A Bi-objective Feature Selection Algorithm for Large Omics Datasets

Luís Cavique [1,2] [0000-0002-5590-1493], Armando B. Mendes [3,4] [0000-0003-3049-5852], Hugo F.M.C. Martiniano [1,5] [0000-0003-2490-8913] and Luís Correia [1] [0000-0003-2439-1168]

[1] MAS-BioISI, FCUL, Lisboa, Portugal,
[2] Universidade Aberta, Lisboa, Portugal,
[3] Universidade Açores, Ponta Delgada, Portugal,
[4] Algoritmi, Universidade do Minho, Portugal,
[5] Instituto Dr. Ricardo Jorge, Lisboa, Portugal,
luis.cavique@uab.pt, armando.b.mendes@uac.pt,
hfmartiniano@fc.ul.pt, luis.correia@ciencias.ulisboa.pt

Abstract. Feature selection is one of the most important concepts in data mining when dimensionality reduction is needed. The performance measures of feature selection encompass predictive accuracy and result comprehensibility. Consistency based methods are a significant category of feature selection research that substantially improves the comprehensibility of the result using the parsimony principle. In this work, the bi-objective version of the algorithm Logical Analysis of Inconsistent Data is applied to large volumes of data. In order to deal with hundreds of thousands of attributes, heuristic decomposition uses parallel processing to solve a set covering problem and a cross-validation technique. The bi-objective solutions contain the number of reduced features and the accuracy. The algorithm is applied to omics datasets with genome-like characteristics of patients with rare diseases.

Keywords: feature selection, logical analysis of data, heuristic decomposition, bi-objective optimization

1. Introduction

In feature selection two main objectives sustain the performance measures, the predictive accuracy and the result comprehensibility [Liu, Yu 2005]. Some models do not consider the predictive accuracy, whereas others tend to destroy the underlying semantics of the features after reduction. It would be highly desirable to find a method that could not only reduce the number of features, but also preserve the data semantics.

In this context, Rough Set theory emerges as an important tool to discover data dependencies and reduce dimensionality. Rough Set theory was initially proposed as a tool to reason about vagueness and uncertainty in information systems by Pawlak [Pawlak 1982] and later it was also proposed for attribute selection [Pawlak 1991]. Rough Sets determine a lower and an upper approximation for each class, rather than correct or exclude data inconsistencies.

In parallel, Peter Hammer's group [Crama et al. 1988], [Boros et al. 2000], with works in discrete optimization, developed LAD, Logical Analysis of Data. The key features of LAD are the discovery of the minimum number of attributes necessary to explain all

observations and the detection of hidden patterns in a dataset with two classes. An extension of the Boolean approach uses nominal non-binary attributes.

LAD and Rough Set approaches are a subset of filter models that aim to reduce the number of attributes of datasets using two phases: problem transformation and optimization. Their specificity is to keep the semantics of the data by only removing the redundant data.

By combining the two approaches, we proposed Logical Analysis of Inconsistent Data, LAID [Cavique et al. 2013], which blends the best characteristics of both methods. The LAID method should deal with integer attributes associated with costs as in LAD and be tolerant to inconsistency as in Rough Sets. The integration of both approaches is so close that LAID can be seen as a Rough Set extension.

One key area for the application of the LAID method is high-dimensional omics datasets, such as those generated by high-throughput sequencing technologies. In the past few years, technological developments in this area have resulted in an astonishing growth of the amount of data produced. As most disease related omics datasets are subject to restricted access, artificial datasets with similar characteristics have been created.

The goal of this work is to solve a feature selection problem with a dataset involving two thousand observations and one million attributes. To deal with millions of attributes a heuristic decomposition is used in a parallel computer environment. We will consider two performance measures for comprehensibility of the result and accuracy. In the comprehensibility of the solution, we aim to achieve the minimum number of attributes with the maximum actionable knowledge that better explains the dataset to the final user. The accuracy is given by a cross-validation technique.

This document extends on works with LAID characterization [Cavique et al. 2011], [Cavique et al. 2013]. The application of LAID to larger datasets with the development of a heuristic decomposition approach with a sub-problem algorithm was presented in [Cavique et al. 2017]. Following the previous work, the novelty of this document is the presentation of the master problem algorithm with a bi-objective approach for omics datasets. The bi-objective reflects both the predictive accuracy concern and the result comprehensibility.

This document is organized as follows. In Section 2, we present the related concepts of feature selection, omics datasets and heuristic decomposition. In Section 3, we present the sub-problem algorithm that finds the feature selection with the minimum number of attributes and computes the accuracy. In Section 4, the heuristic decomposition algorithm and the bi-objective master problem algorithm are presented. In Section 5, the computational results are shown. Finally, in the last section we draw some conclusions about this new approach.

2. Related work

This work combines the areas of attribute selection and problem decomposition to deal with omics datasets. Consequently, in this section we introduce the related topics: feature selection, omics data and heuristic decomposition. As they will be combined in the proposed sub-problem algorithm, two feature selection methods, Rough Sets and LAD are detailed.

2.1. Feature Selection

The motivation to reduce the dimensionality of the feature space is closely related to the decreased time required to double information in the world every year. Surveys on feature selection methods can be found in [Liu, Yu 2005], [Chandrashekar, Sahin 2014].

In feature selection, given thousands to millions of features, the goal is to select the most relevant ones. The performance measures have two main goals, the predictive accuracy and the result comprehensibility. In this work we emphasize the comprehensibility of the results, following Occam's razor principle that aims to obtain the simplest model.

We consider a dataset $D=\{O, X \cup C\}$ where $O=\{O_1,O_2,\ldots,O_n\}$ is a non-empty set of observations (instances or cases), $X=\{X_1,X_2,\ldots,X_m\}$ is a non-empty set of features (attributes or columns) and C is the class attribute.

In this brief review we use the feature selection taxonomies reported in [Liu, Yu 2005]. There are three basic models in feature selection: Filter, Wrapper and Hybrid model.

In the Filter model the most popular independent criteria are consistency measures, distance measures, information measures and dependency measures. In this sub-section the distance measure and the consistency measure are detailed because they are used in this document.

Distance measure criteria are applied in observations within the same class and in observations of diverse classes. In the same class, to obtain the disagreement value, the logic operator XOR is applied, where it returns True if $(O_x, X_a) \neq (O_y, X_a)$ and False otherwise. With diverse classes, the logic operator XOR can also be applied, where if $(O_x, X_a) \neq (O_y, X_a)$, feature X_a should be chosen because it differentiates the classes. Comparisons of observations within the same class measure the incoherence or noise of the feature. On the other hand, comparisons of observations between different classes measure how strong a feature is in the discrimination or separation of the classes. RELIEF algorithm [Kira, Rendell 1992] evaluates a feature subset based on the subtraction between the distance of observations in different classes and the distance of observations within the same class.

