

**Titre:** Developed Algorithms for Maximum Pattern Generation in Logical  
Title: Analysis of Data

**Auteur:** Sara Tagarian  
Author:

**Date:** 2016

**Type:** Mémoire ou thèse / Dissertation or Thesis

**Référence:** Tagarian, S. (2016). Developed Algorithms for Maximum Pattern Generation in  
Citation: Logical Analysis of Data [Mémoire de maîtrise, École Polytechnique de Montréal].  
PolyPublie. <https://publications.polymtl.ca/2378/>

 **Document en libre accès dans PolyPublie**  
Open Access document in PolyPublie

**URL de PolyPublie:** <https://publications.polymtl.ca/2378/>  
PolyPublie URL:

**Directeurs de  
recherche:** Soumaya Yacout, & Hany Osman  
Advisors:

**Programme:** Maîtrise recherche en génie industriel  
Program:

UNIVERSITÉ DE MONTRÉAL

DEVELOPED ALGORITHMS FOR MAXIMUM PATTERN GENERATION  
IN LOGICAL ANALYSIS OF DATA

SARA TAGARIAN

DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL  
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION  
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES  
(GÉNIE INDUSTRIEL)

DÉCEMBRE 2016

© Sara Tagarian, 2016.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

DEVELOPED ALGORITHMS FOR MAXIMUM PATTERN GENERATION  
IN LOGICAL ANALYSIS OF DATA

présenté par : TAGARIAN Sara

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. ADJENGUE Luc-Désiré, Ph. D., président

Mme YACOUT Soumaya, D. Sc., membre et directeur de recherche

M. OSMAN Hany, Ph. D., membre et codirecteur de recherche

M. OUALI Mohamed-Salah, Doctorat, membre

## DEDICATION

*I would like to dedicate this thesis to*  
*my mother,*  
*for all her inimitable love and support...*

## ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude and special appreciation to my advisor Prof. Soumaya Yacout for her continuous support of my research, for her patience and motivation. I am so thankful of you for allowing me to grow and finding correct ways during my work. Also, I really appreciate of Dr. Hany Osman my co-director during my research. Your great guidelines were the keys for my closed doors during my research. I would like also to thank my committee members, Dr. Luc-Déseiré Adjengue and Dr. Mohamed-Salah Ouali for serving as my committee members considering its difficulties. I also appreciate you for your valuable comments and suggestions and making my defense session as a memorable event. Also, I really appreciate of Prof. François Soumis. Your particular supports and helps facilitated my path to work and progress.

A special thanks to my family. Your encouragements, supports and prayers cause me to be sustained so far. I would like to appreciate all of my friends who advised and supported me during my research and made me more decisive to achieve my goal.

## RÉSUMÉ

Les données sont au cœur des industries et des organisations. Beaucoup d'entreprises possèdent de grandes quantités de données mais échouent à en tirer un bénéfice conséquent, bien souvent parce que ces données ne sont pas utilisées de façon productive. Il est indispensable de prendre des décisions importantes au bon moment, en utilisant des outils adaptés permettant d'extraire de l'information pratique et fiable de grandes quantités de données. Avec l'augmentation de la quantité et de la variété des données, le recours aux outils traditionnels facultatifs a été abandonné alors que l'importance de fournir des méthodes efficaces et prometteuses pour l'analyse de données se fait grandissante. La classification de données est l'un des moyens de répondre à ce besoin d'analyse de données.

L'analyse Logique de Données (**LAD** : Logical Analysis of Data) est une nouvelle méthodologie d'analyse de données. Cette méthodologie qui combine l'optimisation, l'analyse combinatoire et la logique booléenne, est applicable pour le problème de classification des données. Son but est de trouver des motifs logiques cachés qui séparent les observations d'une certaine classe de toutes les autres observations. Ces motifs sont les blocs de base de l'Analyse Logique de Données dont l'objectif principal est de choisir un ensemble de motifs capable de classifier correctement des observations. La précision d'un modèle mesure à quel point cet objectif est atteint par le modèle.

Dans ce projet de recherche, on s'intéresse à un type particulier de motifs appelé  $\alpha$ -motif «  $\alpha$ -pattern ». Ce type de motif permet de construire des modèles de classification LAD de très grande précision. En dépit du grand nombre de méthodologies existantes pour générer des  $\alpha$ -motifs maximaux, il n'existe pas encore de méta-heuristique adressant ce problème. Le but de ce projet de recherche est donc de développer une méta-heuristique pour résoudre le problème des  $\alpha$ -motifs maximaux. Cette méta-heuristique devra être efficace en termes de temps de résolution et aussi en termes de précision des motifs générés.

Afin de satisfaire les deux exigences citées plus haut, notre choix s'est porté sur le recuit simulé. Nous avons utilisé le recuit simulé pour générer des  $\alpha$ -motifs maximaux avec une approche

différente de celle pratiquée dans le modèle BLA. La performance de l'algorithme développé est évaluée dans la suite. Les résultats du test statistique de Friedman montrent que notre algorithme possède les meilleures performances en termes de temps de résolution. De plus, pour ce qui est de la précision, celle fournie par notre algorithme est comparable à celles des autres méthodes. Notre précision possède par ailleurs de forts niveaux de confiance statistiques

## ABSTRACT

Data is the heart of any industry or organization. Most of the companies are gifted with a large amount of data but they often fail to gain valuable insight from it, which is often because they cannot use their data productively. It is crucial to make essential and on-time decisions by using adapted tools to find applicable and accurate information from large amount of data. By increasing the amount and variety of data, the use of facultative traditional methods, were abolished and the importance of providing efficient and fruitful methods to analyze the data is growing. Data classification is one of the ways to fulfill this need of data analysis.

Logical Analysis of Data is a methodology to analyze the data. This methodology, the combination of optimization, combinatorics and Boolean logic, is applicable for classification problems. Its aim is to discover hidden logical patterns that differentiate observations pertaining to one class from all of the other observations. Patterns are the key building blocks in LAD. Choosing a set of patterns that is capable of classifying observations correctly is the essential goal of LAD. Accuracy represents how successfully this goal is met.

In this research study, one specific kind of pattern, called maximum  $\alpha$ -pattern, is considered. This particular pattern helps building highly accurate LAD classification models. In spite of various presented methodologies to generate maximum  $\alpha$ -pattern there is not yet any developed meta-heuristic algorithm. This research study is presented here with the objective of developing a meta-heuristic algorithm generating maximum  $\alpha$ -patterns that are effective both in terms of computational time and accuracy.

This study proposes a computationally efficient and accurate meta-heuristic algorithm based on the Simulated Annealing approach. The aim of the developed algorithm is to generate maximum  $\alpha$ -patterns in a way that differs from the best linear approximation model proposed in the literature. Later, the performance of the new algorithm is evaluated. The results of the statistical Friedman test shows that the algorithm developed here has the best performance in terms of computational time. Moreover, its performance in terms of accuracy is competitive to other methods with, statistically speaking, high levels of confidence.

## TABLE OF CONTENTS

DEDICATION .....	III
ACKNOWLEDGEMENTS .....	IV
RÉSUMÉ.....	V
ABSTRACT .....	VII
TABLE OF CONTENTS .....	VIII
LIST OF TABLES .....	XI
LIST OF FIGURES.....	XII
LIST OF SYMBOLS AND ABBREVIATIONS.....	XIII
CHAPTER 1 INTRODUCTION.....	1
CHAPTER 2 LITRATURE REVIEW .....	3
2.1 Introduction .....	3
2.2 Logical Analysis of Data.....	5
2.2.1 Data binarization and support feature selection .....	6
2.2.2 Pattern generation.....	10
2.2.3 Classification rule or theory formation .....	15
2.3 Objective of the Research .....	18
CHAPTER 3 EVALUATING THE BLA MODEL.....	19
3.1 Introduction .....	19
3.2 Background .....	19
3.2.1 Maximum $\alpha$ – Patterns.....	20
3.2.2 Sets, parameters, and decision variables of the original and BLA models .....	20
3.2.3 The original model .....	20

3.2.4	The BLA model.....	21
3.2.4	Example 3.1.....	22
3.3	Methodologies.....	26
3.3.1	Data Preprocessing.....	27
3.3.2	Generating Patterns Based on the BLA (module application 1).....	27
3.3.3	Providing tools to calculate discriminant function.(module application 2).....	28
3.3.4	Calculating the accuracy of the generated patterns (module application 3).....	29
3.4	Implementation results of BLA model for 5 datasets.....	32
3.5	Conclusions.....	34
CHAPTER 4 DEVELOPING A META-HEURISTIC ALGORITHM.....		35
4.1	Introduction.....	35
4.2	Simulated Annealing.....	35
4.3	A developed pattern generation algorithm:.....	39
4.3.1	SA-BLA algorithm.....	40
4.3.2	Applied functions in SA-BLA.....	42
4.4	Calculating Accuracy.....	47
4.5	Performance evaluation of SA-BLA algorithm:.....	49
4.6	A review on results.....	50
4.7	Conclusions.....	51
CHAPTER 5 NUMERICAL RESULTS; COMPARISON AND ANALYSIS.....		52
5.1	Introduction.....	52
5.2	Numerical results.....	52
5.3	Comparison and Analysis.....	54
5.3.1	Ranking in terms of computing time.....	58

5.3.2	Ranking in terms of accuracy .....	59
5.4	Conclusions .....	60
CHAPTER 6	CONCLUSION .....	61
6.1	Summary .....	61
6.2	Synthesis of the study.....	62
6.3	Future enhancements.....	63
BIBLIOGRAPHY	.....	64

## LIST OF TABLES

Table 2.1: Basic dataset.....	16
Table 2.2: Binarized dataset of Table 2.1.....	16
Table 2.3: Small dataset for pattern generation example .....	17
Table 2.4: Generated patterns based on the small dataset of Table 2.3 .....	17
Table 3.1 Data set related to example 3.1 .....	22
Table 3.2 : Feasible solutions of the first stage related to example 3.1 .....	24
Table 3.3: Feasible solutions of the second stage related to example 3.1 .....	24
Table 3.4 : Feasible solutions of the third stage related to example 3.1 .....	25
Table 3.5 : Feasible solutions of the fourth stage related to example 3.1. ....	26
Table 3.6 The results for both BLA and cbm-LAD methods.....	33
Table 4.1 : The relation of Thermodynamic simulation and Combinatorial Optimization.....	37
Table 4.2 : Size of train and test set for assumed datasets(1).....	48
Table 4.3: Results of comparisons among three methods .....	50
Table 5.1: Datasets selected .....	53
Table 5.2 : Respective performance of cbm-LAD, BLA, SA-BLA.....	53
Table 5.3 Size of train and test set for assumed datasets (2).....	56
Table 5.4 : Rank of cbm-LAD, BLA and SA-BLA in terms of computational time .....	57
Table 5.5 : Rank of cbm-LAD, BLA and SA-BLA in terms of accuracy.....	57
Table 5.6 : The result of Friedman Test in-terms of computational time .....	58
Table 5.7. Wilcoxon signed–rank results for each pairs of methods .....	59
Table 5.8 : The result of Friedman test in-terms of accuracy .....	59

## LIST OF FIGURES

Figure 3.1 : Process and strategy in applying BLA.....	27
Figure 3.2 : Algorithm for Module application 1 .....	28
Figure 3.3 : Algorithm for module application 2 .....	29
Figure 3.4 : Algorithm for module application 3 .....	31
Figure 4.4.1: Simulated Annealing algorithm.....	39
Figure 4.2 : Pseudo code of SA-BLA algorithm.....	45
Figure 4.3 : Pseudo code of costf function.....	46
Figure 4.4 : Pseudo code of check - constraint function .....	46
Figure 4.5 : Pseudo code of neighborhood function .....	47
Figure 4.6 : Pseudo code of Remove - covered – observation.....	47
Figure 4.7 : Pseudo code to calculate weight and coverage vectors .....	48
Figure 4.8 : Pseudo code to calculate accuracy.....	49

**LIST OF SYMBOLS AND ABBREVIATIONS**

AI	Artificial Intelligence
BLA	Best Linear Approximation
CBM	Condition Based Maintenance
CHA	Combined Heuristic Algorithm
DTCs	Decision Tree Classifiers
IDEA	Iterative Discriminant Elimination Algorithm
IID	Independent and Identically Distributed
IoT	Internet of Things
LAD	Logical Analysis of Data
SVMs	Support Vector Machines
MPP	Maximum Prime Patterns
MSP	Maximum Strong Patterns
NNs	Neural Networks
RUTCOR	Rutgers Center for Operation Research
SA	Simulated Annealing
SA-BLA	Simulated Annealing of Best Linear Approximation

## CHAPTER 1 INTRODUCTION

Data Analysis is the process of systematically using statistical and/or logical techniques for explaining, refining, condensing, evaluating and modelling data. Different analytical methods lead to discover characteristics of data through inductive reasoning. These methods are designed with the aim to find fruitful information and to assist in decision making or in building learning models. These methods are capable of distinguishing the interesting and applicable part of data from statistical errors or implementation faults. (Shamoo & Resnik, 2003)

Logical Analysis of Data (LAD) is a logic based methodology used for analysing data. Specifically, it is used for the analysis of observations that possess a set of determined attributes associated to a specific problem. The main missions of LAD are; detecting the minimal sets of attributes that are representative of all the observations in order to express their traits (specifications) and finding the logical patterns aimed to distinguish observations belonging to one class from those belonging to the opposite class (for the two class LAD model). A set of combined attribute values that are observed simultaneously in some of the observations forms a pattern characteristic. These patterns are later combined and applied to generate a general decision rule or classification model. In brief, LAD is composed of three sequential stages; data binarization and support feature selection, pattern generation and classification rule formation (Boros et al., 2000).

Patterns maintain the structural information concealed in data. Managing an optimal set of patterns due to one or more preference criteria is a combinatorial optimization problem, which makes the application of LAD a heavy computing load, considering that the number of patterns detected in a dataset is very large. As a matter of fact, the basic reasoning behind the choice of the support features in LAD is to decrease the computational complexity of pattern generation. The presented methods of pattern generation in literature are categorized in three groups; enumeration based and constructive methods, heuristic algorithms, and based on a MILP (Mixed Integer Linear Programming) model. Later, these patterns are accumulated to build a discriminant function. This aggregation of patterns is applied to determine the class of new observations. Patterns are inferred as a performance core of LAD and pattern generation has been the main part of data analysis in LAD. (Ryoo & Jang, 2009)

There are different kinds of patterns applied in LAD to build a decision rule. Each kind of patterns has its specific characteristics. Maximum  $\alpha$ -pattern is one type of pattern currently considered a lot among researchers. Maximum  $\alpha$ -patterns are significant to build highly accurate LAD classification models. Moreover, LAD classification models based on maximum patterns, contrary to more classic models, are easy to adapt to different users. To generate maximum  $\alpha$ -patterns, similar to any other kind of pattern, a number of exact and heuristic methods have been presented. (Boros et al., 2000)

Although various methods of pattern generation already exist, the growing demand for efficient decision models, capable of making accurate and relatively quick predictions from large datasets, motivates researchers to adapt and improve the previous models and algorithms, creating new methods of pattern generation. Therefore, the objective of this study is the development of a new algorithm that, by using meta-heuristic methods, attempts to meet the need for fast and precise pattern generation. The performance of the presented algorithm is evaluated by processing large datasets, but its capacity accommodates even larger ones.

In this study, the first meta-heuristic algorithm using Simulated Annealing is developed, whose goal is the generation of maximum  $\alpha$ -patterns. The performance of the developed algorithm, called SA-BLA, is compared with an Integer Linear Programming model built based on the Best Linear Approximation technique to generate maximum  $\alpha$ -patterns. This model generates approximate maximum patterns and was referred to as the Best Linear Approximation (BLA) model. Also, the outputs of the SA-BLA, are compared with cbm-LAD (Yacout, Salamanka & Mortada, 2012). This software is available in the laboratory of Mathematics and Industrial Engineering of Ecole Polytechnique Montreal and can be used to generate patterns which are applicable in Condition Based Maintenance (CBM). Finally, a statistical Friedman test is done to evaluate the performance of SA-BLA and the other available methods.

The thesis is organized as follows. A brief review of related literature is presented in next chapter, which is followed by the implementation of the BLA model and an evaluation of its performance in comparison with cbm-LAD in chapter 3. In chapter 4 an algorithm using meta-heuristics is presented, then follows a comparison and analysis of the numerical results in chapter 5. Finally, the conclusion and future enhancements is discussed in chapter 6.

## CHAPTER 2 LITRATURE REVIEW

### 2.1 Introduction

This chapter contextualizes this thesis's research in regard to previous works.

Smart devices and the Internet of Things (IoT), today, generate a huge amount of digital data. Data can be analysed to form patterns and thus help to inform intelligent decision-making, with tools such as Machine Learning. Supervised learning is a machine-learning task that makes it possible to gain information of certain data characteristics, such as class or quality, by learning from training data (Kim & Choi, 2014). Training data or Train Set is a given historical dataset and consists of input observations. Train Set is a set of vectors that are built based on the values of features (attributes) at certain instances. The process of supervised learning starts first by analysing training data. Then, it builds a model or a function from the training data and applies that model to find the correct result for the new input data. This new data is considered the test data or Test Set (Bishop, 2006).

