems,* vol. 32, no. 1, pp. 62-74, Jan 2017.
[20] F. Neri, C. Cotta, and P. Moscato, *Handbook of memetic algorithms*. Springer, 2011.
[21] A. Abu Zaher, R. Berretta, A. S. Arefin, and P. Moscato, "FSMEC: A feature selection method based on the minimum spanning tree and evolutionary computation," presented at the Proceedings of the 13th Australian Data Mining Conference (AusDM 2015), Sydney, Australia, 2015.
[22] A. Abu Zaher, R. Berretta, N. Noman, and P. Moscato, "A new computational intelligence approach for feature selection using proximity graphs," in *Proceedings of the Applied Informatics and Technology Innovation Conference (AITIC)*, 2016.
[23] P. Bose *et al.*, "Proximity graphs: E, Δ, Δ, χ and ω," *International Journal of Computational Geometry & Applications,* vol. 22, no. 05, pp. 439-469, 2012.
[24] M. A. Carreira-Perpinán and R. S. Zemel, "Proximity Graphs for Clustering and Manifold Learning," in *Neural Information Processing Systems (NIPS)* 2004, vol. 17, pp. 225-232.
[25] M. Inostroza-Ponta, R. Berretta, A. Mendes, and P. Moscato, "An automatic graph layout procedure to visualize correlated data," in *IFIP International Conference on Artificial Intelligence in Theory and Practice*, 2006.

[26] P. Moscato, *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms* (Caltech Concurrent Computation Program, C3P Report). 1989.
[27] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper–filter feature selection algorithm using a memetic framework," *IEEE Transactions on Systems and Cybernetics,* vol. 37, no. 1, pp. 70-76, 2007.
[28] J. Lee and D. W. Kim, "Memetic feature selection algorithm for multi-label classification," (in English), *Information Sciences,* vol. 293, pp. 80-96, Feb 1 2015.
[29] A. Moser and M. N. Murty, "On the scalability of genetic algorithms to very large-scale feature selection," in *Real-World Applications of Evolutionary Computing*: Springer, 2000, pp. 77-86.
[30] J. Y. Lin and Y. P. Chen, "When and What Kind of Memetic Algorithms Perform Well," (in English), *2012 Ieee Congress on Evolutionary Computation (Cec),* 2012.
[31] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," (in English), *Ieee Transactions on Evolutionary Computation,* vol. 7, no. 2, pp. 204-223, Apr 2003.
[32] H. Iba and N. Noman, *New frontier in evolutionary algorithms: theory and applications*. World Scientific Publishing Co., Inc., 2011.
[33] S. Ray *et al.*, "Classification and prediction of clinical Alzheimer's diagnosis based on plasma signaling proteins," *Nature medicine,* vol. 13, no. 11, pp. 1359-1362, 2007.
[34] M. G. Ravetti and P. Moscato, "Identification of a 5-protein biomarker molecular signature for predicting Alzheimer's disease," *PloS One,* vol. 3, no. 9, p. e3111, 2008.
[35] H. Craig and R. Whipp, "Old spellings, new methods: automated procedures for indeterminate linguistic data," *Literary and Linguistic Computing,* vol. 25, no. 1, pp. 37-52, 2010.
[36] U. Alon *et al.*, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences,* vol. 96, no. 12, pp. 6745-6750, 1999.
[37] S. L. Pomeroy *et al.*, "Prediction of central nervous system embryonal tumour outcome based on gene expression," *Nature,* vol. 415, no. 6870, pp. 436-442, 2002.
[38] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, 2011.
[39] J. Marsden, D. Budden, H. Craig, and P. Moscato, "Language individuation and marker words: shakespeare and his maxwell's demon," *PloS One,* vol. 8, no. 6, p. e66813, 2013.
[40] R. Berretta, W. Costa, and P. Moscato, "Combinatorial optimization models for finding genetic signatures from gene expression datasets," in *Bioinformatics*: Springer, 2008, pp. 363-377.