Using the consistency measure criteria, an inconsistency occurs when two or more observations have the same values in all attributes but belong to different decision classes. This measure is used in algorithms that attempt to find the minimum number of features with the minimum number of inconsistencies. The well-known algorithm FOCUS [Almuallim, Dietterich 1991] uses the concept of consistency, where irrelevant

features are removed. The LAD algorithm [Crama et al. 1988], [Boros et al. 2000] is developed in two steps. In the first step, a disjoint matrix is obtained using the XOR operator to compare observations from different classes. In the second step, a set covering algorithm is applied to ensure data consistency.

In the feature selection process using addition or removal of attributes, the information measure is given by the information gain from an added or removed feature.

The dependency measure of a feature is related to how important the correlation is between the feature and the class.

The Filter model is divided into two sequential steps. The feature selection step is executed before the learning phase of the prediction model, and there is no interaction between the selection and the prediction model.

The Wrapper model [John et al. 1994] is also divided into two steps, but with strong interaction between the feature selection phase and the learning phase, where the results of the prediction are used as a criterion of feature choice.

Filter models are more intuitive and show better performance since they build the solution in a constructive process without iterations. On the other hand, they present the disadvantages of ignoring the predictive process. Consequently, most of the relevant features might not be adequate in the prediction model and the selection criterion is hard to estimate.

In the Wrapper models, the selection criterion is easy to estimate since the features are chosen by the prediction model, and therefore they are classified as model-aware as they incorporate the knowledge of the predictor. Contrary to the Filter models, Wrapper models are computationally expensive and less intuitive. The main disadvantage of this method is the increasing overfitting risk when the number of observations is insufficient. In other words, the Wrapper models do not identify statistical dependency, so the features might not be the main explanatory variables and therefore the model can lose its theoretical basis.

Hybrid methods have been proposed to reduce features in classification by combining the advantages of the two previous methods. The main disadvantage of the method is the significant low scalability when the feature number increases.

To sum up, in Filter methods, the selection criterion is hard to estimate, whereas Wrapper methods tend to destroy the underlying feature semantics.

In this work, we opt for Filter methods for two main reasons: (i) given that large volumes of data should be processed, Filter methods are more computationally efficient, (ii) given that the result comprehensibility is an important criterion, Filter methods are preferable since they preserve the semantics of the data.

2.2. Rough Sets and Logical Analysis of Data (LAD)

In this sub-section two Filter methods are presented, Rough Sets and Logical Analysis of Data (LAD). Both methods try to preserve the semantics of the data and the algorithms are composed of two-phases, the problem transformation and the problem optimization. Rough Sets algorithm transforms data into a discernibility matrix and obtains the result by solving a SAT optimization problem. LAD firstly converts data into a disjoint matrix and then applies the Set Covering problem to obtain the result.

Rough Sets

Rough Sets theory was initially proposed as a tool to reason about vagueness and uncertainty in information systems by Pawlak [1982] and later it was also proposed for attribute selection by Pawlak [1991]. The applications of the Rough Sets method are wide leading to meaningful results in many fields, such as conflict analysis, finance, industry, multimedia, medicine, and most recently bioinformatics [Polkowski 2002] [Peters, Skowron 2010]. A detailed example of attribute selection with Rough Sets can be found in Cavique et al. [2013].

In Rough Sets, table values can have any integer value rather than binary values. Inconsistencies occur when two cases have the same values for all attributes but belong to different decision classes. A practical example is two sick people that have the same symptoms but different diseases. With real data this is possible, because the table might have a missing attribute that could discriminate between them.

Rough Sets do not correct or exclude the inconsistencies, but for each class they determine a lower and an upper approximation. Given D={O,X∪C}, the subset of objects Y⊆O and the subset of attributes B⊆X, Pawlak's Rough Sets theory defines two approximation spaces: the lower and upper rough approximation. The lower approximation $B_L(Y)$ is the least composed set that is contained in Y, and the upper approximation $B^U(Y)$ is the greatest composed set that contains Y.

As a consequence the approximation space leads to $B_L(Y) \subseteq Y \subseteq B^U(Y)$. Also, the lower and upper approximations of a subset Y⊆O can be seen as operators in the universe of objects dividing it into three disjoint regions, the positive region POS(Y), the negative region NEG(Y) and the boundary region BR(Y): $POS(Y) = B_L(Y)$, $NEG(Y) = O - B^U(Y)$ and $BR(Y) = B^U(Y) - B_L(Y)$.

When the lower and upper approximations are equal, $B_L(Y)=B^U(Y)$, there are no inconsistencies and the rough set is called crispy rough set. Another way to identify the roughness of the set is using measures. The accuracy approximation measure is given by: $\alpha(Y) = \frac{|B_L(Y)|}{|B^U(Y)|}$ where |Y| denotes the cardinality of Y≠0 and $0 \leq \alpha(Y) \leq 1$. If $\alpha(Y)=1$, X is crisp; otherwise, it is a Rough Set.

The goal of Rough Sets is to discover decision rules from dataset D. The minimum number of attributes required to explain all the observations. Discovering the minimum number of attributes is an NP-hard problem. One of the following techniques is normally used: Reduction by Heuristics or Discernibility Matrix.

In Reduction by Heuristics, the search for a core is given by the following procedure: for each iteration, one attribute is removed, and the augmentation of inconsistency is verified. As already referred, inconsistency occurs when two or more observations have the same values for all attributes but belong to different decision classes. If the inconsistency does not increase, the attribute should be removed. When no further attributes can be removed, the remaining ones are considered indispensable and thus the core is found. Using a discernibility matrix of D, denoted by M, an (n×n) matrix is defined as follows, where M(i,j)=∅ denotes that this case does not need to be considered.

$$M(i,j) = \begin{cases} \{x \in X : x(o_i) \neq x(o_j)\} & if \quad c(o_i) \neq c(o_j) \\ \emptyset & otherwise \end{cases}$$

The discernibility matrix keeps the distinct attributes for each pair of observations belonging to different classes. Discernibility function F(B) is a Boolean function, written in the disjunctive normal form (DNF), that is a normalization of a logical formula which is a conjunction of disjunction clauses. F(B) determines the minimum subset of attributes that allows the differentiation of classes: F(B)= $\wedge\{\vee$ M(i,j): i, j=1,2, …, n; M(i,j) $\neq\emptyset\}$. The F(B) decision problem is equivalent to the Satisfiability problem (SAT), which was the first known example of an NP-complete problem.

Logic Analysis of Data (LAD)

The LAD method developed by Peter Hammer´s group [Crama et al. 1988], [Boros et al. 2000] refers to the discovery of the minimum number of attributes that are necessary to explain all observations and the detection of hidden patterns in a dataset with two classes.

The method works on binary data. Let D be the dataset of all observations, then each observation is described using several attributes, and each observation belongs to a class. An extension of the Boolean approach is needed when nominal non-binary attributes are used. The binarization (or discretization) of these attributes is performed by associating a string of Boolean variables to each attribute.