Supervised learning is one of the two categories of the artificial intelligence (AI) approach. The second category is unsupervised learning. Both categories consist of two steps; training and testing. In the case of supervised learning, the classes associated with the observations are known while they are not identified for unsupervised learning (Mortada, Yacout, & Lakis, 2014).

There are many algorithms that employ supervised learning. These algorithms are used for classification problems. The classification problems are categorized into binary classification and multi-class classification. The multi-class classification is associated with  $k$  as the number of classes in a dataset. For  $k = 2$ , the classification problems are categorized as binary or two-class classification problem.

There are a variety of algorithms to solve binary classification problems. The literature presents different classification methods: decision tree classifiers (DTCs) (Safavaian & Landgrebe, 1991), support vector machines (SVMs) (Schölkopf, & Smola, 2002), and neural networks (NNs) (Hagan, Demuth, & Beale, 1996). After these methods were applied in numerical experiences,

they could not meet the need of casual relationship between the inputs (observations) of the introduced method and their outputs (class of observations).

To refine the mentioned methods, an efficient method has been presented and employed. This method has shown high accuracy and interpretability power in various domains like medical, marketing, engineering, and manufacturing. This method is called Logical Analysis of Data (LAD) (Hammer, Kogan, & Lejeu, 2012).

Appealing characteristics of LAD, that distinguishes this method from the other methods, are: strong explanatory power, substantial flexibility, robustness according to the noise and measurement errors, ability to handle missing data, ability to build accurate classification models comparable to (and frequently more accurate than) other methods, and vast applicability regardless of having specific assumptions for the data (Boros et al., 2000). LAD is a methodology based on combinatorics, Boolean logic, and optimization. It was first introduced by Crama, Hammer, & Ibaraki (1988). It is a supervised learning classification approach and builds decision models, or rules, based on Boolean patterns. These patterns are extracted from the Train Set.

CBM (Condition-Based Maintenance) is one of the specific domains in employing LAD as a classification model. Applying LAD in this domain reveals the importance of this method to the other classification methods. Applications of CBM can be divided into two categories: diagnostic and prognostic. Statistical and Artificial Intelligence approaches as data-driven methods are applied for diagnosis and prognosis faults in CBM. The common disadvantage of statistical approaches is their dependence on statistical assumptions. For example, they need the use of independent and identically distributed (IID) input data as a precondition. Also, the unknown relationship between the input data and the diagnostic decision, made based on the input data, results in the loss of interpretability. LAD, as an Artificial Intelligence approach, modifies this limitation. This decision model can generate patterns which are transparent and explainable by experts. Having interpretable patterns is important in the equipment health management field. In the field of CBM, LAD has proved its efficiency as a classification tool and it is competitive to the other classification techniques (Mortada, Yacout, & Lakis, 2011).

The specific place of LAD among various classification methods has motivated researchers recently to work further on this method. In this study, we attempt to apply meta-heuristics to develop a pattern generation algorithm that satisfies our needs. In the remaining section LAD and corresponding stages of it as the decision model for this study is explained comprehensively.

## **2.2 Logical Analysis of Data**

Logical analysis of data, as a supervised data mining approach, was presented by the RUTCOR (Rutgers Center for Operations Research) researchers at Rutgers University in USA. First, LAD was applied by Crama et al. (1998) as a Boolean technique, used to identify the causes of a certain event by investigating a set of factors. In general, LAD is aimed to extract knowledge hidden in observations of the given dataset, or, in other words, the Train Set. This extraction is used to discover the causes set which results in certain effects. The set of causes is considered to be the set of attributes for LAD. (Crama et al.,1988)

In binary classification problems, observations are categorized into positive and negative classes. Each observation can be shown as binary, numerical, or nominal vectors. Every observation includes the value of certain characteristic attributes (features). Basically, LAD has been used as two-class classification technique (Boros et al., 2000). A specific characteristic of LAD is the detection of collections of patterns which are the interactions between variables for either positive or negative observations in the dataset. LAD can be employed as pattern-based classifier of the new observations that do not belong to the Train Set. These new observations are chosen from the original dataset and considered as the Test set. The accuracy of LAD model is calculated based on the Test Set. (Crama et al.,1988)

There are specific stages to construct the LAD classification model. These stages are similar for both two-class and multi-class LAD decision models. This study will develop a pattern generation algorithm for the two-class LAD decision model. Therefore, the presented literature is concentrated on the two-class. The three stages of the LAD classification model will be explained in detail. These three steps are data binarization and support feature selection, pattern generation, and classification rule and theory formation.(Ryoo& Jang, 2009). Following sections explains every stage of LAD briefly.

### 2.2.1 Data binarization and support feature selection

Firstly, LAD was applied to the Boolean attributes. In mathematics and abstract algebra, a Boolean domain is a set consisting of exactly two elements whose interpretations are either false or true. Boolean attributes are applicable to many real-life problems. But many more problems use more complex data, like numerical (e.g. weight) or nominal (e.g. gender) variables. Moreover, in the majority of problems, the variables are ordered. This happens when the attributes are numerical (e.g. low weight or high weight) or nominal but comparable (e.g. educational degree: elementary, high school, college/university).

LAD must be applicable to any kind of problem. These problems can have various kinds of variables. This leads to techniques that modify LAD to the model compatible with different kinds of variables. Hammer, Liu, Simeone, & Szedmák. (2004.1) proposed “Logical Analysis of Numerical Data” as an approach to use numerical values for LAD decision model.

Basically, binarization involves the transformation of each numerical data as a non-binary attribute (variable) to a number of binary attributes. This technique is implicitly done by comparing the values of numerical attributes with standard “cut-points” (critical values) (Boros et al., 2000).

At first, Boros, Hammer, Ibaraki, & Kogan. (1997) developed the theoretical foundations of the binarization process. In Boros et al. (2000), two types of Boolean variables associated with each numerical attributes were introduced. The first is level variable. Level variables are related to every cut-point. It is assumed that  $y$  is a numerical attribute and  $c$  is a cut point. The corresponding level variable  $b_{y,c}$  is defined as follows:

$$b_{y,c} = \begin{cases} 1 & ; \text{ if } y \geq c \\ 0 & ; \text{ if } y < c \end{cases}$$

Therefore, every numerical attribute is replaced with  $n$  binary attributes where  $n$  is the number of cut-points. The second type of Boolean variable is interval variable. This type of variable is correlated with a pair of cut-points. This Boolean variable defines whether the numerical attribute  $y$  belongs to the interval consisting of cut-points  $c', c''$  as its end points. This interval variable is introduced as:

$$b_{y, c', c''} = \begin{cases} 1; & \text{if } c' \leq y \leq c'' \\ 0 & ; \text{otherwise} \end{cases}$$

Finally, every numerical attribute is transformed to the  $n$  binary attributes when  $n$  is the number of pairs of cut-points (Boros et al.,2000).

All discretization methods try to minimize the number of cut-points and the number of inconsistencies. Having fewer cut-points results in having fewer binary attributes obtained from the transformation of numerical attributes, thus result in a simpler model, but may also result in a lack of discretizing power. A lack of discretizing power is known as a lack of correlation between attributes and classes. In literature there are two optimality approaches for binarization. Those, respectively, are: 1) binarization based on the number of cut-points, 2) binarization based on the number of inconsistencies (Kotsiantis et al.2006). These two approaches considered as simplicity and consistency can be obtained by the support sets. Crama et al. (1997) developed support sets.

The support set is a collection of binary attributes. This set maintains the consistency of the classification even if all the other binary attributes are eliminated. Later, a set-covering model, with the objective of finding the minimal support set, was introduced. This model minimizes the number of binary attributes according to the constraints of support set. A minimal support set consists of at least one binary attribute, such that removing that binary attribute leads to have identical observations for different classes. After, Boros et al. (2000) rectified the presented model in three directions. This modification aimed to assure a certain minimum level of distinguishing power for the generated support sets. The first part assures that two classes can be discriminated by more than one feature. The second one assigns the weights as the coefficients of variables for the objective function. These weights are defined according to the estimated distinguishing power for a single binary attribute. Distinguishing power demonstrates the discriminating quality of a binary attribute for the observations. Different statistical measures can be applied to calculate the distinguishing power. Finally, the third modification guarantees that the cut-points separate the observations correctly.

Chvatal (1979) presented a greedy heuristic to solve the support set problem. This heuristic works recursively and generates a near optimal solution. There are three dependent steps for this algorithm. First, it begins with a set that includes all features. Second, the algorithm iterates like

a finite loop. It aims to remove features, one by one, in every repetition. Achieving the maximum diminution of the objective function is the criterion to remove the feature. The last step is the satisfaction of the stopping criterion. The removing feature continues until eliminating any more of it results in having equal observations for different classes. By Boros et al. (2000) a greedy method that builds the support set, iteratively, was proposed. For this method, at first, a null set was considered as the support set. The algorithm chooses one feature at each iteration to add to the set. The feature is selected such that it results in the maximum decrease of the objective function regarding the satisfaction of constraints. Boros, Horiyama, Ibaraki, & Makino (2003) presented a new approach. In this approach, different measures of separation between different classes were introduced, in order to have a support set including essential attributes. An integer linear programming model, with the objective of maximizing separation power, and based on the defined measures to separate, was introduced. In other words the objective function attempts to maximize the separation power instead of minimizing the number of variables in the support set. Several heuristic algorithms were presented to solve this ILP model. Some of them perform efficiently to find the smallest set of essential attributes.

Liu, Hussain, Tan, & Dash (2002) considered accuracy of the classification model as the factor, other than simplicity and consistency, to evaluate the quality of the discretization method. The accuracy of the LAD decision model is the ratio of the correctly classified observations to the total number of observations. Having a high quality discretization method, results in having more power in discretizing the observations correctly. This event subsequently leads to an accurate classification model. The time of discretizing is the other effective parameter in the quality of the discretization method. Achieving a reasonable balance between the accuracy of the decision model and the time of discretization is a point that has to be considered.

As mentioned earlier, LAD was known currently as the classification model distinguished from other methods because of its interpretability and transparency for classification. Data binarization as the pre-processing phase of the LAD decision model is investigated by the researches.

Mayoraz & Moreira (1999) introduced a number of algorithms for data binarization as a pre-processing step for LAD. IDEA, which stands for Iterative Discriminant Elimination Algorithm, is an eliminative algorithm. It tries to find a minimal discriminant set consistent with the set of training examples. This algorithm iteratively removes discriminants earlier nominated

as cut-points. The criterion to remove the cut-point is whether or not it has the maximum inconsistency with the training data set. Allmuallim & Dietteriech (1994) presented two exact and three greedy heuristic algorithms for binarization of the LAD learning model. The two exact algorithms are solved in quasi-polynomial time, but the greedy ones are solvable in less computational time for optimality. The simple greedy proposed by Allmuallim & Dietteriech (1994), like other presented greedy algorithms, operates iteratively. The adopted approach for the simple greedy is contrary to the one that presented for the IDEA. The simple greedy, in every iteration, attempts to add a discriminant or cut-point with the maximum consistency with the Train Set.

The binarization procedure, in any global algorithm, begins by ranking. In rising order, all the distinct values of the numerical attribute A are as  $v_a^1 < v_a^2 < \dots < v_a^l$  ( $l \leq M$ ).  $l$  shows the number of separate values of the attribute A and M is the number of observations. The cut-point  $cut_{a,j}$  defined as  $cut_{a,j} = (v_a^{(i)} + v_a^{(i+1)})/2$  such that  $v_a^{(i)}$  and  $v_a^{(i+1)}$  respectively belong to the positive (negative) class and negative (positive) class of observations (Hammer et al., 2000). By this definition, every cut-point indicates a boundary that is capable of distinguishing observations into different classes as positive or negative. This characteristic leads to meeting the consistency goal. The simplicity of discretization method can be satisfied by some algorithms, like IDEA for the LAD classification model. Later, Bruni (2007) presented a method to satisfy the accuracy goal. The accuracy shows how well the discretization method helps the classifier to be more accurate. In this author's view, the power of distinguishing for every cut-point is a criterion that has to be considered to satisfy the accuracy. Knowing the real class of each observation is necessary to calculate the accuracy. After determining the class of observation by the classifier (classifier is made based on Train set), a comparison between the outcome of the classifier and the real class of the observation, which belongs to Test set, has to be done. The result of this comparison shows the accuracy of the classifier.

Alexe (2006.2) presented the pattern-based approach for choosing the attributes. A two-step algorithm was proposed to select the attributes. The first step is the filtering phase. In this step, a small number of attributes is selected based on some criteria. In the second step, the number of times every selected attribute appears in the patterns is evaluated. This factor determines the priority of attributes to choose. The attributes are then sorted in decreasing order. Finally, the

first half of the attributes is removed to build a new model. This process will iterate until there is no improvement in the accuracy of the constructed model. In next section the definition of patterns and a brief description of different kinds of patterns in addition to various methods to generate patterns is introduced.

### **2.2.2 Pattern generation**

Pattern generation is the main step in LAD decision model. Patterns are the key building block in the Logical Analysis of Data (Boros et al.,2000). A pattern is a term with some specific characteristics. A term is a conjunction of distinct literals. A literal is either a binary variable or its negation. The negation of an attribute refuses the attribute. A term does not include both a literal and its negation.

A positive (negative) pattern is a combination of literals that can cover at least one positive (negative) observation and none of the negative (positive) observations. An observation is covered by a pattern when all the literals of the pattern reflect the attributes value of the observation (Hammer et al., 2004.1). The set of all observations that are covered by the pattern  $P$  is written as  $Cov(p)$  and is called the coverage of pattern. The number of literals of a pattern is named the degree of the pattern. Patterns with low degree are more likely to have a bigger coverage, while patterns with high degree probably have fewer covered observations.

In LAD, patterns are used as inference rules in the modelling process. We expect high quality patterns to employ for test data. Prevalence and Homogeneity of the patterns as quality criteria of patterns are the two most important measures which are used for determining LAD performance. Prevalence of a positive (negative) pattern is the ratio of the coverage of the pattern to the total number of positive (negative) observations. Homogeneity of a positive (negative) pattern is the proportion of the number of positive (negative) observations to the total number of positive and negative observations that the pattern covers. (Ryoo & Jang , 2009)

Based on the definitions of the prevalence and homogeneity, the prevalence of the pattern shows the capability of the pattern to find the observations of each class. The homogeneity of a pattern demonstrates the distinguishing power of the pattern. The homogeneity illustrates how accurately

a pattern distinguishes the observations of its class from another class (Alexe & Hammer, 2006.3).

There are four specific types of patterns introduced by Hammer et al. (2004.1) and Hammer, Kogan, Simeone, & Szedmák (2004.2): prime, spanned, strong and maximum  $\alpha$ -pattern. Prime patterns are the simplest type of the patterns. Removing any literal of the prime pattern does not result in any other pattern. The spanned patterns are the conjunction of the maximum number of literals for the same number of covered observations. As a comparison between prime and spanned patterns it is worthy to say that prime patterns are simpler while spanned patterns are more selective. A pattern  $P_i$  is strong if there is no any other pattern  $P_j$  with more coverage than  $P_i$ . A maximum- $\alpha$  pattern is the pattern that has the most coverage among all the patterns that covers observation  $\alpha$ .

The most fundamental stage of the LAD is to generate patterns. It is desirable in the set of observations that these patterns express well the characteristics of their associated class. Various models and algorithms for generating different kind of patterns have been presented in the literature. All of them attempt to meet the basic demands for having patterns.

At the time LAD was developed, two procedures to generate prime patterns were introduced (Crama et al., 1988). The first procedure applies a set covering model to generate prime pattern as its minimal solution. The second one is the enumeration method. This procedure enumerates easily all the short and bounded primary combinations of literals. After, the method examines whether this combination can be a pattern or not. This algorithm is solvable in polynomial time when the patterns degree (number of literals) is predefined.

Two “top-down” and “bottom-up” approaches were considered to generate prime patterns (Boros et al., 2000). In pattern generation methods, not missing any of the “best” patterns is important. The top-down method begins with a correlated pattern of the characteristic term for every observation. This method continuously removes the literals until arriving to a prime pattern. The bottom-up method considers short patterns. It begins with a pattern that has just one literal. If this positive (negative) pattern does not cover any negative (positive) observation it would be a prime pattern. Otherwise, the method searches for all the possible patterns with degree two. The process of adding literals one by one continues until all the associated observations are covered

by the pattern. The approach of (Boros et al., 2000) was based on applying two principles to generate patterns: simplicity and comprehensiveness. The simplicity attempts to generate short patterns while comprehensiveness implies having patterns that can cover as many observations as possible such that all the observations can cover by a pool of all patterns. This objective is achieved by the hybrid bottom-up and top-down approach. The hybrid method starts with the bottom-up approach to achieve patterns with a pre-defined degree. These patterns have usually four or five literals in their structures. Then the top-down approach is applied to cover all the remaining observations that are not covered in the first phase.