Dataset D is given as a D^+ set for 'positive' observations and as a set D^- set for 'negative' observations, where $D = D^+ \cup D^-$ and the sets are disjoint $D^+ \cap D^- = \emptyset$. Observations are classified as positive or negative based on a hidden function, and the goal of the LAD method is to approximate this hidden function with a union of intervals. In order to systematize the process, a disjoint matrix [a(i,j)] will be defined as:

$$a(i,j) = \begin{cases} 1 & \forall i : x_j(o_a) \neq x_j(o_b), C(o_a) \neq C(o_b), (o_a, o_b) \in O \text{x} O \\ 0 & otherwise \end{cases}$$

Given the disjoint matrix the Set Covering optimization problem is used to find the selected attributes.

2.3. Omics Datasets

Omics is a neologism associated with biology suffixes such as genomics, transcriptomics, proteomics, or metabolomics. Omics datasets are typically characterized by high dimensionality and small samples, that is, a large number of features and few observations. One prime example is data from DNA sequencing, especially those generated by the third generation, also called next-generation or high-throughput sequencing machines.

A typical genome differs from the reference human genome from 4.1 to 5.0 million variants. For large cohorts, the number of genetic variants is even higher. The 1000 Genomes Project sequenced the genomes of 2504 individuals from 26 populations producing a total of 88 million genetic variants [The 1000 Genomes Project Consortium, 2015].

Presently, the amount of sequencing data is doubling every seven months [Stephens et al. 2015] and, with the advent of new, high-throughput sequencing machines, the growth is expected to accelerate even more. When combined with the high dimensionality of the datasets, this growth makes the development of scalable and accurate feature selection methods even more essential.

Since biomedical genome sequence datasets are subject to restricted access, we generated artificial datasets with similar characteristics using SeqSIMLA [Yao, Chung 2016; Chung et al. 2014] (http://seqsimla.sourceforge.net/). SeqSIMLA generates simulated sequencing and phenotype data for case-control cohorts with allele frequencies and linkage disequilibrium patterns based on real datasets. The user can specify around 20 parameters, like disease prevalence or odds ratio for the marginal effects or interactions.

2.4. Heuristic Decomposition Approach

To solve a problem with hundreds of thousands of attributes, a Decomposition Heuristic is introduced. In Linear Programming optimization some approaches can be mentioned such as Column Generation, Dantzig-Wolfe decomposition and Benders decomposition. A detailed introduction can be found in [Boyd et al. 2008]. Decomposition is the act of breaking a large problem into sub-problems. Decomposition of a problem is possible if the structure of the problem is maintained. If problem A is separable into sub-problems A^1, A^2, …, A^{max_k} it can also run in a parallel computer environment.

Instead of minimizing the evaluation function f(A) the decomposition method minimizes the sub-problems $f(A^1)$, $f(A^2)$, ..., $f(A^{max\_k})$ and finally minimizes the master problem using function g(), which aggregates solutions of the sub-problems. Algorithm 1 presents the parallel version and Algorithm 2 the sequential version.

Algorithm 1: General Heuristic Decomposition (parallel version):
Input: sub-problems A^1, A^2,…, $A^{1..max\_k}$
Output: solution of the master-problem algorithm
1. solve the sub-problems:
   sub-problem 1: y_1=minimize f(A^1)
   sub-problem 2: y_2=minimize f(A^2)
   ...
   sub-problem k: y_{max_k}=minimize f(A^{max_k})
2. solve the master-problem:
   master = minimize g(y_1, y_2,…, y_{max_k})

In the sequential version of the heuristic decomposition, Algorithm 2, updates the master-problem at each iteration by adding additional information from the solved sub-problem.

Algorithm 2: General Heuristic Decomposition (sequential version):
Input: sub-problems A^1, A^2,…, $A^{1..max\_k}$
Output: solution of the master-problem algorithm
1. for k=1 to max_k
1.1. solve sub-problem: y_k = minimize f(A^k)
1.2. solve master-problem: master = minimize g(master, y_k)
2. end for

The decomposition principle is applied in exact methods and in metaheuristic methods [Joncour et al. 2010]. Given the large number of attributes, a heuristic decomposition approach, coined by [Smet et al. 2016] is used in this work.

In Section 3 and Section 4, we detail the sub-problem and the master problem algorithms for the Logical Analysis of Inconsistent Data to deal with large datasets.

3. Sub-problem Algorithm

As already stated, in this work we propose a feature selection technique to deal with inconsistent data in large datasets. The inconsistent data is processed using the LAID algorithm. The sub-problem algorithm returns two criteria, the information about the quality of the filter (number of features) and the prediction quality (accuracy). Thus, the output of this filter method is the pair (number of features, accuracy).

To solve the sub-problem, in Algorithm 3, we describe the LAID Algorithm for the bi-criteria feature selection. Firstly, to remove any inconsistency, we add a dummy binary variable named “je ne sais quoi”, ‘jnsq’, since two observations with the same attributes that belong to different classes work against the consistency measure. In a second step, the Disjoint Matrix Generation [$A_{i,j}$] and Cost Vector [c_j] are created, where both are based on the definition of the distance measure. In the third step, a Heuristic for the Set Covering problem is applied, returning the reduced set of features. In the fourth step, a cross-validation method is applied to retrieve the accuracy of the solution. Finally, the

algorithm returns the pair (number of features, accuracy). The algorithm can be specified as follows:

Algorithm 3: Logical Analysis of Inconsistent Data
Input: dataset D = {O, X∪C} with binary variables
Output: (number of features, accuracy)
1. check data inconsistencies and add dummy variable 'jnsq' as a discriminant feature
2. disjoint matrix generation $[A_{i,j}]$ and cost vector $[c_j]$
3. number of features = Minimum Set Covering Problem
4. accuracy = Cross-validation

To exemplify the different steps of the algorithm a running example with six features and five observations is used.

3.1. Data inconsistencies

Given the inconsistency of two observations O_a and O_b with the same values for all the attributes $X(O_a)=X(O_b)$ and belonging to different classes, a dummy binary variable "je ne sais quoi" [Cavique et al. 2013] is added, assigning a value of 1 whenever the class=1, or:

$$\text{jnsq}_a = \begin{cases} 1 & \text{if } (X(O_a) = X(O_b)) \wedge \big(\text{class } (O_a) \neq \text{class } (O_b)\big) \wedge (\text{class}(O_a) = 1) \\ 0 & \text{otherwise} \end{cases}$$

Instead of using the complex approximations of the Rough Sets keeping the data inconsistencies, LAID does not exclude the inconsistency by adding a dummy variable that allows the subsequent application of the other steps of the LAD method.

Given the dataset with six features $\{X_1, \ldots, X_6\}$ and five observations $\{O_1, \ldots, O_5\}$, the first step of LAID is verifying the existence of inconsistencies in the dataset. For each inconsistency the dummy variable 'jnsq' is assigned '1' in class 1.

In Table 1 the dummy variable 'jnsq' is added in the last feature column to avoid data inconsistencies between observation O2 and O3, which present the same values for all the features.

Table 1. Variable 'je ne sais quoi' is added

observation\feature	**X_1**	**X_2**	**X_3**	**X_4**	**X_5**	**X_6**	**X_7 (jnsq)**	**Class**
O_1	1	1	0	1	1	1	0	0
O_2	1	0	1	1	1	0	0	0
O_3	1	0	1	1	1	0	1	1
O_4	1	0	1	0	0	1	0	1
O_5	0	0	1	0	0	0	0	1

3.2. Disjoint Matrix Generation [$A_{i,j}$] and Cost Vector [c_j]

In the second step the Cost Vector [c_j] and Disjoint Constraints Matrix Generation [$A_{i,j}$] are created. Both, the Disjoint Matrix Generation [$A_{i,j}$] and Cost Vector [c_j] use the concept of distance measure for independent criteria in feature selection.

The Cost Vector [c_j] is obtained by adding the pair comparisons within the same class. This calculation is used to measure the general incoherence, or noise of feature j.

On the other hand, the Disjoint Matrix [$A_{i,j}$] is obtained by using the pair comparisons belonging to different classes. The disjoint matrix [$A_{i,j}$] for each attribute X_j and pair of observations (O_a, O_b), is defined as:

$$A_{i,j} = \begin{cases} 1 & \forall i:\ X_j(O_a) \neq X_j(O_b),\ \text{class}(O_a) \neq \text{class}(O_b),\ (O_a, O_b) \in O \times O \\ 0 & \text{otherwise} \end{cases}$$

The dimension of index *i* in matrix [$A_{i,j}$] depends on the constraint structure of dataset D. Each constraint results from the comparison of two different arbitrary observations O_a and O_b that belong to distinct classes. If one attribute *j* is different in the observations O_a and O_b the value of $A_{i,j}$ becomes value 1, denoting that at least one column (attribute) *j* must be maintained in order to differentiate the rows (or constraints) *i*.

The cost vector [c_j] and the disjoint matrix [$A_{i,j}$] will be used as input in the Minimum Set Covering problem, where all constraints (or lines *i*) must be covered, at least once by some attributes.

Table 2 presents pairs of observations of the same class. For each observation the features of the same class should be equal to avoid disagreement. The sum of the disagreement of each feature is called cost and stored in the Cost Vector [c_j]. As stated before, for observations O_a, O_b and attribute X_j, the value of disagreement is given by the operator XOR that returns True if (O_a, X_j) ≠ (O_b, X_j) and False otherwise.

Table 2. Cost vector [c_j] shows the disagreement within the same class

Pairs same class	**X_1**	**X_2**	**X_3**	**X_4**	**X_5**	**X_6**	**X_7 (jnsq)**	**Class**
O_1,O_2		1	1			1		1
O_3,O_4				1	1	1	1	0
O_3,O_5	1			1	1		1	0
O_4,O_5	1					1		0
cost [c_j]	2	1	1	2	2	3	2	

To obtain the disjoint matrix [$A_{i,j}$], for each pair of observations, the features of different classes should also be different in order to be able to discriminate them. The number of lines generated in [$A_{i,j}$] is N.M, where N and M are the number of observations of class 0 and class 1, respectively. The value k_j corresponds to the number of lines covered for each feature j.

Table 3 presents the pairs of observations that belong to different classes, where the logic operator XOR can also be applied, since, if $(O_a, X_j) \neq (O_b, X_j)$, the feature X_j should be chosen because it distinguishes the two classes.

Table 3. Matrix $[A_{i,j}]$ shows the disjoint between different classes

Pairs different classes	X_1	X_2	X_3	X_4	X_5	X_6	X_7 **(jnsq)**
1 - O_1,O_3		1	1			1	1
2 - O_1,O_4		1	1	1	1		
3 - O_1,O_5	1	1	1	1	1	1	
4 - O_2,O_3							1
5 - O_2,O_4				1	1	1	
6 - O_2,O_5	1			1	1		
k_j	2	3	3	4	4	3	2

In this running example the unitary cost of each column is $f(c_j, k_j) = c_j/k_j$. In Table 4 the unitary cost $f(c_j, k_j)$ is shown.

Table 4. Unitary cost of each column $f(c_j, k_j)$

Features	X_1	X_2	X_3	X_4	X_5	X_6	X_7 **(jnsq)**
c_j	2	1	1	2	2	3	2
k_j	2	3	3	4	4	3	2
c_j/k_j	1	1/3	1/3	1/2	1/2	1	1

In this example two classes were used. However, the method can be extended to multiple classes. In this extension, observations of different classes should be compared in the generation of the disjoint matrix $[A_{i,j}]$.

3.3. Minimum Set Covering Problem

In the third step of LAID a heuristic is applied to solve the Set Covering Problem. Given matrix $[A_{i,j}]$ and variables y_j that denote which columns/features are selected, the optimization problem that finds the minimum number of columns/features covering all the rows is the Set Covering problem and can be defined in binary linear programming as:

$$\text{Minimize } f = \sum c_j . y_j$$

$$\text{Subject to } \sum A_{i,j} . y_j \geq 1$$

$$\text{and } y_j \in \{0,1\} \qquad j=1,\ldots,m$$

In this sub-section, a greedy heuristic approach is used to solve the Minimum Set Covering problem, which reuses the algorithm proposed in [Chvatal 1979]. Algorithm 4 is composed of two phases, a constructive phase that adds columns to cover all the lines, and in a second phase removes the redundant columns.

Algorithm 4: Set Covering Problem
Input: sub-matrix $[A_{i,j}]$ and vector $[c_j]$
Output: a subset of attributes with minimum cost
-- Constructive Phase
1. Initialize R = $[A_{i,j}]$, S=Ø
2. While R ≠ Ø do
2.1. Choose the best line i*∈R such that $|A_{i^*,j})|=\min |A_{i,j}|$, ∀j
2.2. Choose the best column j* that covers line i*, considering $f(c_j, k_j)$
2.3. Update R and S, $R=R\backslash A_{i,j^*}$, ∀i, $S=S\cup\{j^*\}$
4. End while
-- Remove redundant columns
5. Sort cover S by descending order of costs
6. For each S_i do if ($S\backslash S_i$ is still a cover) then $S=S\backslash S_i$
7. Return S

In the Constructive Phase the algorithm chooses line i* with the fewest elements (2.1). Next it selects column j* that covers line i*, with the lowest cost considering $f(c_j, k_j)$ (2.2). In the first iteration from features {X_1, X_2, X_3, X_4, X_5, X_6, X_7} solution {X_2} is obtained with cost=3. This solution is not admissible since there are 3 uncovered lines. In the second iteration X_4 is added to the solution, with a global cost of 7, where one covered line is missing. In the third iteration the admissible solution {X_2, X_4, X_7} is found, with cost 9. The constructive phase ends and the redundancies removal phase occurs, where a new admissible solution {X_4, X_7} is found, with cost 6.

Given the subset {X_4, X_7}, in Table 5 the reduced dataset is presented. The number of features is reduced from 7 to 2 and the number of observations is also reduced from 5 to 3. Nearly all the combinations with 2 features are presented, except for X_4=0 and X_7=1 because no observations occur.

Table 5. Reduced dataset

observation\feature	X_4	X_7 (jnsq)	Class
O_1, O_2	1	0	0
O_3	1	1	1
O_4, O_5	0	0	1
no observations	0	1	?