By Alexe & Hammer (2006.3) for the first time two systematic enumeration algorithms to generate spanned patterns were introduced, named as SPAN and SPIC. Both algorithms are solved in linear time. The results show the high efficiency of SPIC and its noticeable stability by increasing the number of cut points. Increasing the number of cut points does not affect the complexity of the SPIC. While Ryoo & Jang (2009) explained that incrementing the number of cut points leads to increasing the complexity of algorithms that generate prime patterns. Alexe & Hammer (2006.3) illustrates that spanned patterns are highly robust. It means that the reduction of prevalence and homogeneity by the spanned patterns is low in test sets in comparison with the train sets. Moreover, it was shown that spanned patterns have less number of classification errors in comparison with the prime patterns. Also, spanned patterns have better performance in low-quality datasets than prime patterns.

Alexe, Alexe, Hammer & Kogan (2008) considered comprehensible and comprehensive patterns. Comprehensible patterns have lower degrees in their structure. Because of this specific characteristic, comprehensible patterns can cover more observations in their class. A drawback is that, in the two-class LAD model, they tend to have a high coverage for the wrong class. On the contrary, comprehensive patterns can cover less number of observations in their associated class. The coverage of these patterns for their opposite class is low, which is an advantage for the comprehensive patterns. Prime patterns are comprehensible patterns while spanned patterns are comprehensive patterns. Results show that using comprehensive patterns does not increase much the accuracy of classification.

By Hammer et al. (2004.2) three preference criteria were introduced to compare the efficiency of patterns. The first one is simplicity. Based on the definition of simplicity; a pattern  $P_i$  is

simplicity-wise preferred to pattern  $P_j$  if and only if the set of literals of pattern  $P_j$  includes the set of literals of pattern  $P_i$ . The simplicity preference induces short patterns or the patterns with low degree. Simplicity preference results in reducing the number of “false negatives” or negative observations that are covered by positive patterns incorrectly. But it does not guarantee that positive patterns can cover all positive observations or to be safe for “false positives”. This leads to reject the assumption that introduces prime patterns as best pattern. The next preference criterion is selectivity. A pattern  $P_i$  is selectivity-wise preferred than pattern  $P_j$  if and only if the set of wrongly covered observations by  $P_i$  is the subset of the ones of  $P_j$ . This kind of preference is totally opposite of the simplicity-wise preferred. The last preference criterion is evidentially. A pattern  $P_i$  is evidentially preferred to a pattern  $P_j$  if and only if the coverage of pattern  $P_j$  is the subset of the one of pattern  $P_i$ . This is exactly compatible with the definition of strong patterns. It was shown that strong pattern has a good performance. The other preference is also presented by Hammer et al. (2004.2). This is a combination of selectivity-wise and evidentially-wise preference. The patterns that are optimal with respect to this preference are called spanned patterns. Strong patterns that are moderated by simplicity preference are nominated as strong prime and the ones that are moderated by selectivity nominated as strong spanned patterns. It was shown that strong prime patterns result in less number of unclassified observations while strong spanned patterns lead to less number of classification errors.

Two pattern generation algorithms generating strong spanned and strong prime patterns are presented by Alexe et al. (2008). These algorithms respectively aim to do a comparison between large, comprehensive patterns and small, comprehensible ones. The computation time of these algorithms increase linearly with the number of literals of all possible patterns. It is, however, still quite high. The efficiency of the classification models that are built based on the patterns of these algorithms is almost similar. Due to the dataset the suitable algorithm to generate strong spanned or strong prime pattern is determined.

For a positive (negative) observation  $\alpha$ , Hammer & Bonates (2006) introduced a maximum positive (negative)  $\alpha$ -pattern. This pattern has a maximum coverage among all positive (negative) patterns that cover  $\alpha$ . The high coverage is the only available criterion that can inform the user about the accuracy of the model. Based on Boros et al. (2000) patterns with more coverage can perform better for the test set than the ones with lower coverage. They give stronger indication of

the class of new observations. A polynomial set covering problem to generate maximum  $\alpha$ -pattern was presented by Hammer & Bonates (2006). Later, they applied the best linear approximation (BLA) in  $L_2$  for the defined polynomial objective function. The results demonstrated the utility of the approximate maximum patterns (Bonates, Hammer & Kogan, 2008).

Bonates et al., (2008) presented two more heuristics to generate maximum  $\alpha$ -patterns. MPP (Maximized Prime Patterns) and MSP (Maximized Strong Patterns) are two heuristics that were introduced respectively to develop Maximized Prime Patterns and Maximized Strong Patterns. The MPSP is built with the combination of MPP and MSP heuristics. The CHA (Combined Heuristic Algorithms) is introduced as the combination of the three last algorithms. Results show the best performance for the CHA in terms of accuracy and computational time, while the running time for MPP and MSP respectively was sensitive to the number of binary attributes and number of observations for large datasets. Also, it was concluded that maximum patterns are fruitful to have accurate LAD decision models.

Ryoo & Jang (2009) developed an MILP (Mixed Integer Linear Programming) model for pattern generation. This approach can generate LAD patterns that are optimal due to various preferences. These preferences defined as simplicity, selectivity and evidentially. The method of Ryoo & Jang (2009) can identify patterns based on the requirements of the users. These requirements can be prevalence, homogeneity and the complexity of the generated patterns. The MILP model attempts to minimize the number of uncovered observations by the optimal patterns that are generated regarding the defined preferences. It was shown that efficiency and utility by LAD can be increased with MILP approach.

The new version of MILP model was introduced by Guo & Ryoo (2012). The new model can generate strong prime and strong spanned patterns with fewer number of variables than the model of Ryoo & Jang (2009). These patterns respectively can decrease the number of unclassified and misclassified observation.

As it was shown, a general categorization of all presented algorithm for pattern generation is enumeration and constructive algorithms, heuristic algorithms and finally MILP model. The next section introduces the last stage of the LAD model.

### 2.2.3 Classification rule or theory formation

The last stage of the LAD decision model is called Theory formation. In this step a decision rule is built from the patterns previously generated. The patterns are generated based on the observations of the *Train Set*. Each positive (negative) observation is covered by at least one positive (negative) pattern. Every positive pattern indicates, to a certain degree, whether or not an observation from the *Test Set* should be marked as positive. So, it is expectable that having a sufficient number of patterns leads to a general classification rule. This rule is called Theory.

It is assumed that  $p_1^+, p_2^+, \dots, p_n^+$  and  $p_1^-, p_2^-, \dots, p_m^-$  are respectively the set of positive and negative patterns of a *Train Set*. Theory formation attempts to make a decision rule, called discriminant function, from those patterns. Thus, discriminant function is a weighted sum of positive and negative patterns whose result could classify new observations. This function is defined as follows; (Mortada et al., 2014)

$$\Delta(o) = \sum_{i=1}^n w_i^+ p_i^+(o) - \sum_{j=1}^m w_j^- p_j^-(o) \quad (2.1)$$

$p_i^+(o)$  is equal to one if the positive pattern  $p_i^+$  covers the observation “o” otherwise it is equal to zero. The same applies for  $p_j^-(o)$ , and it is equal to one when the negative pattern  $p_j^-$  covers the observation “o”. Both  $w_i^+$  and  $w_j^-$  are respectively the weights of positive and negative patterns. Finally, having  $\Delta(o) > 0$ ,  $\Delta(o) < 0$ ,  $\Delta(o) = 0$  categorizes observation “o” in positive, negative and unclassified observation, respectively (Bonates et al., 2008).

Multiple ways to calculate the weight of patterns were presented by Boros et al. (2000). One of them as a sample to calculate the weight is the ratio of the coverage of each pattern to the total coverages of total patterns. The other one is based on solving a linear programming model, whose objective is maximizing the separation power of discriminant function.

## 2.3 How LAD works?

In this section a brief description for the basic operations of LAD based on some examples related to each stage is presented.

Firstly, it is supposed that Table 2.1 demonstrates a dataset includes numerical and nominal variables. These variables by applying corresponding cut-points and based on the definition of interval and level variables that were explained earlier, are changed to binary variables presented in Table 2.2. It is shown by these tables that for every numerical/nominal variable there are a number of binary variables. (Boros et al., 2000)

Table 2.1: Basic dataset

	$X_1$	$X_2$	$X_3$	$X_4$
$S^+$	1	Green	Yes	31
	4	Blue	No	29
	2	Blue	Yes	20
	4	Red	No	22
$S^-$	3	Red	Yes	20
	2	Green	No	14
	4	Green	No	17

Table 2.2: Binarized dataset of Table 2.1.

$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$	$b_9$	$b_{10}$	$b_{11}$	$b_{12}$	$b_{13}$
0	0	0	1	0	0	1	1	1	0	0	0	0
1	1	1	0	1	0	0	1	1	0	0	0	0
1	0	0	0	1	0	1	1	0	1	1	0	1
1	1	1	0	0	1	0	1	1	0	0	0	0
1	1	0	0	0	1	1	1	0	0	1	1	1
1	0	0	1	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0

The binarized Table 2.2 is obtained from Table 2.1 by using the following level variables

$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$	$b_9$
$X_1 \geq 1.5$	$X_1 \geq 2.5$	$X_1 \geq 3.5$	$X_2 = \text{Green}$	$X_2 = \text{Blue}$	$X_2 = \text{Red}$	$X_3 = \text{Yes}$	$X_4 \geq 17$	$X_4 \geq 21$

And the following interval variables

$b_{10}$	$b_{11}$	$b_{12}$	$b_{13}$
$1.5 \leq X_1 \leq 2.5$	$1.5 \leq X_1 \leq 3.5$	$2.5 \leq X_1 \leq 3.5$	$17 \leq X_4 \leq 21$

After binarization the second stage of LAD that is pattern generation is clarified more by a small example. It is supposed that a small dataset by Table 2.3 has given. Table 2.3 is a binarized dataset.

Table 2.3: Small dataset for pattern generation example

observations	class	binary attributes				
		$b_1$	$b_2$	$b_3$	$b_4$	$b_5$
1	Positive(+)	1	1	1	0	0
2		1	1	1	1	1
3		1	0	1	1	0
4		1	1	1	1	0
5	Negative(-)	1	0	1	0	0
6		0	0	0	0	0
7		0	0	1	0	0
8		1	0	0	0	0

Based on the definition of the pattern presented before, the corresponding positive and negative patterns for the given dataset by Table 2.3 are shown in Table 2.4. The positive patterns  $b_2$  and  $b_4$  can cover three positive observations and they cannot cover any of the negative observations. Also, the negative pattern that is introduced as  $\overline{b_2 b_4}$  can cover all the negative observations and it does not cover any positive observation.

Table 2.4: Generated patterns based on the small dataset of Table 2.3

The generated <i>positive</i> patterns	Description	Covered observations
$P_1^+$	$b_2$	1,2,4
$P_2^+$	$b_4$	2,3,4
The generated <i>negative</i> patterns	Description	Covered observations
$P_1^-$	$\overline{b_2 b_4}$	5,6,7,8

The last stage of LAD is classification rule or theory formation. In the following how this theory is formed by the generated patterns of Table 2.4 is explained. Also, the application of this theory to classify new observations that do not belong to Train set, which is Table 2.3, is shown.

Based on the equation (2.1) and the positive and negative patterns showed in Table 2.4, the class of new observation is determined. It is supposed that  $w_i$  and  $w_j$  considered as the weight of

positive and negative patterns and defined as:  $w_i^+ = \frac{cov(p_i^+)}{\sum_i^n cov(p_i^+)}$  and  $w_j^- = \frac{cov(p_j^-)}{\sum_j^n cov(p_j^-)}$  considering

the generated patterns shown in Table 2.4 the  $w_1^+ = \frac{cov(p_1^+)}{\sum_i^2 cov(p_i^+)} = w_2^+ = \frac{cov(p_2^+)}{\sum_i^2 cov(p_i^+)} = \frac{3}{3+3} = 0.5$  are

the weights of positive patterns while the weight of negative pattern is  $w_1^- = \frac{cov(p_1^-)}{cov(p_1^-)} = \frac{4}{4} = 1$ .

The calculated theory based on the patterns weight is :  $\Delta(o) = 0.5p_1^+(o) + 0.5p_2^+(o) - p_1^-(o)$ .

For a new observation (1,1,0,1,0) that does not belong to Train set the value of the discriminant

function calculated as:  $\Delta(1,1,0,1,0) = 0.5 p_1^+(1,1,0,1,0) + 0.5 p_2^+(1,1,0,1,0) - p_1^-(1,1,0,1,0) =$

$0.5 + 0.5 - 0 = 1$ . So, the corresponding observation is classified as positive while the new

observation (1,0,0,0,1) has the discriminant value equal to one based on the discriminant

function

$\Delta(1,0,0,0,1) = 0.5 p_1^+(1,0,0,0,1) + 0.5 p_2^+(1,0,0,0,1) - p_1^-(1,0,0,0,1) = 0 + 0 - 1 = -1$ .

Therefore, this new observation belongs to the negative class. (Ragab, Yacout & Oulai, 2015)

## 2.4 Objective of the Research

Due to the presented review for pattern generation algorithms there is no meta-heuristic to generate maximum  $\alpha$ -patterns. Furthermore, increasing the size of problems, engaging with a large amount of data in data analysis domain, need of manipulating those data in reasonable computing time and promoting available tools to generate patterns, motivates this research study to develop a meta-heuristic algorithm to generate maximum  $\alpha$ -patterns.

Then, the result of this algorithm is compared with different available methods in-terms of computational time. Later, the accuracy of the learning models built by the patterns of the developed algorithm as the second criterion is also investigated.

## CHAPTER 3 EVALUATING THE BLA MODEL

### 3.1 Introduction

Maximum patterns are shown to be useful for highly accurate LAD classification models (Bonates et al., 2008). BLA (Best Linear Approximation) is an Integer Linear Programming model to generate approximate maximum patterns. It has shown considerable results in small datasets (Bonates et al., 2008). This chapter demonstrates the performance of the BLA model for different size of datasets and compares the results with the results of the cbm-LAD, which is the software that exists in the laboratory of Department of Mathematics and Industrial Engineering at Ecole Polytechnique Montreal (Yacout et al., 2012). To do so, three sequential steps are considered in the methodology; generating patterns based on the BLA, providing tools to calculate the discriminant function, calculating the accuracy. The results show that the accuracy of the patterns constructed by BLA model is low in comparison with the patterns of cbm-LAD, which are generated heuristically, for most of the datasets; while computational elapsed time is remarkable and even in some of the datasets is highly noticeable.

### 3.2 Background

In this section, some basic definitions and original model to generate exact maximum patterns is presented. Then, a description of the BLA model, stands for Best Linear Approximation of the original model, to generate approximate maximum patterns is introduced. This model after original model has been presented by Bonates et al. 2008. To build the BLA model the objective function of original model is replaced by its Best Linear Approximation in  $L_2$ . By this method the original model which is a generalized set covering problem changes to a weighted linear set covering problem (Bonates et al., 2008). At the end, two examples to clarify the BLA model are presented.

### 3.2.1 Maximum $\alpha$ – Patterns

A maximum positive(negative)  $\alpha$  – pattern is a positive(negative) pattern that has maximum coverage among all positive(negative) patterns capable to cover  $\alpha$ . The parameter  $\alpha$  can be any positive(negative) observations. All the following explanations are considered for maximum positive  $\alpha$  – pattern. (Bonates et al., 2008)

### 3.2.2 Sets, parameters, and decision variables of the original and BLA models

**Sets:**

- $j \in \{1, 2, \dots, n\}$  : set of binary features where  $n$  shows the number of features.
- $\Omega^+ = \{\beta_1, \beta_2, \dots, \beta_m\}$ , Set of positive observations
- $\Omega^- = \{\gamma_1, \gamma_2, \dots, \gamma_k\}$ , Set of negative observations

**Parameters:**

- $\alpha$  : The model attempts to generate an  $\alpha$  – pattern. So,  $\alpha$  is considered as one parameter for the model and  $\alpha \in \Omega^+ \subset \{0, 1\}^n \setminus \Omega^-$
- $\omega(\beta) = |\{j: \beta_j \neq \alpha_j\}|$  : It counts the number of features for a specific positive observation as  $\beta$  that have different value in corresponding features with  $\alpha$

**Decision variables:**

$$y_j = \begin{cases} 1 & \text{if feature } j \text{ is present in positive maximum } \alpha \text{ – pattern} \\ 0 & \text{if not} \end{cases}$$

or

$$y_j = \begin{cases} 1 & \text{if } j\text{th variable of the pattern is equal to } \alpha_j \\ 0 & \text{if not} \end{cases}$$

### 3.2.3 The original model

The original model to generate maximum  $\alpha$  – pattern that is a “Nonlinear Integer Programming Model”, based on its structure, is presented as follows:

$$\max \sum_{\beta \in \Omega^+} \prod_{\substack{j=1 \\ \beta_j \neq \alpha_j}}^n \bar{y}_j \quad (3.1)$$

$$\text{Subject to } \sum_{\substack{j=1 \\ \gamma_j \neq \alpha_j}}^n y_j \geq 1 \quad \forall \gamma \in \Omega^- \quad (3.2)$$

$$y_j \in \{0,1\} \quad \forall j \in \{1,2,\dots,n\} \quad (3.3)$$

( $\bar{y}_j$ , is considered as complement of variable  $y_j$  )

The objective function of the model, at first, finds the minimum conditions for satisfying constraints. Then, for all the positive observations tries to put the value zero for the variables that their corresponding attributes in positive observations has the value different from the one of  $\alpha$ . These kinds of solutions result in generating the patterns that explicitly can cover as much positive observations as possible. The generated patterns can cover observation  $\alpha$ . So, finally the model generates maximum positive  $\alpha$ -patterns.

### 3.2.4 The BLA model

The integer linear programming model is formulated as follows:

$$\min \sum_{j=1}^n \sum_{\substack{\beta \in \Omega^+ \\ \beta_j \neq \alpha_j}} \left( \frac{1}{2^{\omega(\beta)-1}} \right) y_j \quad (3.4)$$

$$\text{Subject to } \sum_{\substack{j=1 \\ \gamma_j \neq \alpha_j}}^n y_j \geq 1 \quad \forall \gamma \in \Omega^- \quad (3.5)$$

$$y_j \in \{0,1\} \quad \forall j \in \{1,2,\dots,n\} \quad (3.6)$$

and  $\omega(\beta) = |\{j: \beta_j \neq \alpha_j\}|$

Regarding to the given constraints, each solution of the model reveals for all the features corresponding to the literals presented in the maximum positive  $\alpha$  – pattern, there is at least one feature that has different value with  $\alpha$  for the negative observations. These constraints not only satisfy (are compatible with) the definition of positive pattern, but also guarantee the existence of  $\alpha$  – pattern.

The objective function of the model minimizes the number of observations which are not covered by the maximum positive  $\alpha$  – pattern. This happens implicitly with the parameter  $\omega(\beta)$ . This

parameter determines the coefficients of variables in objective function. A specific variable like  $y_k$ , corresponds to the feature  $k$ . If there are lots of observations different with  $\alpha$  in associated feature then the coefficient of  $y_k$  will be a big value. So, the objective function is reluctant to put the value equal to one for this variable. If based on the constraints this variable can have the value equal to zero then objective function considers zero value for this variable. Otherwise, the objective function attempts to choose another variable, which is candidate for having the value equal to one, with the less coefficient value. These explanations are clarified more by the following example.

### 3.2.4 Example 3.1

Data set related to Example 3.1 is shown in Table 3.1

Table 3.1 Data set related to example 3.1

	observations	class	binary attributes				
			$b_1/y_1$	$b_2/y_2$	$b_3/y_3$	$b_4/y_4$	$b_5/y_5$
$\beta_1$	1	Positive( $\Omega^+$ )	1	1	1	0	0
$\beta_2$	2		1	1	1	1	1
$\beta_3$	3		1	0	1	0	0
$\beta_4$	4		1	1	1	1	0
$\Upsilon_1$	5	Negative( $\Omega^-$ )	1	0	1	1	0
$\Upsilon_2$	6		0	0	0	0	0
$\Upsilon_3$	7		0	0	1	0	0
$\Upsilon_4$	8		1	0	0	0	0

The First stage: As shown before in the first example the assumptions for this stage are as follow:

$\alpha = (1,1,1,0,0) = \beta_1$   $\omega(\beta_1)=0$ ,  $\omega(\beta_2)=2$ ,  $\omega(\beta_3)=1$ ,  $\omega(\beta_4)=1$  and the model is:

$$\min \left( \frac{1}{2^{\omega(\beta_3)-1}} \right) y_2 + \left( \frac{1}{2^{\omega(\beta_2)-1}} + \frac{1}{2^{\omega(\beta_4)-1}} \right) y_4 + \left( \frac{1}{2^{\omega(\beta_2)-1}} \right) y_5 = \min \left( y_2 + \frac{3}{2} y_4 + \frac{1}{2} y_5 \right)$$

Subject to :  $y_2 + y_4 \geq 1$

$$y_1 + y_2 + y_3 \geq 1$$

$$y_1 + y_2 \geq 1$$

$$y_2 + y_3 \geq 1 \quad \& \quad y_1, y_2, y_3, y_4 \in \{0,1\}$$

For this stage of the example to understand better how the model works, at first step, the constraints are checked. To satisfy all the constraints of the model at least one of the associated variables to each constraint has to be set to one. Later, the objective function is investigated. The minimization objective function is planned to put the value zero, as much as possible, for its formed variables. If the objective function is limited to consider the value equal to one for specific variables it will choose the one with the less coefficient value in comparison to the other variables.

In this example, for the variable  $y_5$ , which is in the objective function, there is no constraint. So, the minimization objective function considers its value equal to zero. Based on the first constraint of the model, one of the remaining variables of the objective function, which is  $y_2$  or  $y_4$ , have to be set to one. According to the coefficients values of these variables the minimization objective function for all its optimal solutions set the values of variables  $y_2$  and  $y_4$  equal to one and zero respectively. Other variables of the model that are not in the objective function, considering the constraints, may have either zero or one. The significant point is that for the variable  $y_2$  there is one observation that its second attribute has the value different from the second attribute of the observation  $\alpha$ . Whereas, the number of observations that their fourth attribute are different from the one of  $\alpha$  is equal to two. So, the model automatically generates solutions which are compatible with selected  $\alpha$  and covers as much observations as possible.

It is assumed that  $S = (y_1, y_2, \dots, y_n)$  is the solution vector corresponding to the variables of the model. If  $S_1 = (0,1,0,0,0)$  is one of the optimal solution for first phase, the associated pattern for this solution, considering the definition of the variable  $y_j$ , will be equal to  $P_1 = b_2$ . Because the optimal solution shows that the second component of the solution vector  $S_1$  is equal to the value one. So, the associated pattern just has one literal related to the second attribute. Also, this literal has to be compatible with the selected  $\alpha$  which is the first observation. This leads to have a pattern that covering at least observation  $\alpha$ .

The feasible solutions for the model and associated patterns based on  $\alpha = \beta_1$  is shown in the Table 3.2. In this table  $S_i$  and  $P_i$  stand for solution  $i$  and pattern  $i$ .

Table 3.2 : Feasible solutions of the first stage related to example 3.1

$S_i$	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$P_i$	Coverage of $P_i$
$S_1$	0	1	0	0	0	$P_1 = b_2$	3
$S_2$	1	1	0	0	0	$P_2 = b_1b_2$	3
$S_3$	0	1	1	0	0	$P_3 = b_2b_3$	3
$S_4$	1	1	1	0	0	$P_4 = b_1b_2b_3$	3

The objective function of all feasible solutions of Table 3.2 has the value equal to 1. It shows that any of these feasible solutions can be the optimal solution. For this stage all of the generated patterns cover three observations among all positive observations. It means that all the patterns have the same importance for LAD model.

The Second Stage: In this stage  $\alpha$  is considered as second observation. So,  $\alpha = (1,1,1,1,1) = \beta_2$  and  $\omega(\beta_1) = 2, \omega(\beta_2) = 0, \omega(\beta_3) = 3, \omega(\beta_4) = 1$ . The corresponding model for this  $\alpha$  as follows:

$$\min \left( \frac{1}{2^{\omega(\beta_3)-1}} \right) y_2 + \left( \frac{1}{2^{\omega(\beta_1)-1}} + \frac{1}{2^{\omega(\beta_3)-1}} \right) y_4 + \left( \frac{1}{2^{\omega(\beta_1)-1}} + \frac{1}{2^{\omega(\beta_3)-1}} + \frac{1}{2^{\omega(\beta_4)-1}} \right) y_5 = \frac{1}{4} y_2 + \frac{3}{4} y_4 + \frac{7}{4} y_5$$

$$\text{Subject to: } y_2 + y_5 \geq 1$$

$$y_1 + y_2 + y_3 + y_4 + y_5 \geq 1$$

$$y_1 + y_2 + y_4 + y_5 \geq 1$$

$$y_2 + y_3 + y_4 + y_5 \geq 1 \quad \& \quad y_1, y_2, y_3, y_4, y_5 \in \{0,1\}$$

The feasible solutions for the model and associated patterns based on  $\alpha = \beta_2$  is shown in the Table 3.3. The value of the objective function for all of the feasible solutions is  $\frac{1}{4}$ .

Table 3.3: Feasible solutions of the second stage related to example 3.1

$S_i$	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$P_i$	Coverage of $P_i$
$S_1$	0	1	0	0	0	$P_1 = b_2$	3
$S_2$	1	1	0	0	0	$P_2 = b_1b_2$	3
$S_3$	0	1	1	0	0	$P_3 = b_2b_3$	3
$S_4$	1	1	1	0	0	$P_4 = b_1b_2b_3$	3

In this stage like the first stage the number of observations that cover by any pattern of this model is 3. So, there is no any superiority among the patterns to choose in-terms of their coverage.

The Third stage: In this phase  $\alpha$  is considered as the third positive observation to obtain maximum positive  $\alpha$ -pattern. So,  $\alpha = (1,0,1,0,0) = \beta_3$  and the related parameters and the model are:  $\omega(\beta_1) = 1$ ,  $\omega(\beta_2) = 3$ ,  $\omega(\beta_3) = 0$ ,  $\omega(\beta_4) = 2$

$$\min \left( \frac{1}{2^{\omega(\beta_2)-1}} \right) y_5 + \left( \frac{1}{2^{\omega(\beta_2)-1}} + \frac{1}{2^{\omega(\beta_4)-1}} \right) y_4 + \left( \frac{1}{2^{\omega(\beta_1)-1}} + \frac{1}{2^{\omega(\beta_2)-1}} + \frac{1}{2^{\omega(\beta_4)-1}} \right) y_2 = \frac{1}{4} y_5 + \frac{3}{4} y_4 + \frac{7}{4} y_2$$

$$\text{Subject to } y_4 \geq 1$$

$$y_1 + y_3 \geq 1$$

$$y_1 \geq 1$$

$$y_3 \geq 1 \quad y_1, y_2, y_3, y_4, y_5 \in \{0,1\}$$

Results are given in Table 3.4. The table has the same format of the Table 3.2 discussed earlier. The value of the objective function for all the feasible solutions is equal to  $\frac{3}{4}$ . Also, the generated patterns have the same coverages.

Table 3.4 : Feasible solutions of the third stage related to example 3.1

$S_i$	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$P_i$	Coverage of $P_i$
$S_1$	0	0	0	1	0	$P_1 = \overline{b_4}$	2
$S_2$	0	0	1	1	0	$P_2 = \overline{b_1 b_4}$	2
$S_3$	1	0	0	1	0	$P_3 = b_3 \overline{b_4}$	2

As it is observed in the Table 3.4, all the generated maximum patterns by the BLA model associated to the  $\alpha = \beta_3$  have the coverage equal to two. It is clear that by changing  $\alpha$  from  $\beta_1$  and  $\beta_2$  to  $\beta_3$  the coverage of the generated patterns reduced to two. These continuous stages of solving BLA shows the dependency of the coverages of the generated maximum patterns to the selected  $\alpha$ . In addition to the maximum patterns for this stage the strong patterns related to the basic dataset of the Table.3.2 are;  $P_{strong_1} = b_2$  and  $P_{strong_2} = b_2 b_3$ . Both of these patterns with coverage value equal to three were generated in the first and the second stages of solving BLA

for the assumed dataset. These explanations proved the importance of choosing  $\alpha$  as the parameter of BLA to obtain maximum patterns with the most coverage value.

The Fourth Stage: In the last stage  $\alpha$  is considered as the last positive observation. The associated BLA model for  $\alpha = (1,1,1,0)$  and other parameters are:  $\omega(\beta_1)=1$ ,  $\omega(\beta_2)=1$ ,  $\omega(\beta_3)=2$ ,  $\omega(\beta_4)=0$ ,

$$\min \left( \frac{1}{2^{\omega(\beta_3)-1}} \right) y_2 + \left( \frac{1}{2^{\omega(\beta_1)-1}} + \frac{1}{2^{\omega(\beta_3)-1}} \right) y_4 + \left( \frac{1}{2^{\omega(\beta_2)-1}} \right) y_5 = \frac{1}{2} y_2 + \frac{3}{2} y_4 + y_5$$

Subject to:  $y_2 \geq 1$

$$y_1 + y_2 + y_3 + y_4 \geq 1$$

$$y_1 + y_2 + y_4 \geq 1$$

$$y_2 + y_3 + y_4 \geq 1 \quad \& \quad y_1, y_2, y_3, y_4, y_5 \in \{0,1\}$$

The value of the objective function for all of the feasible solutions is equal to  $\frac{1}{2}$ . The outcomes of this stage are shown in Table 3.5.

Table 3.5 : Feasible solutions of the fourth stage related to example 3.1.

$S_i$	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$P_i$	Coverage of $P_i$
$S_1$	0	1	0	1	0	$P_1 = b_2$	3
$S_2$	1	1	0	0	0	$P_2 = b_1 b_2$	3
$S_3$	0	1	1	1	0	$P_3 = b_2 b_3$	3
$S_4$	1	1	1	0	0	$P_4 = b_1 b_2 b_3$	3

### 3.3 Methodologies

In order to implement the BLA model, a program composed of three modules is applied. Using these modules with the data from the literature necessitates a data preprocessing phase. So, this phase is added to the program. The first module is generating patterns based on BLA. The second module provides tools to calculate discriminant function by using the patterns generated in first module. The modules and data preprocessing phase are shown in Figure 3.1. Then five dissimilar datasets, diversing in the number of observations and the features, obtained from the UC Irvine repository are solved by the BLA model (Blake & Merz, 1998).

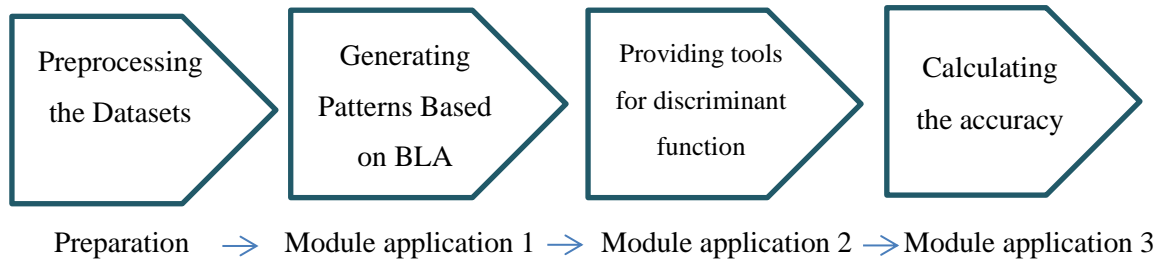


Figure 3.1 : Process and strategy in applying BLA

### 3.3.1 Data Preprocessing

The BLA model is an integer linear programming model designed to extract patterns (rules) from the binarized data. The patterns are aggregated into a classification model. This model is capable of distinguishing between positive and negative observations. Binarization is the initial step in LAD model. According to literature review, data binarization is considered as preparation stage to generate patterns (Yacout, 2012). Data binarization involves the transformation of the data used to train the LAD decision model to binary data with a binarization technique. This technique translates each numerical feature to a set of binary attributes (Mortada et al., 2014). To this purpose, cbm-LAD is employed for data binarization at data preprocessing phase.

### 3.3.2 Generating Patterns Based on the BLA (module application 1)

In order to generate a maximum positive (negative)  $\alpha$ - pattern at any iteration, the dataset is read completely at the beginning of the implementation. By considering each positive observation (negative observation) as candidate for parameter  $\alpha$ , in any iteration CPLEX solves the model and generates a maximum positive  $\alpha$ -pattern (maximum negative  $\alpha$ -pattern). The algorithm for the procedure of module 1 is shown in Figure 3.2.

### Module application 1

---

```

1: procedure generate patterns based on the BLA heuristic
2: Read all positive and negative observations (the data);
3: for every positive observation do
4:     alpha = current positive observation.
5:     Solve the BLA model and generate the optimal solution according to the data.
6:     represent the  $\alpha$ - pattern based on the solution
7: end for
8: end procedure

```

---

Figure 3.2 : Algorithm for Module application 1

### 3.3.3 Providing tools to calculate discriminant function. (module application 2)

Individual generated patterns of module 1 provide indications about the new observations. So, an adequately chosen collection of patterns is used for constructing a classification model (Bores et al., 2000). This classification model is an extension of partially defined Boolean function and it is called the theory (Hammer et al., 1986). The proposed method of building a theory involves a weighted sum of positive and negative patterns. This weighted sum is called a discriminant function (Bores et al., 2000).