3.4. Leave-one-out cross-validation

The major method for supervised learning model validation is cross-validation. Cross-validation technique involves the partition of the sample dataset into *n* subsamples, using (*n*-1) for repeated training and the remaining for testing. In this work, we adopt the special case of Leave-One-Out cross-validation, which consists in removing one observation from the original sample, training the algorithm with the remaining observations and then, testing the observation using the resulting model. The input file

for the Leave-One-Out cross-validation is the reduced dataset and not the original one, which results in a reduced computational effort.

To compare observation *i* with the testing observation *x*, we use the Hamming distance, H(i,x). The Hamming distance between the known observation O_1=(1,1,0,0) and testing observation x=(1,0,1,0) is H(1,x)=2, since there are two differences, namely in the second and third elements.

In the predicting process, the original LAD algorithm that compares observations belonging to two disjoint classes uses the so-called discriminant function, which is similar to the k-nearest neighbor classification algorithm.

In the k-nearest neighbor prediction algorithm, k observations are chosen which present the lowest Hamming distance in comparison with the testing observation.

As the class of the observation is known, we can count the number of observations of class 0 and class 1. The prediction of the x class is given by the most frequent class, which in statistics is the mode. Therefore, the prediction class of x is the mode of the classes of the k observations:

predicted_class(x) = mode (class(O_1), class(O_2), …, class(O_k))

The performance of a classification model is based on the total counts of correct and incorrect predictions. These counts are tabulated in a table known as Confusion Matrix. The Confusion Matrix is a specific square table, actual class versus predicted class, which allows the visualization of the performance of an algorithm. Based on the Confusion Matrix, many performance measures can be extracted. In this work the accuracy or hit-rate measure is used, expressed by:

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}$$

The sub-problem algorithm returns the pair (number of features, accuracy). The number of features is given by Algorithm 4, the Set Covering Problem, and the accuracy by the Leave-one-out cross-validation. This sub-problem algorithm is used by the Heuristic Decomposition presented in the next section.

4. Heuristic Decomposition and Master-problem Algorithm

In many Linear Programming optimization problems, the constraints and variables may be decomposed in blocks, where each constraint set only involves a variable set, with the following structure:

$$\begin{aligned} \text{Minimize} \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ \text{Subject to} \quad & x_1 + x_2 \geq 1 \\ & x_3 + x_4 \geq 1 \\ & x_5 + x_6 \geq 1 \end{aligned}$$

As our problem is not so regular, we must resort to a Heuristic Decomposition for the Set Covering problem. The matrix $[A_{i,j}]$ is divided into max_k sub-problems resulting in max_k sub-matrices $[A_{i,j}]^k$. The structure of the problem is maintained since each sub-problem returns a feasible (or admissible) solution. Comparing this approach with meta-heuristics the sub-matrices correspond to different neighborhoods in Variable Neighborhood Search or to new regions in the diversification process of Tabu Search, where feasible solutions can be found.

Each sub-problem returns the pair (number of features, accuracy) for the bi-criteria feature selection, where the goal is to minimize the number of features and to maximize the accuracy.

4.1. Heuristic Decomposition

The original problem with one million features is very hard to solve, given its dimensionality. In the decomposition each sub-problem should return an admissible solution, that is, all lines should be covered by the chosen features/columns. Given the dimension of each sub-problem the value of max_k can be found.

As each sub-problem returns a feasible solution, the problem is perfectly separable or independent. A parallel version of the Heuristic Decomposition for the bi-criteria feature selection is implemented. Algorithm 5 reuses the LAID procedure, described in Algorithm 4, and we consider the following notation:
nf – number of features
ar – accuracy

Algorithm 5: Heuristic Decomposition for the Bi-criteria Feature Selection
Input: sub-problems $A[i,j]^{1..max\_k}$
Output: Pareto front with pairs (nf, ar)
1. solve the sub-problems:
   sub-problem 1: $(nf^1, ar^1) = LAID\ (A[i,j]^1)$
   sub-problem 2: $(nf^2, ar^2) = LAID\ (A[i,j]^2)$
   ...
   sub-problem max_k: $(nf^{max\_k}, ar^{max\_k}) = LAID\ (A[i,j]^{max\_k})$
2. solve the master-problem:
   find Pareto optimal set $((nf^1, ar^1), (nf^2, ar^2), \ldots, (nf^{max\_k}, ar^{max\_k}))$

In our Heuristic Decomposition for feature selection, using parallel processing, the matrix A[i,j] is divided into max_k sub-matrices. Feasible sub-solutions are returned by the LAID procedure with the metrics, number of features and accuracy. Each pair (number of features, accuracy) corresponds to a point in a bi-objective optimization problem. To find the Pareto optimal set, the master problem bi-criteria decision making is detailed in sub-section 4.2.

4.2. Bi-objective Master-problem Algorithm

Multi-objective optimization (MOO), or multi-criteria optimization, deals with the optimization of two or more conflicting objectives, subject to a set of constraints.

Given the vector $x=(x_1,\ldots, x_m)$ of decision variables, *M* is the number of objectives and *J* the number of constrains, multi-objective optimization can be stated as follows [Collette, Siarry 2011].

Minimize / Maximize $f_m(x)$, $m = 2,\ldots, M$ objectives
Subject to $c_j(x)$ $\{\leq, =, \geq\}$ 0 $j = 1, 2,\ldots, J$ constrains
and $x_i \geq 0$

In multi-objective optimization more than one optimal solution can be obtained, whereas classic optimization has only one objective. Variable S represents the set of feasible solutions associated with equality and inequality constraints. $F(x) =(f_1(x), f_2(x), ..., f_m(x))$ is the vector of objectives to be optimized.

In multi-objective optimization the dominance concept is central. In the maximization problem, the objective vector $u=(u_1, \ldots, u_m)$ dominates $v=(v_1, \ldots, v_m)$, denoted $u > v$, if and only if, $u_i \geq v_i$: $\forall i$, and at least one component v is smaller, $u_i > v_i$: $\exists i$.

A solution is non-dominated, or Pareto solution, if and only if, there is no solution that dominates it. In other words, a solution $x^* \in S$ is Pareto optimal if for every $x \in S$, F(x) does not dominate $F(x^*)$.

The Pareto optimal set, P^*, includes all the Pareto solutions, i.e., the set of all solutions whose associated vectors are non-dominated.

When plotted in space, non-dominated vectors are collectively known as the Pareto front, PF^*. The procedure to generate Pareto front is to compute as many points as possible and then build a surface that includes those points. The surface created by the Pareto front can be linear, convex or concave. The Pareto front is defined as $PF^* = \{F(x), x \in P^*\}$.

The ideal vector contains the best solutions considering the *m* objectives separately, at the same point. A point $y^* = (y^*_1, y^*_2, ..., y^*_m)$ is an ideal vector if it optimizes each objective function f_i in F(x).

Given a large number of feasible solutions in MOO problems, it is important to differentiate two stages: the optimization of the objective functions and the decision-making process. Three strategies are commonly used: (i) *a priori*, making decisions before optimizing; (ii) *a posteriori*, optimizing before making decisions and (iii) compromising between optimizing and decision making.