The discriminant function is used to classify a new observation based on its function value. There are various formulas to calculate the weight of each generated pattern (shaban et al., 2015) In this thesis, the ratio of positive (negative) pattern coverage to the total coverage of positive (negative) patterns is employed to calculate the *weight* of a positive (negative) pattern. Coverage of positive is the total number of positive observations that are covered by the positive pattern. The coverage and the weight of each pattern are the tools required to calculate the discriminant function. The algorithm to determine these tools are shown in Figure 3.3. Also, ReadData and ReadPattern are the functions that are used in module 2. Moreover, coverages vector and weights vector are the vectors corresponding to coverages and weights of generated patterns of module 1. They are employed in the algorithm of the module application 2. Furthermore, the mathematical equation to calculate the discriminant function as Delta [obs] with its related components is given in the equations (3.7), (3.8) and (3.9)

## Module application 2

---

```

1: procedure providing tools to calculate discriminant function
2: Read all positive and negative patterns by Read Patterns;
3: Read all positive and negative observations by Read Data;
4: while (there is a positive (negative) pattern to investigate) do
5:   while (there is a positive (negative) observation to read) do
6:     if (positive (negative) observation is covered by positive (negative) pattern) then
7:       add value to the total coverage of positive (negative) pattern;
8:     else if (take another observation)
9:       end while
10: updating the coverages vector of positive (negative) patterns;
11: calculate the weight of each pattern;
12: updating the weights vector of positive (negative) patterns
13: end while
14: print out weights vector as output
15: end procedure

```

---

Figure 3.3 : Algorithm for module application 2

$$Delta[obs] = \sum_{i=1}^{N^+} \gamma_i^+ p_i^+(obs) - \sum_{i=1}^{N^-} \gamma_i^- p_i^-(obs) = DeltaPositive + DeltaNegative \quad (3.7)$$

St:

1.  $p_i^+ : i = 1, \dots, N^+, p_i^- : i = 1, \dots, N^-$  : Refer to positive and negative patterns, respectively such that  $N^+, N^-$  show the number of positive and negative patterns.

$$2. p_i^+(obs)(p_i^-(obs)) = \begin{cases} 1 & \text{if the pattern } p_i^+(p_i^-) \text{ covers the observation } obs \\ 0 & \text{otherwise} \end{cases}$$

$$3. \gamma_i^+ = \frac{cov(p_i^+)}{\sum_{i=1}^{N^+} cov(p_i^+)} \quad (3.8), \quad \gamma_i^- = \frac{cov(p_i^-)}{\sum_{i=1}^{N^-} cov(p_i^-)} \quad (3.9)$$

:  $\gamma_i^+ (\gamma_i^-)$  is the the weight of positive (negative) pattern  $i$ .

### 3.3.4 Calculating the accuracy of the generated patterns (module application 3)

As mentioned earlier, LAD is provided basically to build a classifier (discriminant function). By using this classifier the new observations will be classified as a positive or a negative observation. The performance of the classifier is measured by calculating its accuracy in classifying new observations. (Mortada et al.,2014)

In this section the k-fold cross validation method to evaluate the accuracy of LAD models is used. The k-fold method randomly partitions the dataset into k approximate equal partitions. The

ratio of positive and negative observations in the dataset should be taken into consideration while forming these partitions. Subsequently, one of those partitions is utilized as a testing set while the other  $k-1$  partitions are considered as one training set. This training set is the dataset that LAD model is learned initially. The testing set is the set used in calculating the accuracy of LAD model. This process is repeated  $k$  times and the accuracy of the model is tested for each time. Eventually, the average value of accuracy of the  $k$  experiments is considered as the accuracy of the model ( Bonates et al.,2008).

Module 3 uses the output values of module 2 for each observation in the test set. These outputs are based on the patterns that are generated in module 1, and aims to calculate the discriminant function. If the  $\Delta[obs] > 0$ ,  $\Delta[obs] < 0$ ,  $\Delta[obs] = 0$ , the corresponding observation situates in positive, negative, or unclassified class respectively. Among different formulas to calculate the accuracy of LAD model in literature, the one that is used in module 3 defines as:

$$\text{Accuracy} = 1/2 [a + e + 1/2 (c + f)] \quad (3.10)$$

where  $a$ ,  $b$ ,  $c$  ( $d$ ,  $e$ ,  $f$ ) represent the proportion of positive (negative) observations which are predicted by LAD model to be positive, negative, or unclassified respectively. (Alex et al.,2008 & Shaban et.al.,2015 & Bonates et al.,2008)

Table 3.6 : Classification Matrix (Bonates, Hammer & Kogan, 2008).

	Predictions		
	Positive	Negative	unclassified
%Positive observations	a	b	c
%Negative observations	d	e	f

For module 3 that is coded in C++ as an object oriented programming language, the pattern is introduced as a class. It is to be noted that the word class in an object oriented programming language is completely different than the word class which is used in a classification problem. In other words, the corresponding objects of this class are patterns. For this class (pattern) the number of features (attributes) in observations, weight of pattern, and the number of literals of the pattern define elements of each object. The inputs of module application 3 are the testing

dataset in addition to the patterns and their associated weights generated in modules 1 and 2. Also, ReadData and ReadPattern are the functions employed in module 3 used for reading the testing data and the generated patterns. *DisPosObs* and *DisNegObs*, employed in the algorithm of the module application 3, are the discriminant vectors corresponding to all positive and negative observations. The developed algorithm of module application 3 is shown in Figure 3.4.

### Module application 3

---

```

1: procedure calculating the accuracy of the generated pattern
2: Read all positive and negative patterns and corresponding elements by Read Pattern;
3: Read all positive and negative observations by Read Data;
4: while (there is a positive observation in test set to study) do
5:   while (there is a positive pattern to cover) do
6:     for (all j over the set of pattern features)
7:       if (observation can cover by the pattern) then
8:         update the DeltaPositive in (1);
9:       end for
10:    end while
11:   while (there is a negative pattern to cover) do
12:     for (all j over the set of pattern features)
13:       if (observation can cover by the pattern) then
14:         update the DeltaNegative in (1);
15:       end for
16:    end while
17:   Update DisPosObs;
18: end while
19: while (there is a negative observation in test set to study) do
20:   while (there is a positive pattern to cover) do
21:     for (all j over the set of pattern features)
22:       if (observation can cover by the pattern) then
23:         update the DeltaPositive in (1);
24:       end for
25:    end while
26:   while (there is a negative pattern to cover) do
27:     for (all j over the set of pattern features)
28:       if (observation can cover by the pattern) then
29:         update the DeltaNegative (1);
30:     end for
31:   end while
32:   Update DisNegObs;
33: end while
34: calculate the coefficients of a,b,c;
35: calculate the coefficients of d,e,f;
36: calculate the Accuracy of the model;
37: end Procedure

```

---

Figure 3.4 : Algorithm for module application 3

### 3.4 Implementation results of BLA model for 5 datasets

The outcomes achieved by applying the three Modules for 5 datasets of the UC Irvine Repository are presented in Table 3.6. For calculating the accuracy of the LAD model the number of positive and negative observations for each dataset is reported in columns 2 and 3. Regarding these numbers, the K for using K-fold cross validation method is chosen. Secondly, the ratio of the positive and the negative observations is calculated uniformly for all of the folds (categories). The 10-fold cross validation method is applied to partition Breast-w, Credit-a and Pima datasets, while Bupa, Hepatitis datasets are partitioned using the 5-Fold cross validation method.

In almost all of the datasets, except Hepatitis, which include a small number of binary attributes, the results show that the computational time for BLA is considerably less than cbm-LAD. This consequence is substantial in two data sets; Breast-w and Pima with large number of observations. It is comprehensible that the computational time elapsed to generate patterns for these two datasets in BLA method is almost four times less than cbm-LAD.

The Total Coverage columns show the total coverages of all positive (negative) patterns in each dataset. The outputs for these columns in BLA method are reflected to the last column. The last column lists the values of the accuracy, which entails the quality of the generated patterns. These values imply that for all datasets the patterns generated by cbm-LAD are more accurate than the patterns generated by the BLA method. On the other hand, the computational elapsed time in generating maximum  $\alpha$ -patterns by BLA method is shorter than the computational time of cbm-LAD except for the dataset Hepatitis. We can conclude that cbm-LAD performs far better than the BLA algorithm because the computational time of cbm-LAD is quite reasonable in terms of the size of these datasets, in addition to the resulted higher quality of cbm-LAD patterns as compared to the quality of the patterns generated by the BLA algorithm.

Table 3.6 The results for both BLA and cbm-LAD methods

DataSet	Observ	Binary attributes	Method	No.Patterns		Time(s)	Total coverage by		Accuracy
				Pos	Neg		Pospatts	NegPatts	
Breast-w	699	76	BLA	136	68	<b>112.853</b>	11018	23184	83.6%
			cbm-LAD	25	18	<b>545.2</b>	1306	3312	84.82%
Credit-a	690	856	BLA	294	360	<b>370.5</b>	322	480	70.23%
			cbm-LAD	74	68	<b>466.23</b>	2067	3409	83.18%
Hepatitis	155	28	BLA	1	1	<b>10.515</b>	12	143	85%
			cbm-LAD	1	1	<b>0.146</b>	12	143	93%
Bupa	345	275	BLA	138	197	<b>90.751</b>	1519	2302	61%
			cbm-LAD	51	52	<b>117.187</b>	455	681	62.28%
Pima	768	891	BLA	259	476	<b>511.994</b>	268	592	64.69%
			cbm-LAD	102	114	<b>2258.18</b>	1089	4535	68.93%

Observ : Total number of positive and negative observations

No. Patterns (pos /neg):Number of positive patterns / Number of negative patterns

Total coverage by (PosPatts /NegPatts): shows the total number of observations that are covered by the (Positive patterns /Negative patterns)

### 3.5 Conclusions

The result of Table 3.6 presents a comparison between the performance of the BLA model and cbm-LAD. It shows the computational time elapsed in generating patterns by BLA in most of the datasets is significantly less than the cbm-LAD. In two datasets with more than 200 observations the computational time elapsed in generating patterns by BLA is almost four times less than cbm-LAD. In one small dataset including less than 200 observations, the BLA computational time was long. The reason is justifiable based on the structure of the BLA model.

Secondly, the generated patterns by the BLA model in some datasets do not have a satisfactory coverage value. It results in patterns with less quality than the patterns which are generated by the cbm-LAD.

The reason of generating patterns result in less accurate LAD model than patterns of cbm-LAD is justifiable based on the structure of BLA and cbm-LAD. BLA generates maximum  $\alpha$ -patterns without any determined rule for selecting the best or at least the better  $\alpha$  among all the candidate observations for the parameter  $\alpha$ . As it was shown in Table 3.7 the number of generated patterns by BLA is really huge. So, having large number of patterns that all of them necessarily do not have high coverage value, leads to less accurate LAD model by BLA than cbm-LAD. Moreover, cbm-LAD does not have any limitation to select  $\alpha$  and generate  $\alpha$ -pattern. It can generate strong patterns in its patterns pool. As it was explained in the third stage of the example 3.1, BLA can generate a number of maximum  $\alpha$ -patterns that are exactly the same as strong patterns when a suitable  $\alpha$  is chose. But BLA cannot select the fruitful  $\alpha$  which result in generating patterns with high coverage value. Furthermore, there is not any explicit variable in the BLA model to calculate the coverage value of the generated patterns.

These weak points of the BLA model can be rectified by various methods. By modifying BLA considering its capability to generate maximum  $\alpha$ -patterns fast, achieving accurate LAD classification model comparable with cbm-LAD with less computational elapsed time is not out of mind. As one contribution, this research study, in the next chapter, attempts to utilize meta-heuristic algorithms to generate maximum patterns by considering the required modifications for BLA.

## CHAPTER 4 DEVELOPING A META-HEURISTIC ALGORITHM

### 4.1 Introduction

This chapter is intended to fulfill the objectives of this research/this study and to expand on the conclusions presented in chapter 3. This part is aimed to generate a maximum  $\alpha$ -pattern using a meta-heuristic algorithm. “A meta-heuristic refers to a master strategy that guides and modifies other heuristic to produce solutions beyond those that are normally generated in a quest for local optimality.” (Glover & Laguna, 1997). Simulated Annealing as a selected meta-heuristic is applied to solve the problem of generating a maximum  $\alpha$ -pattern. Later, the performance of the developed algorithm is evaluated by applying a number of known datasets from UC Irvin repository. Finally, the results of this evaluation are compared with the ones of available methods that were mentioned in this research study.

### 4.2 Simulated Annealing

During the development of human civilization, heuristics or meta-heuristics were the approaches that have been applied to solve the problems by trial and error. Most of significant findings were done by “thinking outside of the box” and they were frequently stochastic; this is exactly heuristic. Archimedes Eureka moment was a heuristic triumph. As a matter of fact, the daily learning experiences are frequently heuristics.

Alan Turing was the first one who applied heuristics in the Second World War. Later in 1948 he presented his ideas in machine learning, neural network and evolutionary algorithms. Two fruitful decades for developing evolutionary algorithm were the 1960's and the 1970's. In the 1980's and the 1990's there was a great enthusiasm in meta-heuristics among researchers. Developing Simulated Annealing in 1983 caused a revolution in meta-heuristic methods. (Yang, 2010).

Simulated annealing (SA) is a probabilistic method pioneered by Krikpatrick, Gelatt et Vecchi (1983) and Cenney (1985). This is a technique for approximating the global optimum of a given function that might have several local optima. Specifically, it is a metaheuristic used to approximate global optimization in a large search space. While this technique is unlikely to

find optimum solution, it can often find a very good solution, even in the presence of noisy data. This algorithm explicitly escapes from local optima by conditional acceptance for the solutions that are worse than its current solution. (Bertsimas & Tsitsiklis, 1993)

SA is a nature-inspired meta-heuristic that imitates the process of annealing in metallurgy. It aims to build the crystal structure of the materials with the lowest possible energy. This process includes heating the material to a high temperature which, for metals results in their fusion, then cooling the material gradually and slowly. This process allows making a material both softer and less brittle, and decrease the hardness of the material.

The scientific reason behind the annealing process which leads to have materials with high ductility and less hardness is explained by the question; “Why is the solid material ‘A’ broken?”. A solid material ‘A’ is broken because of its molecules energy. The higher energy of molecular structure, the more fragile ‘A’ becomes. This is exactly similar to the state of a material ‘B’ which is in an arbitrary height. This material can fall because of its potential energy. If the material ‘B’ does not have any potential energy it cannot collapse. A solid material ‘A’ cannot break easily when its molecular structure does not have any energy or it has a little energy.

The annealing process is explained in brief as follows; at the beginning of the annealing process a solid material is heated to a high temperature. In this condition the molecules move freely. The movement of the molecules results in redistributing and rooting out the irregularities within the crystalline structure. Then the temperature is gradually decreased. The process of reducing the temperature and cooling the material (metal or ceramics) is done slowly and this leads to thermodynamic equilibrium at every temperature. The process of cooling the material continues till any further reduction of the temperature no longer causes any change in the molecular structure of the material. At this state the highest stability is possible at low temperatures.

To show the role of the annealing process for SA, it is assumed that SA is applied for the given problem:

$$\left\{ \begin{array}{l} \text{Min } z = f(X) \\ s. t \\ X \in \text{solution space} \end{array} \right.$$

To solve this problem with SA, the algorithm begins with an initial solution  $s$  that is chosen heuristically or randomly. Simultaneously, the parameter  $T$  considered as temperature is initialized at a high value. At this level of temperature a new solution from the neighborhood of the initial solution is selected as  $s' \in N(s)$ . If the inequality  $f(s') < f(s)$  is satisfied, the new solution is accepted. Otherwise, the new solution is accepted based on the probability defined as  $P(f(s'), f(s), T)$ . This probability is calculated according to the Boltzmann distribution and it would be:  $P = \exp\left(-\frac{f(s')-f(s)}{T}\right)$ . (Blum & Roli, 2003)

A Boltzmann distribution is a probability distribution applied in mathematics and statistical mechanics. It is defined as;  $P(st) = e^{-\frac{E(st)}{KT}}$  such that  $E$  is the state energy that changes by changing the state.  $K$  is called the Boltzmann constant and  $T$  is the thermodynamic temperature. The relative probability of two different states named  $st_1, st_2$  with their associated energy  $E(st_1)$  and  $E(st_2)$  is calculated as:  $P(\text{relative}) = \frac{P(st_1)}{P(st_2)} = \frac{e^{-\frac{E(st_1)}{KT}}}{e^{-\frac{E(st_2)}{KT}}} = e^{-\frac{\Delta E}{T}}$  and it is considered as the Boltzmann Factor. This Factor is applied to calculate the probability of accepting the unsatisfactory solutions for SA. Changes between energy states are associated to changes in the objective function value of Simulated Annealing and each state of the molecular structure ( $st$ ) is corresponds to the one solution ( $s$ ) in SA. The mutual relation of thermodynamic simulation and SA is explained by the Table 4.1.

Table 4.1 : The relation of Thermodynamic simulation and Combinatorial Optimization

Thermodynamic simulation	Combinatorial Optimization
System States	Solutions
Energy	Cost
Change of State	Neighboring Solutions
Temperature	Control Parameter T
Frozen State	Heuristic solution

At the beginning of the process of SA the temperature ( $T$ ) is considered at a high level. Based on  $P = \exp(-\frac{f(s')-f(s)}{T})$  which is the Boltzmann factor for the objective function  $f$ , the probability ( $P$ ) of accepting a new solution that is worse than the initial solution is high and is close to the value equal to one in high temperatures. In SA to test the acceptance condition for new solutions based on the calculated probability, a random number ( $rand$ ), having uniform distribution in the interval  $(0, 1)$ , is selected. If the inequality  $(rand) \leq P$  is satisfied, then the new solution is selected. This way is applied since ( $rand$ ) follows a uniform distribution, the probability for it to be less than  $P$  is equal to  $P$ . (if  $X \sim U(0,1) \Rightarrow F(x) = P(X \leq x) = x$ )

Choosing worse solutions in high temperatures results in increasing the exploration of search space during the first iterations of the algorithm. In exploration phase the algorithm attempts to increase the size of the searched space. This phase is associated to the global search. When the temperature is fixed the probability of selecting the new solution depends on  $(f(s')-f(s))$ . At this fixed temperature, having new solutions worse than the old one reduces the probability of choosing the new solution. Then the temperature is gradually reduced. Decreasing the temperature leads to reducing the probability of choosing new solutions that are worse than the old solutions. By reducing the temperature, the algorithm moves to the exploitation. In fact exploration searches for new and unknown regions in the search space. Whereas exploitation by utilizing the knowledge obtained at areas visited previously, attempt to find the best solution which is the local optimum (Chen, Xudiera & Montgomery, 2012).

Starting the algorithm at a high temperature and decreasing the temperature gradually, which leads to reducing the chance of having unsatisfactory solutions and helps the algorithm converge to a near optimal solution, is the intelligence of the Simulated Annealing algorithm. The method by which the temperature is reduced, or the cooling schedule, is very important to achieve a good result with this algorithm. The algorithm has to make a balance between exploration and exploitation (diversification versus intensification). It has been shown that by choosing cooling schedule as  $T_{m+1} = \frac{\beta}{\log(m+m_0)}$  such that  $m_0$  is a constant value and  $\beta \in IR$ , obtaining the global optimum is guaranteed. But this method is not applicable in the implementation domain because it takes a lot of time to finalize, while the other method that is applied a lot for practical goals is:  $T_{m+1} = \theta T_m$  such that  $0 \leq \theta \leq 1$  (Blum & Roli, 2003).

The pseudo-code of the Simulated Annealing algorithm is shown in Figure 4.1. (Yang, 2010) Simulated annealing is one of the noteworthy algorithms among mathematicians, because its convergence to an optimal solution under specific conditions has been proved (Bertsimas & Tsitsiklis, 1993). The figurative smooth annealing and observing various states for infinite times ensures the optimality of simulated annealing (Dorigo & Stutzle, 2004).

### Simulated Annealing Algorithm

---

```

1: Procedure simulated Annealing
2: Defining the objective function  $f(x)$  such that  $x$  is an  $n$  dimensional vector
3: Initialize the temperature as  $T_0$  and initial random solution  $x_0$ .
4: Set final temperature  $T_f$  and maximum number of iterations  $M$  in each temperature
5: Define cooling function;  $T \leftarrow \theta T$  such that  $0 < \theta < 1$ 
6:   While ( $T > T_f$  and  $m < M$ )
7:     Select new solution  $x_{n+1}$  randomly
8:     Calculate  $\Delta f = f_{n+1}(x_{n+1}) - (x_n)$ 
9:     If (the new solution is better than the previous one) do
10:       Accept the new solution
11:     else
12:       choose a random number rand
13:       if ( $(\exp(-\Delta f/T) > rand)$ ) do
14:         Accept new solution
15:       end if
16:     end if
17:     Update the best solution  $x^*$  and the best value of the objective function as  $f^*$ 
18:      $m = m + 1$ 
19:   end while
20: end procedure

```

---

Figure 4. 1: Simulated Annealing algorithm

### 4.3 A developed pattern generation algorithm:

In the following section, a pattern generation algorithm using the structure of Simulated Annealing is developed. This algorithm attempts to remove the weaknesses of the BLA model

introduced in chapter 3. This algorithm is nominated as SA-BLA which stands for “Simulated Annealing of Best Linear Approximation”.

The aim of the SA-BLA is generating maximum  $\alpha$ -pattern. Based on the definition of maximum positive  $\alpha$ -pattern this algorithm is planned to generate a pattern that has the maximum coverage among all the patterns that can cover observation  $\alpha \in \{0,1\}^n \setminus \Omega^-$  such that  $\Omega^-$  is the negative set of observations. To achieve this goal a maximization objective function is applied to generate a pattern. Due to the selected  $\alpha$ , the algorithm generates positive (negative) pattern that covers as many positive (negative) observations as possible for the corresponding dataset. Selecting  $\alpha$  continues until all the observations are covered by the generated patterns. Patterns generated by SA-BLA as its optimal solutions are shown differently from the ones of the BLA model. Optimal solutions of the BLA model are binary vectors. While in SA-BLA the patterns are the vectors of 0, 1, and 2 with a dimension equal to the total number of attributes.

This characteristic differentiates structure of SA-BLA from other algorithms to generate maximum  $\alpha$ -pattern. If  $P = b_2 \overline{b_3} b_4$  is a pattern of the set of observations with five attributes, it is defined with a vector of five components for the SA-BLA as  $P = (2, 1, 0, 1, 2)$ . The value 2 shows that the pattern does not include literals  $b_1$  and  $b_5$ , while value 1 shows that this pattern includes literal  $b_2$  and  $b_4$  and it can cover binary observations that have a value of 1 in corresponding attributes. The value 0 shows that the pattern includes literal  $\overline{b_3}$  and it can cover binary observations that have a value of 0 in their corresponding attribute. So, the pattern is free in attributes  $b_1$  and  $b_5$  but it said to be captive in other attributes.

The pseudo-code of SA-BLA is shown in Figure 4.2 and its structure is based on four nested loops. The order of the loops from the innermost to the outermost is respectively as follows: 1. feasible solution loop, 2. neighborhood loop, 3. temperature loop, 4. coverage loop. A concise description of this algorithm is explained in the next subsection.

### 4.3.1 SA-BLA algorithm

**Step1:** At first, one of the observations is selected randomly as  $\alpha$ . The selected observation is considered as initial solution and assumed as best solution. Choosing such kind of initial solution along with the definition of *neighborhood* function guarantees that eventually an

$\alpha$ -**pattern(s)** is generated and has approximately the maximum coverage for the observations of its associated class. This step, as for all of the following steps, is done in the outermost loop named the *coverage loop*.

**Step 2:** In this step a pre-calculation has to be done before starting the inner loops. This is performed to find a suitable neighborhood solution by applying the *neighborhood* function. This function will be explained further in the following section. To do so, the difference between every positive observation and  $\alpha$  is calculated. This is obtained by calculating the difference between corresponding components that belong to both positive observation and  $\alpha$ . The final results are transferred to (*difference  $\alpha$ -pos*) vector with a dimension equal to the number of positive observations.

**Step 3:** Here the temperature is initialized. This initialization is done by the value equal to 1000 for the temperature. Then the number of iterations for the *temperature loop* and *neighborhood loop* are determined respectively. SA-BLA proceeds then to the *neighborhood loop*.

In *neighborhood loop* the initial solution is applied and it generates neighbor solutions with the *neighborhood* function. The number of iterations of this loop shows the *maximum* number of feasible neighborhood solutions that can be built. Every neighborhood solution has to satisfy the constraints of the problem, which means that the positive (negative) pattern does not cover any negative (positive) observations.

The *check-constraint* function checks if the neighborhood solution can satisfy the constraints or not. If the neighborhood solution *could* satisfy the constraints then three values are calculated; 1. The objective function value for this solution *or* the number of observations covered by this solution, 2. Identification of all the observations covered by this solution, 3. Difference between the values of the objective function for the initial solution and those of its neighborhood as  $\Delta f$ . If the neighborhood solution *could not* satisfy the constraints, for the maximum number of iterations  $G$ , the process of searching a feasible solution continues by *feasible solution loop* which is the innermost loop. If, ultimately, no new neighborhood solution which satisfies the constraints is found then  $\alpha$  is considered as the feasible neighborhood solution.

When all the iterations of the *neighborhood loop* are finished the solution with minimum cost which is the pattern with maximum coverage will be chosen.

This step reveals the contribution of SA-BLA relative to SA. SA-BLA generates a number of neighborhood solutions instead of just one. This method differentiates SA-BLA as a population-based search algorithm from SA which is a single point search algorithm.

**Step 4:** In this step which is done after finishing *neighborhood loop*, Simulated Annealing conditions are evaluated. Regarding to the Simulated Annealing conditions every solution selected by the *neighborhood loop* is replaced with the best solution if it costs less than the best solution. Otherwise, this solution based on the Boltzmann Factor is chosen with an assumed probability.

**Step 5:** After passing all previous steps and choosing a new solution, the temperature, which was initialized at the beginning and defined by the iterations of the third inner loop, is now reduced. SA-BLA algorithm continues with the best available solution. The temperature gradually decreases in any iteration of the third inner loop. This loop is finished when the temperature is equal to or less than  $T_f$  which is defined as the final temperature. At the end of the third inner loop considered as an iteration of SA-BLA, the best solution selected is saved in patterns pool.

**Step 6:** At this step, by applying the “Remove-covered-observation” function all the observations covered by the selected pattern in step 5 are removed. The number of observations that cannot be covered is transferred to next step.

**Step 7:** finally, the outer loop of SA-BLA is applied. The number of uncovered observations is used as a stopping criterion for this loop. If the number of observations uncovered by the generated patterns is equal to zero (all the observations are covered by the patterns pool) this loop is finished and the SA-BLA algorithm is finalized.

### 4.3.2 Applied functions in SA-BLA

SA-BLA is composed of four functions. These functions will be called during the execution of SA-BLA. The first function is; *costf*. The inputs of this function is the set of positive (negative) observations and a positive(negative) pattern generated by SA-BLA. *Costf* counts the number of

observations covered by the corresponding pattern. This value is shown by *pat-cov* in the pseudo-code of SA-BLA and considered as the first output of *costf*. Moreover, this function returns a *coverobs* vector. The dimension of this vector is equal to the total number of positive (negative) observations to determine the observations that can be covered with the associated positive (negative) pattern. The entities of this vector are initialized at value of one and if the pattern cannot cover any observation then the corresponding entity of this observation in the *coverobs* vector changes to zero. Finally the sum of these values returns the coverage of the related pattern.

The second function is *check-constraint*, whose objective is checking whether the constraints of the model are satisfied or not. The constraints are built based on the constraint of the BLA model. These constraints guarantee that any generated pattern for a specific class do not cover any observations from its opposite class. This function has two inputs. A positive (negative) pattern generated by SA-BLA and a set of negative (positive) observations that should not be covered by the corresponding pattern. The *check-constraint* function manages its objective by two *in-captive* and *constraint* vectors. The *in-captive* vector defines which literals in the pattern should include zero or one. Based on the definition of the pattern for SA-BLA these literals are not free or they are captive. The constraint vector does the logical comparison between the values of the positive (negative) pattern's literals, defined by *in-captive* vector, and the corresponding attributes for every positive (negative) observation.

For example, it is assumed that  $P = P_1P_2P_3P_4P_5$  a positive pattern for a set of observations composed of 5 attributes. Two Matrixes  $M = [pos]_{3 \times 5}$  and  $N = [neg]_{10 \times 5}$  are considered as sets composed of 3 positive and 10 negative observations. A vector *in-captive* = [1 3 5] shows that  $P$  has a value equal to 0 or 1 for *literal*  $P_1$  and  $P_3$  and  $P_5$  and equal to 2 for the others. The *constraint* vector's dimension will be equal to the number of negative observations. Assuming that  $C$  is a *constraint* vector such that  $C = (c_1, c_2, c_3 \dots, c_{10})$  and  $c_i \in \{0, 1\}$  are the components of this vector. If, for a given value of  $i$ ,  $c_i = 1$  is satisfied, then the corresponding observation can be covered by the pattern. Also  $c_i = 1$  if and only if ( $P_1 = neg_{i,1}$  and  $P_3 = neg_{i,3}$  and  $P_5 = neg_{i,5}$ ). So, this solution is infeasible because the generated positive covers a negative observation. Consequently, the SA-BLA does not select it for its patterns pool.

The third function is *Remove-covered-observation*, whose objective is removing observations from the positive(negative) set that are covered by the selected positive(negative) pattern .

This function guarantees that at the end of the algorithm there will be a set of patterns (positive/negative) in the patterns pool that cover all the observations (positive/negative) .This function has two inputs and two outputs. The inputs of *Remove-covered-observation* are the *coverobs* vector which is an output of the *costf* function and a set of positive (negative) observations. The *coverobs* vector defines the positive (negative) observations covered by the positive (negative) pattern generated in any iteration of SA-BLA. The *Remove-covered-observation* by using *coverobs* removes all of the covered observations simultaneously. New positive (negative) set of observations are replaced with the old one. This new set plus the number of uncovered observations as outputs of this function are used to finalize the execution of SA-BLA. The execution of SA-BLA is over when the number of uncovered observations equals zero.

The Last function is the *neighborhood* function. This function attempts to produce a new solution (pattern) considered as neighborhood of the initial solution. The *neighborhood* function generates a new  $\alpha$ -pattern, compatible with  $\alpha$ , such that it can cover as many positive observations as possible. This aim is satisfied by *difference pos- $\alpha$*  vector calculated at the beginning of the SA-BLA algorithm. The inputs of the *neighborhood* function are; initial  $\alpha$ -pattern (solution), set of positive observations and *indexm* which is the index of a positive observation with the highest number of differences with  $\alpha$ . By applying such kind of positive observation, the function attempts to generate a new solution (pattern) which includes as few as possible of *in-captive* literals. This method results in increasing the number of observations covered by the new solution. To do so, as it is showed in Figure 4.5, a peer to peer comparison between the attributes of the selected positive observation (with the specification of *indexm*) and literals of the initial solution is done. Then for the indexes with different value for two mentioned vectors the value of the new solution is put as equal to 2 and the remaining literals of the new solution has the same value as the initial one. By this method the *neighborhood* function returns a suitable neighborhood for its next movement.

The pseudo-code of SA-BLA algorithm and its associated functions, by applying Matlab Programming Language, are presented as follows:

**SA-BLA Algorithm**


---

```

1: Procedure SA-BLA (positiveset, negativeset)
2: while ( numuncovered  $\neq$  0) do
3:   randomly choose one of the positive observations and consider it as parameter alpha;
4:   best pat  $\leftarrow$  consider alpha as an initial pattern (solution) and call it pat;
5:   best pat-cov  $\leftarrow$  calculating the coverage of the pat for the positiveset with costf , call it pat-cov
6:   best pat-covpos  $\leftarrow$  defining the observations that are covered by pat with costf ,call it pat-covpos
7:   initialize ( difference alpha-pos) vector with zero
8:   //for ( $\forall i \in$  positiveset) do
9:     difference alpha-pos(i)  $\leftarrow$  ((abs(positiveset(i) – pat))
10:  end for
11:  (m,indexm)  $\leftarrow$  max(difference alpha-pos) // (related to neighborhood function)
12:  Initialize temperature with  $T_0$ 
13:  While ( $T > T_i$ ) do
14:    for (it < NumIter= 25) do
15:      0  $\leftarrow$  g
16:      npat  $\leftarrow$  neighbor(pat, positiveset, indexm)
17:      check  $\leftarrow$  check-constraint(npat)
18:      while (check = 0,  $g < G=10$  ) do
19:        npat $\leftarrow$ neighbor(pat, positiveset, indexm)
20:      end while
21:      if (check = 0 ,  $G = g$ ) then
22:        npat  $\leftarrow$  pat
23:      end if
24:      (npat-cov, npat-covpos)  $\leftarrow$  costf (positivetest,npat)
25:      Deltaf(it)  $\leftarrow$  npat-cov(it) – pat-cov(it)
26:    end for
27:    (maxdeltaf , it)  $\leftarrow$  max(Deltaf)
28:    rand  $\leftarrow$  choose a random number
29:    if ( maxdeltaf > 0 OR exp(maxdeltaf/T) > rand )
30:      pat  $\leftarrow$  npat(T)
31:      pat-cov  $\leftarrow$  npat(T)-cov
32:      pat-covpos  $\leftarrow$  npat(T)-covpos
30:    end
31:    if ( pat-cov > bestpat-cov) then
32:      bestpat $\leftarrow$  pat
33:      bestpat-cov  $\leftarrow$  pat-cov
34:      bestpat-covpos $\leftarrow$ pat-covpos
35:    end if
36:    T  $\leftarrow$  alpha * T
37:  end while
38:  bestpatternpool $\leftarrow$ bestpat
39:  (newpositiveset,numuncovered) $\leftarrow$ Remove-covered-observation(positiveset,bestpat-covpos);
40:  positiveset  $\leftarrow$  newpositiveset
41: end while
42: end procedure

```

---

Figure 4.2 : Pseudo code of SA-BLA algorithm

### Costf Algorithm

---

```

1. Function ( cost, coverobs) ← costf (positiveset, pattern)
2. Initializing all the elements of the cover vector at one
3. for (every positive observation from positiveset)
4:   if (pattern cannot cover the positive observation) then
5:     Update the associated element of cover for this positive observation to zero
6:   end if
7: end for
8: cost ← sum of all components of cover vector
9: end function

```

---

Figure 4.3 : Pseudo code of costf function

### Check-constraint Algorithm

---

```

1: Function check←check-constraint (negativeset, pattern)
2: initialize the constraint vector at zero;
3: initialize variables check and k at one
4: for ( $\forall j \in \text{set of attributes}$ ) do
5:   if ( pattern(j) < 2) then
6:     incaptive(k) ← j
7:     k ← k+1
8:   end if
9: end for
10: if (incaptive ≠ 0) then
11:   for ( $\forall i \in \text{negative observat}$ ) do
12:     if (negativetest (i,incaptive)) = pattern(incaptive)) do
13:       constraint (i) ←1
14:       break
15:     end if
16:   end for
17:   if (sum of all components of constraint =1) then
18:     check=0
19:   end if
20: else
21:   check=0
22: end if
23: end function

```

---

Figure 4.4 : Pseudo code of check - constraint function

### Neighborhood Algorithm

---

```

1: Function npattern ← neighborhood (pattern, positiveset, indexm)
2: npattern ← pattern
3: for ( $\forall j \in \text{set of attributes}$ ) then
4:   if ( $(j) \neq \text{positiv}(\text{indexm}, j)$ ) then
5:     if ( $\text{rand} < k$ ) then
6:       npattern(j) ← 2
7:     end if
8:   end if
9: end for
10: end function

```

---

Figure 4.5 : Pseudo code of neighborhood function

### Remove-covered-observation Algorithm

---

```

1: Function (newpositiveset, numuncovered) ← Remove-covered-observation (positiveset, coverobs)
2: initialize the newpositiveset with positiveset
3: for (all positive observations whose corresponding components in cover vector is =1) do
4:   (remove the associated positive observation from the positiveset )
5: end for
6: positiveset ← Update the newpositiveset
7: numuncovered ← Update the numuncovered
8: end Function

```

---

Figure 4.6 : Pseudo code of Remove - covered – observation

## 4.4 Calculating Accuracy

The formula adopted to calculate the accuracy of the patterns generated by the SA-BLA algorithm is exactly the same as the one presented in chapter 3 to calculate the accuracy of patterns generated by the BLA method.