In this work, given the criteria number of features and accuracy, the MOO is reduced to a bi-objective optimization problem. Our approach is formulated as a Bi-objective Feature Selection, such that the aim is to:

- minimize the number of features (f1) and
- maximize the accuracy of the reduced set of features (f2).

Figure 1, adapted from [Talbi 2009], shows a bi-objective optimization. The black circles of the Pareto solution dominate the solutions represented by triangles. The efficient front is given by the curve that includes all the Pareto solutions. In this case we want to minimize f_1 and maximize f_2.

The ideal solution is obtained by the combination of the solution that minimizes f_1 with a second solution that maximizes f_2.

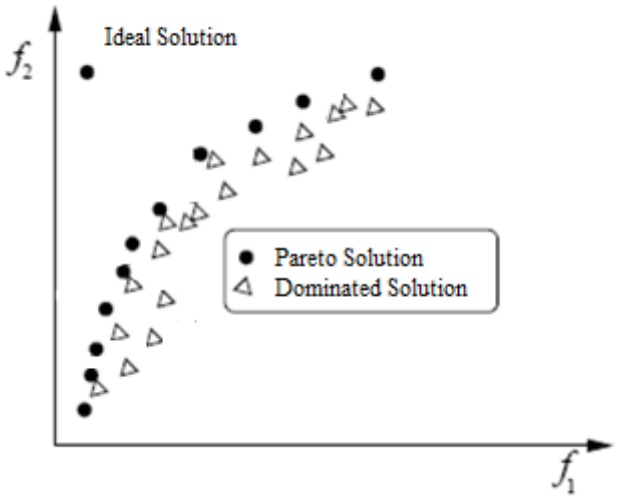

Figure 1. Pareto set to minimize f_1 (number of features) and maximize f_2 (accuracy)

After the generation of the bi-objective sub-solutions generated by the sub-problem, the master problem chooses the non-dominant solutions and includes them in the Pareto optimal set. The Pareto set contains only the solutions with the least number of features and the best accuracy, which substantially reduces the number of bi-criteria solutions that will be analyzed by the decision maker, as in this case, the omics dataset analyst.

Regarding the optimization and decision-making stages, we select the *a posteriori* strategy, which does not require previous information from the decision maker.

4.3. Parallel processing for heuristic decomposition and MOO

The rapid increase of powerful processors and fast networks has enabled the emergence of computer clusters, large-scale network of machines (grids) and platforms for high-performance computing (HPC). In terms of designing parallel heuristics, three major parallel levels are identified: the algorithmic level, the interaction level and the solution level [Talbi 2009].

- In the algorithm level, different instances of the heuristic can run independently in different processors, and the search is equivalent to the sequential execution. The sub-problem is independent, and the sub-solutions are not shared in the processors. The master problem collects the sub-solutions returned by the sub-problems.

- In the interaction level solutions, the sub-problems are independent, although the master problem reuses the sub-solutions by sharing them with the different processors. The main objective of this level is to speed up the algorithm by reducing the search time. This approach can be applied in population-based heuristics.

- Finally, at the solution level, the problem is dependent, i.e., not separable, and so processors return partial (or non-feasible) sub-solutions. The master algorithm should ensure the feasibility of the final solution.

Regarding the parallel levels, we opt for the algorithm level because only feasible sub-solutions are returned from the sub-problems.

In this work, parallel processing perfectly matches large volumes of data and decomposition techniques applied to feature selection objectives and MOO.

To deal with large volumes of omics datasets, the decomposition is a strategy that combines with parallel processing. Large omics datasets are parceled out for the heuristic decomposition approach. Then the heuristic sub-problems can run in parallel resulting in a sequence schematized as follows:

large omics datasets → heuristic decomposition → parallel processing

On the other hand, the feature selection with two objectives influences the MOO approach. MOO collects multiple solutions, which perfectly combine with parallel processing that also deals with many solutions simultaneously. Given the pair (number of feature, accuracy) MOO can be applied. Then MOO can run in parallel processing resulting in the following sequence:

pair (number of feature, accuracy) → MOO → parallel processing

Parallel processing integrates the two concepts, computing multiple solutions for the decomposed heuristic and for MOO. The result is unified in the Pareto optimal set computed by the master algorithm.

5. Computational results

To implement the computational results of this algorithm, some choices such as the computational environment, the datasets and the performance measures must be made.

The computer programs were written in C language and the GCC/Dev-C++ compiler was used. The computational results were obtained from an Intel Core Duo CPU 3.0 GHz processor with 4.0 GB of main memory running under the Windows 10 operating system. The INCD (National Infrastructure for Distributed Computing, Portugal) computer cluster was used to run the LAID algorithm in parallel.

As already stated, since omics datasets with rare diseases information are subject to restricted access, we generated artificial datasets with similar characteristics using SeqSIMLA. The dataset under study is comprised of 1700 observations in the affected group, with class 1, associated with rare disease patients and 300 observations in the control group, with class 0, referring to healthy people. The disease prevalence of 0.015 was chosen. Reference haplotypes and recombination rates from chromosome 1 of the European population of the 1000 Genomes Project were used for data generation [Chung et al. 2014]. Initially 30 variants were chosen as disease sites, of which 19 had allele frequencies larger than 0 (ranging from 0.0001 to 0.447). We use a fixed value of odds ratio of 2.0 for the marginal association with the phenotype for all variants and all non-variant features were removed. In the original dataset the number of attributes is close to one million. The number of patient observations, O_a, is 1700 and the number of observations of controls, O_b, is 300. The generation of matrix $[A_{i,j}]$, for each observation O_a is compared with all the observations O_b, resulting in a number of lines $O_a \times O_b$, as exemplified in Table 3. The number of lines in the matrix $[A_{i,j}]$ is 510,000 (1700 x 300) and the final dimension of the matrix is 510,000 x 1,000,000.

Two performance measures are used to evaluate results, the computational time and quality of the solutions. The quality solution is given by two criteria, the minimum number of attributes that better explains the dataset and the accuracy of the reduced dataset.

5.1. Computational runtime

To study the performance of large matrices the time complexity should be considered to estimate the total runtime.

For a two-class problem the number of observations is given by the tuple (O_0, O_1); (100, 100), (500, 500), (700, 700) and (1700, 300). The number of columns is tested from 100 to 5000 variables.