By using the  $K$ -fold cross validation method, which was explained completely in chapter 3, and applying the (3.10) formula, the accuracy is calculated for all the datasets. The 10-fold cross validation method is applied to partition Breast-w, Credit and Pima, while Bupa and Hepatitis are partitioned using the 5-Fold cross validation method. As it was mentioned in chapter 3, for the 10-fold and 5-fold cross validation methods the corresponding datasets are randomly partitioned respectively into 10 and 5 approximately equal partitions. This partitioning is done based on the number of observations. Then, one of the partitions is used as test set and the others are applied as train set. This process is repeated 10 and 5 times (for 10-fold and 5-fold cross validation).

Final accuracy is calculated based on the average of the calculated accuracy for all repetitions. The size of the train and test sets used by the  $K$ -fold cross validation method for the considered datasets are shown in Table 4.1. The pseudo code of calculating weight and coverage of generated patterns by SA-BLA is presented in Figure 4.7. Also, the pseudo code of the algorithm to calculate accuracy is shown in Figure 4.8. *DisPosObs* and *DisNegObs*, employed in Figure 4.8, are the discriminant vectors corresponding to all positive and negative observations. *DeltaPositive* and *DeltaNegative* are defined based on the (3.7). The variables  $a, b, c, d, e, f$  are considered based on (3.10).

Table 4.2 : Size of train and test set for assumed datasets(1)

Data sets	K-fold	Train set	Test set
Breast-w	10-fold	630	69
Pima	10-fold	8 repetitions : 691 / 2 repetitions : 692	8 repetitions : 77 / 2 repetitions : 76
Credit	10-fold	621	69
Bupa	5-fold	276	69
Hepatitis	5-fold	112	31

### Algorithm to calculate weight and coverage of generated patterns

---

- 1: **procedure** providing tools to calculate discriminant function
- 2: Read all positive and negative patterns by Read Patterns;
- 3: Read all positive and negative observations by Read Data;
- 4: **while** (there is a positive (negative) pattern to investigate) **do**
- 5:     **while** (there is a positive (negative) observation to read) **do**
- 6:         **if** (positive (negative) observation is covered by positive (negative) pattern) **then**
- 7:             add value to the total coverage of positive (negative) pattern;
- 8:         **else if** (take another observation)
- 9:     **end while**
- 10: updating the *coverages vector* of positive (negative) patterns;
- 11: calculate the weight of each pattern;
- 12: updating the *weights vector* of positive (negative) patterns
- 13: **end while**
- 14: print out *weights vector* as output
- 15: **end procedure**

---

Figure 4.7 : Pseudo code to calculate weight and coverage vectors

### Algorithm to calculate the accuracy of the generated patterns

---

```

1: procedure calculating the accuracy of the generated pattern
2: for i=1:Number of positive test observation do
3:   for j=1:Number of positive pattern do
4:     if observation(i) can cover by the positive pattern(j) do
5:       update the DeltaPositive(i);
6:     end if
7:   end for
8:   for j=1:Number of negative pattern do
9:     if observation(i) can cover by the negative pattern(j) do
10:      update the DeltaNegative(i);
11:    end if
12:  end for
13: Update DisPosObs;
14: end for
15: for i=1:Number of negative test observation do
16:   for j=1:Number of positive pattern do
17:     if observation(i) can cover by the positive pattern(j) do
18:       update the DeltaPositive(i);
19:     end if
20:   end for
21:   for j=1:Number of negative pattern do
22:     if observation(i) can cover by the negative pattern(j) do
23:       update the DeltaNegative(i);
24:     end if
25:   end for
26: Update DisNegObs;
27: end for
28: calculate the coefficients of a,b,c;
29: calculate the coefficients of d,e,f;
30: calculate the Accuracy of the model;
31: end Procedure

```

---

Figure 4.8 : Pseudo code to calculate accuracy

#### 4.5 Performance evaluation of SA-BLA algorithm:

In this section the performance of SA-BLA is evaluated using 5 datasets of U.C Irvine Repository. The outputs of three cbm-LAD, BLA and SA-BLA methodologies for respectively Breast-w, Bupa, Hepatitis, Pima and Credit as selected datasets are shown in Table 4.2. Computational time and accuracy are considered as two comparison criteria. The number of generated patterns for each method is considered to make the priorities clearer. The specifications of the CPU which is applied are as follows: The processor is an Intel® Core™ i5 540 M and the size of RAM and Cash memories are respectively 6 GB and 3MB.

Table 4.3: Results of comparisons among three methods

DataSet	observations		Binarized attributes	Method	No.Patterns		Time(s)	Accuracy
	Pos	Neg			Pos	Neg		
Breast-w	241	458	76	BLA	136	68	112.853	83.6%
				cbm-LAD	25	18	545.2	84.82%
				SA-BLA	<b>20</b>	<b>15</b>	<b>29.47</b>	<b>93.3%</b>
Bupa	200	145	275	BLA	138	197	80.31	61%
				cbm-LAD	51	52	117.187	62.28%
				SA-BLA	<b>45</b>	<b>45</b>	<b>61.85</b>	<b>64.44%</b>
Hepatitis	12	143	28	BLA	1	1	13.188	85%
				cbm-LAD	1	1	<b>0.144</b>	<b>93%</b>
				SA-BLA	1	1	0.32	90%
Pima	268	500	891	BLA	294	476	511.98	64.69%
				cbm-LAD	102	114	2258	68.93%
				SA-BLA	<b>97</b>	<b>98</b>	<b>110.23</b>	<b>73.33%</b>
Credit	307	383	856	BLA	296	360	370.5	70.23%
				cbm-LAD	74	68	466.5	<b>83.18%</b>
				SA-BLA	<b>72</b>	<b>72</b>	<b>204.96</b>	78.53%

#### 4.6 A review on results

As it is shown in Table 4.2, the evaluation is done in-terms of time and accuracy among available and developed methods. The number of generated patterns is mentioned to help clarifying the respective priorities of each method. For the first dataset which is Breast-w with 699 observations and 76 attributes, SA-BLA has the best performance. The number of patterns generated by SA-BLA is the lowest for both positive and negative patterns among the three methods. The computational time and accuracy respectively has the lowest and highest value for SA-BLA in comparison with the other methods. For the Bupa, the second dataset having 345 observations with 275 attributes, SA-BLA is the best method. It generated fewer patterns than the two other methods, there are here especially fewer patterns than for BLA and marginally fewer patterns than for cbm-LAD, in the lowest computing time and with the highest accuracy. For the third dataset, which is Hepatitis with the lowest number of observations and attributes among other datasets, the performance of SA-BLA is not notable. This result is not improbable since the meta-heuristics is designed to perform well for large datasets. For the Pima, as the other dataset which includes 768 observations with 891 attributes, SA-BLA has the best performance in comparison with other methods. It could cover all observation with less patterns than cbm-

LAD. The computational time and accuracy respectively has the lowest and highest value than the two other methods. In the last dataset called Credit, formed of 690 observations and 856 attributes, the total number of patterns generated by SA-BLA is competitive with cbm-LAD and fewer than BLA. In addition SA-BLA in terms of the computational time has the first rank and the accuracy of the model built by SA-BLA has a difference of less than 5% compared to cbm-LAD which is ranked first.

## 4.7 Conclusions

In this chapter a metaheuristic algorithm based on the Simulated Annealing for generating maximum  $\alpha$ -pattern was presented. This algorithm applied population based search method instead of the single point search. While SA, originally, uses the single point search method. Performance of this algorithm is validated by the application of 5 datasets compatible with the structure of the problem.

The results demonstrate that SA-BLA methodology has the best performance among other methods in-terms of computational time. The Hepatitis is the only dataset with higher computational time for SA-BLA. This result is justifiable by considering the size of this dataset. This dataset has the smallest number of observations and attributes in comparison with the others, while metaheuristics are fruitful for large datasets. Moreover, accuracy as the second criterion for evaluating the performance of methods is calculated. The outputs show that for all datasets except Hepatitis and Credit, the accuracy of the SA-BLA method is the highest and considerable.

The cbm-LAD, as available software in the Mathematics and Industrial Engineering department of Ecole Polytechnique Montreal, was invented in 2012. This model is capable to generate any kind of pattern like strong patterns (Yacout et al., 2012). So, the cbm-LAD is not limited to generate a specific kind of pattern, while the SA-BLA method is configured to generate just one kind of pattern which is maximum  $\alpha$ -pattern. The limitation of generating just one kind of pattern plus having patterns compatible with one specific observation as  $\alpha$ , leads to difficult conditions for SA-BLA to generate accurate patterns in comparison with cbm-LAD. (Hammer & Bonates, 2006)

## **CHAPTER 5      NUMERICAL RESULTS; COMPARISON AND ANALYSIS**

### **5.1 Introduction**

This chapter develops several performance evaluations of available and developed methods for pattern generation. The Friedman test a nonparametric statistical test compatible with pattern generation problem, is applied. This statistical test is used to evaluate two performance criteria of pattern generation algorithms: computational time and accuracy. The outputs of the Friedman tests, obtained by the SPSS software, demonstrate the performance of the SA-BLA method of maximum  $\alpha$ -patterns generation in comparison with available methods of pattern generation.

### **5.2 Numerical results**

In this section, twenty datasets are considered in order to evaluate the performance of three presented pattern generation methods BLA, cbm-LAD, SA-BLA. The first 5 datasets are from UC Irvine Repository and the remaining datasets are generated randomly by Matlab Software to increase the data pool. The characteristics of the selected datasets are presented in Table 5.1, and Table 5. 2 and they sum up the computing elapsed time and accuracy of the three methods for each dataset.

Table 5.1: Datasets selected

Test	Positive	Negative	Binary	Data set
1	241	485	76	Breast-w (UC Irvine Repository)
2	145	200	275	Bupa(UC Irvine Repository)
3	12	143	28	Hepatitis (UC Irvine Repository)
4	268	500	891	Pima(UC Irvine Repository)
5	307	383	856	Credit (UC Irvine Repository)
6	30	50	25	Random (by Matlab)
7	70	60	40	Random (by Matlab)
8	85	100	65	Random (by Matlab )
9	120	90	30	Random (by Matlab)
10	150	110	50	Random (by Matlab)
11	190	200	80	Random (by Matlab)
12	250	230	100	Random (by Matlab)
13	300	310	120	Random (by Matlab)
14	350	250	150	Random (by Matlab)
15	380	230	175	Random (by Matlab)
16	415	380	180	Random (by Matlab)
17	475	400	200	Random (by Matlab)
18	500	545	200	Random (by Matlab)
19	535	570	400	Random (by Matlab)
20	600	650	800	Random (by Matlab)

Table 5.2 : Respective performance of cbm-LAD, BLA, SA-BLA

Test	Solution time (S)			Accuracy (%)		
	cbm-LAD	BLA	SA-BLA	cbm-LAD	BLA	SA-BLA
1	545.20	112.85	24.47	84.82	83.60	93.3
2	117.18	80.31	61.85	62.28	61	64.44
3	13.18	0.14	0.32	85	93	90
4	2258	511.95	110.03	64.69	68.93	73.33
5	466.50	370.50	204.96	83.18	70.23	78.53
6	12.21	2.13	8.91	87.27	83.91	82.25
7	28.33	25.13	87.57	91.41	87.33	90.02
8	98.43	83.30	65.12	83.12	79.89	81.43
9	141.78	100.21	81.30	81.64	78.11	74.23
10	169.43	171.34	154.21	77.23	79.34	81.23
11	110.21	113.45	95.77	68.33	75.45	71.21
12	173.21	151.19	163.87	80.65	91.24	83.21
13	200.32	234.18	210.27	81.21	79.12	78.21
14	181.43	121.56	178.33	93.41	90.56	92.34
15	287.67	208.43	181.56	82.01	81.90	78.45
16	294.21	267.41	202.93	91.11	90.21	86.78
17	401.34	281.34	293.33	78.21	81.34	84.23
18	509.56	510.34	320.33	87.11	78.56	80.21
19	704.67	753.89	388.31	80.20	80.89	79.33
20	1002.32	810.57	415.32	78.23	67.34	75.23

### 5.3 Comparison and Analysis

In this part, the Friedman Rank test is applied to do a performance comparison of the three mentioned methods of pattern generation. The Friedman Rank test is a statistical nonparametric method. To use this test,  $K$  variables are evaluated by  $N$  samples. Finally, the outputs determine whether or not there is a difference among the rank mean of these variables. Finally, this test shows the rank of each variable when their respective rank means differ. (Friedman,1940). Applying the Friedman rank test does not require any assumption of normality or independency of the values. This is the advantage of this two-tailed test that is used in ranking, since for ranking a number of variables, usually the values of each variable are neither normal nor independent. (Conover, 1999; Friedman, 1937)

The methodology of the Friedman statistical test, used to evaluate the rank mean of  $K$  variables for  $N$  samples, is summarized as follow:

- a. All the values of each sample for  $K$  variables are ranked. The variables are ranked based on their priorities from  $1 \dots K$ .  
The  $x_{i,j} \in \{1,2,3,\dots,K\}$  such that  $i = 1,2,\dots,K$  ,  $j = 1,2,3,\dots,N$  show the rank of variable  $i$  in sample  $j$ .
- b. The following statistic value is calculated :

$$F = \frac{12}{N \cdot K \cdot (K + 1)} \sum_{i=1}^K \left( \sum_{j=1}^N x_{i,j} \right)^2 - 3 \cdot N \cdot (K + 1)$$

This is a *Chi Square* distribution with  $(k-1)$  degrees of freedom

- c. At this step the significance level of  $\Upsilon^1$  is determined. The null-hypothesis in Friedman test expresses that there is no statistical difference among the ranks mean of the variables. If *Friedman statistic value* or  $F$  is bigger than critical

---

<sup>1</sup>Here to avoid confusion the symbol  $\Upsilon$  is used to represent the significance level instead of  $\alpha$

point of Chi Square distribution then with confidence level of  $(1 - \gamma)\%$  the variables rank are different and a significance ranking among variables can be done.

To compare the three mentioned methods for solving the pattern generation problem, a number of random datasets are generated plus the five ones of UC Irvine Repository. All of them are solved by the three methods. The extra datasets are generated with the mind that the validity of any statistical test is improved by increasing the number of statistical samples.

In order to rank the mentioned methods in terms of computing time with the Friedman test, at first, the solution time of SA-BLA, BLA and cbm-LAD are calculated for each dataset. Then, the ranking process is done based on their importance for the corresponding criterion. The Friedman statistical test for this problem is done for  $K=3$  variables and  $N=20$  samples. In each test the method with the lowest computing time has the rank 1 and the one with the highest computing time has the rank 3.

Similarly, this process is done for ranking the samples by their accuracy values, as a second evaluation criterion. Here, rank 1 corresponds to the highest accuracy, and thus the best result. Table 5.4 and Table 5.5 respectively show the rank of all methods in terms of computing time and accuracy. At this step, in the same ways that for chapters 3 and 4, the k-fold cross validation method and formula (3.10) are applied to calculate the accuracy. The size of the train and test sets in all partitions of the k-fold cross validation method for the introduced samples are shown in The information presented in Table 5.4 and Table 5.5 are considered as inputs of the SPSS software. Using the Friedman statistical test enables to calculate the rank of cbm-LAD, BLA and SA-BLA in terms of computing time and accuracy.

Table 5.3 Size of train and test set for assumed datasets (2)

Dataset	K-fold	Train set	Test set
Breast-w	10-fold	630	69
Pima	10-fold	8 repetitions : 691 / next 2 repetitions : 692	8 repetitions : 77 / next 2 repetitions : 76
Credit	10-fold	621	69
Bupa	5-fold	276	69
Hepatitis	5-fold	112	31
Random 1	5-fold	64	16
Random2	5-fold	104	26
Random3	5-fold	148	37
Random4	5-fold	136	34
Random5	5-fold	208	52
Random6	10-fold	351	39
Random7	10-fold	432	48
Random8	10-fold	549	61
Random9	10-fold	630	70
Random10	10-fold	549	61
Random11	10-fold	5 repetitions: 715 / next 5 repetitions: 716	5 repetitions: 80 / next 5 repetitions: 79
Random12	10-fold	5 repetitions: 788 / next 5 repetitions: 787	5 repetitions: 87 / next 5 repetitions: 88
Random13	10-fold	5 repetitions: 941/ next 5 repetitions: 940	5 repetitions: 104/next 5 repetitions: 105
Random14	10-fold	5 repetitions: 995 / next 5 repetitions: 994	5 repetitions: 110/next 5 repetitions: 111
Random15	10-fold	1125	125

Table 5.4 : Rank of cbm-LAD, BLA and SA-BLA in terms of computational time

Test Problem	cbm-LAD	BLA	SA-BLA
1	3	2	1
2	3	2	1
3	3	1	2
4	3	2	1
5	3	2	1
6	3	1	2
7	2	1	3
8	3	2	1
9	3	2	1
10	2	3	1
11	2	3	1
12	3	1	2
13	1	3	2
14	3	1	2
15	3	2	1
16	3	2	1
17	3	1	2
18	2	3	1
19	2	3	1
20	3	2	1

Table 5.5 : Rank of cbm-LAD, BLA and SA-BLA in terms of accuracy

Test Problem	cbm-LAD	BLA	SA-BLA
1	2	3	1
2	2	3	1
3	3	1	2
4	3	2	1
5	1	3	2
6	1	2	3
7	1	3	2
8	1	3	2
9	1	2	3
10	3	2	1
11	3	1	2
12	3	1	2
13	1	2	3
14	1	3	2
15	1	3	2
16	1	2	3
17	3	2	1
18	1	3	2
19	2	1	3
20	1	3	2

### 5.3.1 Ranking in terms of computing time

At this phase, to rank the methods according to their respective computing times, the null-hypothesis expresses that there are no differences among the three methods. By rejecting the null-hypothesis with a confidence level of  $(1 - \gamma)\%$  we can conclude that there is a difference in computing time among the three methods and the rank of each method is predictable.

Table 5.6 : The result of Friedman Test in-terms of computational time

<i>Test name</i>	<i>Friedman Rank Test (by SPSS software)</i>
<i>Sample size (N)</i>	<i>20 (the number of test problems)</i>
<i>Variable (K)</i>	<i>3 (solution time obtained by BLA, CBM and SA)</i>
<i>Statistic name</i>	<i>Chi – Square</i>
<i>Degree free (K-1)</i>	<i>2</i>
<i>Statistic value</i>	<i>15.70</i>
<i>P-Value</i>	<i>0.00 (so, <math>\gamma</math> is near 0)</i>
<i>Confidence level</i>	<i>99.99 (near 100%)</i>
<i>Rank Mean</i>	<i>SA-BLA=1.40 , BLA=1.95 , cbm-LAD=2.65</i>

Based on the information in Table 5.6 the null hypothesis is rejected and the high statistic value for the Friedman test shows, with a confidence close to 100%, the three presented methods are different in terms of computational time. According to the Rank Mean that is a part of the SPSS outputs, the SA-BLA method is placed in first priority, while the BLA and cbm-LAD are respectively in second and third place. The Friedman statistical test does not pinpoint inherently which method is different from each other. To do so a post-hoc test which is Wilcoxon signed-rank on different combinations of methods is applied. To use Wilcoxon signed-rank test a Bonferroni adjustment has to apply. To do so the significance level which is considered in Friedman test is divided by the number of tests. The number of tests is supposed to be done based on the number of combinations of methods to apply Wilcoxon signed – rank. So, there is a new significance level of  $0.01/3 = 0.003$ . This means that if  $P$ -Value is greater than 0.003 there is no statistically significant result (“Friedman Test in SPSS Statistics”, 2016). The Table 5.7 demonstrates the results of Wilcoxon signed-rank test for each pairs of methods. The results show that there is no significant difference between SA-BLA and BLA. Also, there is no significant difference

between BLA and cbm-LAD. But there is a statistically significant difference between SA-BLA and cbm-LAD and SA-BLA has lower computational time than cbm-LAD.

Table 5.7. Wilcoxon signed-rank results for each pairs of methods

	SA-BLA / BLA	SA-BLA /cbm-LAD	BLA / cbm-LAD
<i>P-Value</i>	0.064	0.000	0.025

### 5.3.2 Ranking in terms of accuracy

Similarly, in this step the null-hypothesis states that there is no difference in accuracy among the three methods. By rejecting the null-hypothesis with a confidence level of  $(1 - \gamma)\%$  we can conclude that there is a difference in accuracy among the three methods and that the rank of each method is predictable.

Table 5.8 : The result of Friedman test in-terms of accuracy

<i>Test name</i>	<i>Friedman Rank Test (by SPSS software)</i>
<i>Sample size (N)</i>	20 (the number of test problems)
<i>Variable (K)</i>	3 (solution time obtained by BLA, CBM and SA)
<i>Statistic name</i>	Chi – Square
<i>Degree free (K-1)</i>	2
<i>Statistic value</i>	2.5
<i>P-Value</i>	0.28 (so, $\gamma$ is near 0.29)
<i>Confidence level</i>	0.71
<i>Rank Mean</i>	SA-BLA=2 , BLA=2.25 , CBM=1.75

Based on the Table 5.7, the P-value is high. So, the null hypothesis is not rejected and there is no difference among three methods in-terms of accuracy. But a level of confidence close to 71% shows that the ranking done with the Friedman test is acceptable. Based on the *Rank Mean* result, SA-BLA is placed in the second level, after cbm-LAD and before BLA.

## 5.4 Conclusions

In brief, based on the Friedman statistical test there is difference between three methods; SA-BLA, cbm-LAD and BLA in terms of computational time. Moreover, according to Wilcoxon post-hoc signed rank test, there is a significant difference between SA-BLA method and cbm-LAD and SA-BLA method has lower computational time than cbm-LAD. Also, with a high confidence level, there is no difference in terms of accuracy among the various methods. Therefore, the SA-BLA method is introduced as an efficient algorithm. Because it can produce decision models that perform competitively compared to other methods, in a lower time.

## CHAPTER 6 CONCLUSION

In this chapter, a summary of the significance of developing a pattern generation algorithm based on meta-heuristic approaches is expressed, followed with a corresponding conclusion. Moreover, further improvements for the algorithm and associated models and possible future research areas are suggested.

### 6.1 Summary

Logical Analysis of Data, the logic based methodology, was considered as a selective method because of its interpretability and transparency. This method is not only competitive with other data analysis methods in machine learning but it also performs better than the other learning methods in some areas, such as CBM (condition-based maintenance).

Patterns, the main body of the LAD decision model, are the most influential factors of the performance and efficiency of this kind of learning model. Maximum patterns are a specific type of patterns that lead to highly accurate LAD classification models and using this kind of patterns are recommended and required in some practicable problems.

In spite of exact models and heuristic algorithms that were presented to generate maximum patterns, there is no meta-heuristic algorithm allowing to create (build) this type of patterns in order to produce efficient LAD models with low computational elapsed time. Also, the need of improving cbm-LAD, the software available in the Laboratory, in both computing time and accuracy for better compatibility with a variety of applications, motivated us to apply a meta-heuristic to generate maximum patterns.

In this study, the BLA (Best Linear Approximation) was considered as the starting point of the work. This model has been made heuristically by replacing the objective function of a nonlinear integer programming model, is presented to generate maximum patterns, with its best linear approximation in  $L_2$ . This model works well for small datasets, but it relies on a NP-complete problem and by increasing the size of the problem, which is the number of attributes in the dataset, it quickly becomes expensive to solve.

The Simulated Annealing (SA) meta-heuristic algorithm was applied here to generate maximum patterns based on the BLA. Implementation of the BLA model revealed some weaknesses of this model. Populated Simulated Annealing, a new approach of SA, was applied to meet all corresponding needs with the objective of modifying the BLA model. The performance and efficiency of the developed algorithm which was called SA-BLA was evaluated by applying a number of known datasets of the UC Irvine repository.

To increase the validity of the evaluation, the performance of the SA-BLA was examined by a number of generated random datasets. Later, the Friedman statistical test, which is followed by the Wilcoxon signed rank as a post-hoc statistical test, was applied to have a scientific evaluation for the presented work.

## **6.2 Synthesis of the study**

Significant results were obtained for 5 known datasets. For almost all of the datasets that are sufficiently large, the SA-BLA has the lowest computing elapsed time, in comparison with two other available methods; cbm-LAD, software available in the laboratory, and the BLA model. Also, the accuracy of the SA-BLA for most of the datasets was better than the accuracy of cbm-LAD, the BLA model coming in third place.

The reason for the lower accuracy of SA-BLA compared to cbm-LAD for two of the datasets is interpretable considering the type of patterns generated by cbm-LAD. Cbm-LAD consists of various kinds of patterns, which generate heuristically, including also strong patterns in its patterns pool. There is no limitation or necessity for cbm-LAD to generate patterns which are compatible with a specific observation referred to as  $\alpha$  to generate maximum  $\alpha$  – pattern. These factors lead to having higher accuracy for cbm-LAD than SA-BLA in some of the datasets.

In addition to the 5 known datasets, the Friedman statistical test was done on another 15 datasets (generated randomly by Matlab) in order to enhance the validity of evaluation. The statistical test results show there is a significance difference among three methods in terms of computational time. By applying Wilcoxon signed rank test it was shown that this difference is originated from the significance difference between SA-BLA and cbm-LAD and proved that SA-BLA has lower

computational time than cbm-LAD. Furthermore, when cbm-LAD was placed in the first rank in terms of accuracy, from a statistical point of view, SA-BLA was considered as equally accurate with a high level of confidence.

### **6.3 Future enhancements**

This work can be extended in multiple directions;

- a. Modification of the BLA model and improving it to a better one that can be a linear model and explicitly generate patterns with the maximum coverage of observations.
- b. Combining the SA-BLA model with available heuristics as MPP and MSP that are designed to generate a combination of maximum patterns and other kinds of patterns, which are prime and strong patterns respectively and attempt to have a patterns pool formed by various kinds of patterns. Finally compare the results with cbm-LAD.
- c. Combining the SA-BLA algorithm with another algorithm in the Data mining domain. This new algorithm is made to search for the most fruitful observations which considered as parameter  $\alpha$ , to generate maximum  $\alpha$ -patterns.

## BIBLIOGRAPHY

- Almuallim, H., & Dietterich, T. G. (1994). Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence* 69(1–2), 279–305.
- Alexe, G., Alexe, S., Hammer, P. L., & Vizvari, B. (2006.2). Pattern-based feature selection in genomics and proteomics. *Annals of Operations Research* 148(1), 189–201.
- Alexe, G., & Hammer, P. L. (2006.3). Spanned patterns for the logical analysis of data. *Discrete Applied Mathematics* 154(7), 1039–1049.
- Alexe, G., Alexe, S., Hammer, P. L., & Kogan, A. (2008). Comprehensive vs. comprehensible classifiers in logical analysis of data. *Discrete Applied Mathematics* 156(6), pp. 870–882.
- Bennane, A., & Yacout, S. (2012). LAD-CBM; new data processing tool for diagnosis and prognosis in condition-based maintenance. *Journal of Intelligent Manufacturing*, 23(2), 265–275.
- Bertsimas, D., & Tsitsiklis, B., (1993). Simulated Annealing. *Statistical Science, Report from the Committee on Applied and Theoretical Statistics of the National Research Council on Probability and Algorithms*, 8(1), 10–15
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.
- Blake, C.L., & Merz, C.J., (1998). UCI Repository of Machine Learning Databases, Department of Information and Computer Sciences, University of California, Irvine.
- Boros, E., Hammer, P. L., Ibaraki, T., & Kogan, A. (1997). Logical analysis of numerical data. *Mathematical Programming*, 79(1–3), 163–190.
- 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*, 12(2), 292–306
- Bonates, T. O., Hammer, P. L., & Kogan, A. (2008). Maximum patterns in datasets. *Discrete Applied Mathematics* 156, 846 – 861
- Blum, C. Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Computing Surveys (CSUR)*, 35(3), 268–308
- Boros, E., Horiyama, T., Ibaraki T., & Makino. K., Finding essential attributes from binary data. *Annals of Mathematics and Artificial Intelligence* 39(3), pp. 223–257. 2003.

- Bruni, R. (2007). Reformulation of the support set selection problem in the logical analysis of data. *Annals of Operations Research* 150(1), 79–92
- Cenney, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*. 45: 41–51.
- Chen, S., Xudiera, C., & Montgomery, J. (2012). Simulated Annealing with threshold convergence, WCCI 2012 IEEE World Congress on Computational Intelligence, June, 10-15, 2012, Brisbane, Australia
- Chvatal, V. (1979). A greedy heuristic for the set covering problem. *Mathematics of Operations Research* 4(3), 233–235.
- Conover, W. (1999). *Practical nonparametric Statistics*. New York: Wiley.
- Crama, Y., Hammer, P. L., & Ibaraki, T. (1988). Cause–effect relationships and partially defined boolean functions. *Annals of Operations Research* 16(1), 299–326.
- Dorigo, M. Stutzle, T. (2004). *Ant Colony Optimization*. Massachusetts Institute of Technology, MIT press, Massachusetts, United States of America.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200), 675–701
- Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1), 86–92 .
- Glover, F., & Laguna, M. (1997). *Tabu Search*, Springer New York, doi: 10.1007/978-1-4419-7997-1\_17
- Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural network design*. Boston : PWS Publishing
- Hammer, P. L., Kogan, A., & Lejeune, M. A. (2012). A logical analysis of banks' financial strength ratings. *Expert Systems with Applications*, 39(9), 7808–7821.
- Hammer, P. L., Liu, Y., Simeone, B., & Szedmák, S. (2004.1). Saturated systems of homogeneous boxes and the logical analysis of numerical data. *Discrete Applied Mathematics* 144(1–2), 103–109.
- Hammer, P. L., Kogan, A., Simeone, B., & Szedmák, S. (2004.2). Pareto–optimal patterns in logical analysis of data. *Discrete Applied Mathematics*, 144(1), 79–102.
- Hammer, P. L., & Bonates, T.O. (2006). Logical analysis of data – an overview: from combinatorial optimization to medical applications," *Annals of Operations Research*, 148, 203–225.

Hammer, P. L.(1986). Partially defined Boolean functions and cause–effect relationships. International conference on multi–attribute decision making via or–based expert systems.

Kim, H. H., & Choi, J. Y. (2014). Pattern generation for multi–class LAD using iterative genetic algorithm with flexible chromosomes and multiple populations. *Expert systems with applications*, 42(2), 833–843.

Kotsiantis, S., & Kanellopoulos, D. (2006). Discretization techniques: a recent survey, *GESTS International Transaction Computer Science Engineering*, vol. 32, 47–58.

Liu, H., Hussain, F., Tan, C. L., & Dash, M. (2002). Discretization: An enabling technique. *Data Mining and Knowledge Discovery* 6(4), 393–423.

Mayoraz, E., & Moreira, M. (1999 ) Combinatorial approach for data binarization. *Principles of Data Mining and Knowledge Discovery* 1704, 442–447.

Kirkpatrick, S., Gelatt, Jr., C. D. & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*. 220 (4598), 671–680.

Mortada, M. A., Yacout, S., & Lakis, A.( 2011). Diagnosis of rotor bearings using logical analysis of data, *Journal of Quality in Maintenance Engineering*, vol. 17, 371–397.

Mortada, M.–A., Yacout, S., & Lakis, A. (2014). Fault diagnosis in power transformers using multi–class logical analysis of data. *Journal of Intelligent Manufacturing*, 25(6), 1429–1439.

Ragab, A., Yacout,S., & Ouali, M.S. (2015). Interpretable pattern-based machine learning for Condition Based Maintenance, 61st Annual Reliability and Maintainability Symposium (RAMS 2015), 6 pages.

Ryoo. H. S., 2009. MILP approach to pattern generation in logical analysis of data. *Discrete Applied Mathematics* 157(4), 749 –761.

Guo.C., and Ryoo. H. S., (2012). Compact MILP models for optimal and Pareto–optimal LAD patterns. *Discrete Applied Mathematics*, 10(16–17), 2339–2338.

Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3), 660 – 674.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Shaban, Y., Yacout, S., & Blazinski, M. (2015). Diagnosis of machine tools using Knowledge Extraction and data analysis, *IEEE conference*

Yacout, S., Salamanka, D., & Mortada, M. (2012). Tool and method for fault detection of devices by Condition Based Maintenance. No. Wo 2012/00984. Montreal, QC, Ecole Polytechnique de Montreal.

Yacout, S., (2010). Fault detection and diagnosis for condition based maintenance using the logical analysis of data. In: Computers and Industrial Engineering (CIE), 2010 40th International Conference on. IEEE, pp. 1–6. doi:10.1109/ICCIE.2010.5668357

Yacout, S.(2012). Logical analysis of maintenance and performance data of physical assets. in Proc. Annu. Rel. Maint. Symp. (RAMS), Reno, NV, USA, Jan. 2012, pp. 1–6.

Yang, X.Sh.(2010). Engineering Optimization. New York :*Wiley*