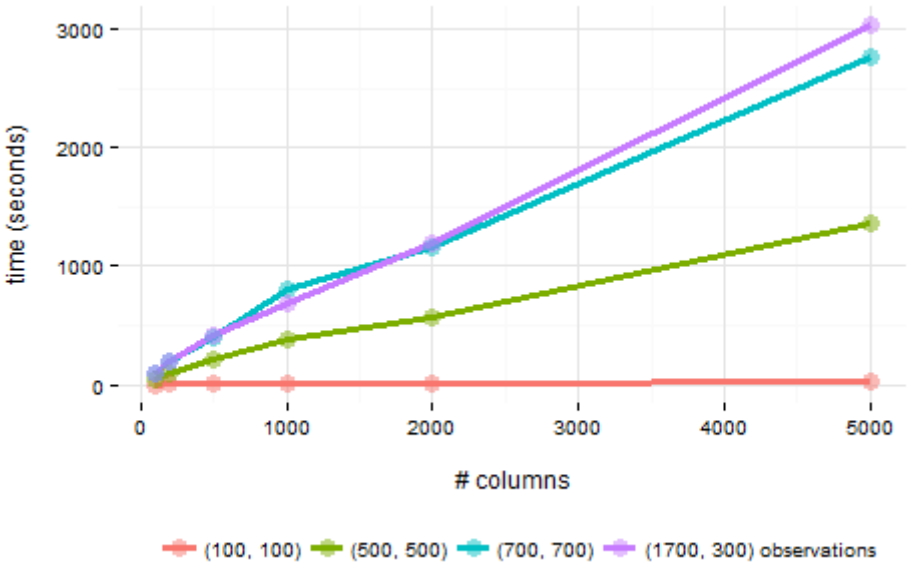

Figure 2. Computational time varying the number of columns and observations

In the Decomposition Heuristic for the Set Covering problem, for each sub-matrix $[A_{i,j}]^k$, the computational time is shown in Figure 2, where a linear behavior can be found between runtime and the number of columns.

The total runtime for matrix $[A_{i,j}]$ with dimension 510,000 x 1,000,000, and k=200, that is 200 sub-matrices $[A_{i,j}]^k$ of 510,000 x 5,000 each, can be estimated in 570,800 seconds, or approximately seven days. Since it is possible to decompose the problem, it can run in a computer parallel environment. In the implementation we distributed the task using 10 machines, resulting in a runtime of 58,000 seconds, or approximately 16 hours.

The speedup S_N is defined as the time T_1 taken to complete a program with one processor divided by the time T_N taken to complete the same program with N processors.

$$S_N = \frac{T_1}{T_N}$$

The efficiency E_N using N processors is defined as the speedup S_N divided by the number of processors N.

$$E_N = \frac{S_N}{N}$$

In our work we opt for the algorithm level, where instances of the heuristic can run independently using different processors, and speedup is linear, with an efficiency E_N of 100%, meaning all the processors are being fully used all the time.

5.2. Quality of the solutions

The challenge of this work is to solve a feature selection problem with a dataset involving one million attributes. In the computational experiments performed with the sub-dataset with less than 1000 columns, we verified that the admissibility of the solutions is found by merging 3 sub-matrices $[A_{i,j}]^k$. For sub-datasets with 5000

columns, feasible solutions are found for each sub-matrix. Running 200 times, hundreds of possible good solutions are found and then used in the master problem phase. Keeping this in mind, we add a new criterion, the accuracy.

As previously stated, the solution quality is given by two objectives, the number of attributes of the feasible reduced set and the accuracy in the cross-validation. The number of attributes is given by the Set Covering heuristic where there are no parameters to point out. To obtain the accuracy in the Leave-One-Out cross-validation, we use k=3 in the k-nearest neighbor prediction algorithm. The Pareto optimal set is the result of the master-problem algorithm.

Figure 3 represents the Pareto front in a chart with axis, number of features and accuracy. The Pareto front contains the set of efficient points. However, in this case only point (19, 0.84) belongs to the efficient set, all the other points are dominated by this one. After analyzing the results three solutions are found with the performance of (19, 0.84).

The feature selection was accomplished given the reduction from 5000 features to 19 features, which is quite expressive.

A scale for classifying the accuracy is: excellent (0.91-1.00), good (0.81-0.90), fair (0.71-0.80), poor (0.61-0.70) and fail (0.51-0.60). So, the accuracy found of 0.84 is considered not excellent but good.

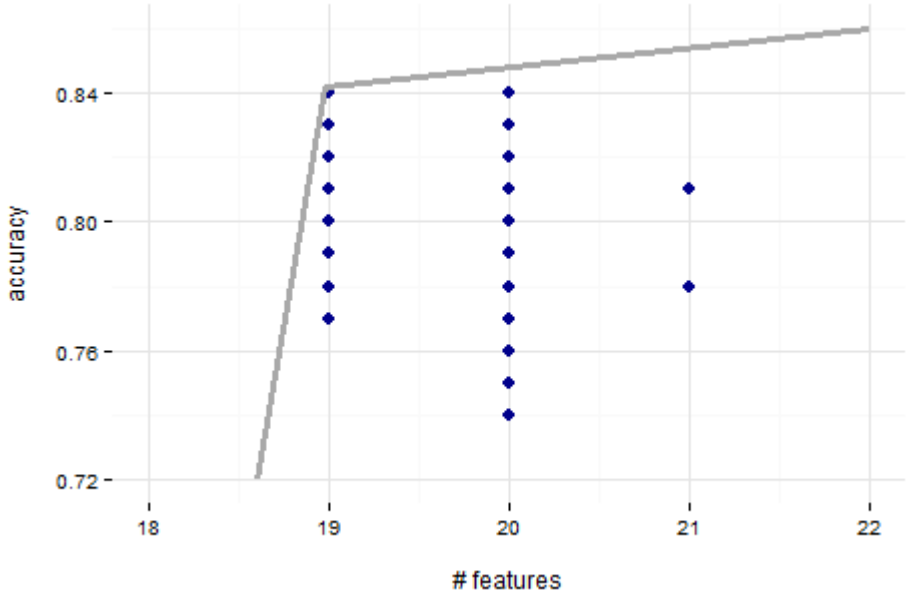

Figure 3. Pareto front of the objectives, accuracy and number of features

Definitive quality assessment of the sub-dataset must be evaluated by human specialists on rare diseases. A huge reduction is attained, from one million features to three reduced sub-datasets with 19 features each.

5.3. Final remarks of the computational experiment

The final remark of this study regarding the large volumes of data can be summarized considering three aspects: the data, the algorithm and the environment.

Regarding the data, large datasets with many dimensions are becoming the new normal, where the in-memory data access should be replaced by the on-disk data access. The Hierarchical Data Format (HDF5), originally developed at the National Center for Supercomputing Applications, is designed to store and organize large amounts of data that can be accessed on-disk in multiple ways.

The new algorithms need low time complexity and problem decomposition is essential to parallelize the problem. The intense use of computational expensive algorithms, like meta-heuristics, should be reduced, giving way to simple heuristics. Furthermore, with meta-heuristics the setting of multiple parameters is nontrivial, while with heuristics the number of parameters is substantially reduced.

Finally, the desired environment is in the cloud, running the programs in parallel in High-Performance Computing. The European research roadmap High-Performance Computing 2030 [European Commission 2018] should open new perspectives.

6. Conclusions

Given the large datasets available in bioinformatics, the aim of this work is to present a feature selection method able to deal with thousands of observations and millions of attributes.

In feature selection, two performance measures are considered: the predictive accuracy and result comprehensibility using the parsimony principle. To preserve the data semantics, the minimum number of attributes that better explain the dataset is found, achieving the maximal actionable knowledge. On the one hand, accuracy is essential to measure the quality of the solution. On the other hand, the consistency measure is a core issue in the result comprehensibility. In this work, we opt for Filter methods for two main reasons, the large volumes of data that should be computed and the data semantics preservation. This document extends the previous work of the LAID algorithm to large datasets, using a bi-objective approach that includes the two performance measures.

The sub-problem algorithm returns the performance metric (number of features, accuracy). We present LAID using four steps. Initially, the consistency is assured by the inclusion of the dummy variable “je ne sais quoi”. In the set covering problem, the disjoint matrix $[A_{i,j}]$ and cost vector $[c_j]$ reuse the distance measure concept for the independent criteria in feature selection. Therefore, the algorithm combines consistency measure and the distance measure approaches. To compute the accuracy the Leave-One-Out cross-validation is used.

In the Decomposition Heuristic algorithm, the disjoint matrix is divided into sub-matrices $[A_{i,j}]^{max\_k}$. Each sub-problem algorithm returns a pair of measures, which

allows the bi-objective optimization. The algorithm finds multiple feasible solutions that are gathered in the master phase. In the master problem a Pareto optimal set is found selecting the subset of non-dominant solutions and reducing the number of feasible sub-solutions, which facilitate the work of the end user.

The parallel processing integrates two concepts, computing multiple solutions for the decomposed heuristic and for MOO. The parallel techniques work perfectly to decompose the large volumes of omics datasets. And the parallel approach also combines with the two objectives of feature selection and the diverse sub-solutions of the bi-objective optimization.

The computational runtime for the feature selection with dimension 2,000 observations and 1,000,000 features, running in parallel, seems suitable for user requirements. In the experiment the Pareto optimal set includes only one point, where 19 features are chosen, and the accuracy achieves the percentage of 84%.

In future work, following the stabilization of the requirements of the end users, we intend to compare our performance measures with other algorithms.

Acknowledgements
The authors would like to thank the FCT support UID/Multi/04046/2013. This work used the EGI, European Grid Infrastructure, with the support of the IBERGRID, Iberian Grid Infrastructure, and INCD (Portugal).

REFERENCES

1. Almuallim, H., Dietterich, T.G. (1991). Learning with many irrelevant features. In: Proceedings of the 9th National Conference on Artificial Intelligence, MIT Press, pp. 547-552.
2. Boros, E., Hammer, P.L., Ibaraki, T., Kogan, A., Mayoraz, E., Muchnik, I. (2000). An Implementation of Logical Analysis of Data. IEEE Transactions on Knowledge and Data Engineering, vol. 12(2), pp. 292-306.
3. Boyd, S., Xiao, L., Mutapcic, A., Mattingley, J. (2008). Notes on decomposition methods. Notes for EE364B, Stanford University, pp. 1-36.
4. Cavique, L., Mendes, A.B., Funk, M. (2011). Logical analysis of inconsistent data (LAID) for a paremiologic study. In: Processing 15th Portuguese Conference on Artificial Inteligence, EPIA.
5. Cavique, L., Mendes, A.B., Funk, M., Santos, J.M.A. (2013). A feature selection approach in the study of azorean proverbs. In: Exploring Innovative and Successful Applications of Soft Computing, Advances in Computational Intelligence and Robotics (ACIR) Book Series, IGI Global, pp. 38-58.
6. Cavique L., Mendes A.B., Martiniano H.F.M.C. (2017). A Feature Selection Algorithm Based on Heuristic Decomposition, In: Oliveira E., Gama J., Vale Z., Lopes Cardoso H. (eds) Progress in Artificial Intelligence, EPIA 2017, Porto, Portugal, Lecture Notes in Computer Science, Springer, vol. 10423, pp.525-536.
7. Chandrashekar, G., Sahin, F. (2014). A survey on feature selection methods. Computers and Electrical Engineering, vol. 40(1), pp. 16-28.
8. Chvatal, V. (1979). A greedy heuristic for the set-covering problem. Mathematics of Operations Research, vol. 4, pp. 233-235.

9. Chung R.H., Tsai W.Y., Hsieh C.H., Hung K.Y., Hsiung C.A., Hauser E.R. (2014). SeqSIMLA2: Simulating Correlated Quantitative Traits Accounting for Shared Environmental Effects in User-Specified Pedigree Structure. Genetic Epidemiology. vol. 39(1) pp.20-24.
10. Collette Y., P. Siarry (2011). Multiobjective Optimization, Principles and Case Studies, Decision Engineering Series, Springer.
11. Crama, Y., Hammer, P. L., Ibaraki, T. (1988). Cause-effect relationships and partially defined Boolean functions. Annals of Operations Research, vol. 16, pp. 299-326.
12. European Commission (2018). The European declaration on High-Performance Computing, url: https://ec.europa.eu/digital-single-market/en/news/european-declaration-high-performance-computing
13. John, G.H., Kohavi, R., Pfleger. K. (1994). Irrelevant Features and the Subset Selection Problem. In: Proceedings of the 11th International Conference on Machine Learning, ICML 94, pp. 121-129.
14. Joncour, C., Michel. S., Sadykov, R., Sverdlov, D., Vanderbeck, F. (2010). Column generation based primal heuristics. Electronic Notes in Discrete Mathematics, Elsevier, vol. 36, pp. 695–702.
15. Kira, K., Rendell, L.A. (1992). The feature selection problem: traditional methods and a new algorithm. In: Proceedings of 9th National Conference on Artificial Intelligence, pp. 129-134.
16. Liu, H., Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering, vol.17, vol. 4, pp. 491-502.
17. Pawlak, Z. (1982). Rough Sets. International Journal of Computer and Information Science, vol. 1, pp. 341-356.
18. Pawlak, Z.: Rough Sets (1991). Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Boston.
19. Peters, J.F. & Skowron A. (2010). *Transactions on Rough Sets XI*. Lecture Notes in Computer Science / Transactions on Rough Sets, Springer.
20. Polkowski, L. (2002). *Rough Sets, Mathematical Foundations.* Advances in Soft Computing, Physica-Verlag Heidelberg, Germany.
21. Smet P., Ernst, A., Vanden Berghe, G. (2016). Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. Computers and Operations Research, vol. 76, pp. 60-72.
22. Stephens Z.D., S.Y. Lee, F. Faghri, R.H. Campbell, C. Zhai, M.J. Efron, R. Iyer, M.C. Schatz, S. Sinha, G.E. Robinson (2015). Big Data, Astronomical or Genomical?, PLoS Biology vol. 13(7): e1002195, doi:10.1371/ journal.pbio.1002195.
23. Talbi E.G. (2009). Metaheuristics, From Design to Implementation, John Wiley & Sons, Inc.
24. The 1000 Genomes Project Consortium (2015). A global reference for human genetic variation, Nature, vol. 526, pp.68-74.
25. Yao PJ, Chung RH (2016). SeqSIMLA2_exact, simulate multiple disease sites in large pedigrees with given disease status for diseases with low prevalence, Bioinformatics, vol. 32(4), pp. 557-562.