Alma Mater Studiorum - Università di Bologna

Scuola di Dottorato in Scienze Economiche e Statistiche
Dottorato di ricerca in

Metodologia Statistica per la Ricerca Scientifica
XXIII ciclo

An Evolutionary Approach to the Design of Experiments for Combinatorial Optimization with an Application to Enzyme Engineering

Matteo Borrotti

Dipartimento di Scienze Statistiche "Paolo Fortunati"
Marzo 2011

Alma Mater Studiorum - Università di Bologna

Scuola di Dottorato in Scienze Economiche e Statistiche
Dottorato di ricerca in

Metodologia Statistica per la Ricerca Scientifica
XXIII ciclo

An Evolutionary Approach to the Design of Experiments for Combinatorial Optimization with an Application to Enzyme Engineering

Matteo Borrotti

Coordinator: Prof. Daniela Cocchi
Tutor: Prof. Alessandra Giovagnoli

Settore Disciplinare: SECS-S/01

Dipartimento di Scienze Statistiche "Paolo Fortunati"
Marzo 2011

Abstract

In a large number of problems the high dimensionality of the search space, the vast number of variables and the economical constrains limit the ability of classical techniques to reach the optimum of a function, known or unknown. In this thesis we investigate the possibility to combine approaches from advanced statistics and optimization algorithms in such a way to better explore the combinatorial search space and to increase the performance of the approaches. To this purpose we propose two methods: (i) Model Based Ant Colony Design and (ii) Naïve Bayes Ant Colony Optimization. We test the performance of the two proposed solutions on a simulation study and we apply the novel techniques on an appplication in the field of Enzyme Engineering and Design.

Contents

Introduction

In the last decade, biology has witnessed a radical paradigm shift from a descriptive and analytical science to an engineering science which has led to the emergence of the new scientific area of synthetic biology (*SB*). In this development, biological systems have shown to be characterized by a highly hierarchical organization involving a large number of interwoven parameters that need to be all simultaneously optimized in order to achieve the desired objective. A good example is the so called Enzyme Engineering Design. The interest in enzyme (re)design lies in the fact that enzymes are capable of promoting (*i.e.* catalyze) chemical reactions and are responsible for the vast majority of chemical reactions occurring within a cell. Within this framework, several groups worldwide have developed a number of techniques to redesign enzymes and alter their functionality and specificity in order to exploit them for useful purposes. However, current techniques are not capable of redesigning enzyme functionality effectively.

The main problem of enzyme engineering and design is the very large experimental space defined by an enormous number of variables and their possible levels. In addition, variables usually interact, with long range and epistatic effects. However, the number of experimental points that can be tested in a lab is severely limited by economic and time constraints. Statistical science plays a pivotal role to engineer biological systems when a comprehensive knowledge is missing or the simultaneous optimization of a large number of variables is required. However, in many optimization problems the large domain and the huge experimental space limit the ability of classical approaches to reach the desired objective (*i.e.* the optimum of a given fitness function). Thus, there is the urgent need to develop novel procedures for the smart exploration of the sequence space in order to identify the best enzyme variants with a reasonable effort (Chapter 1).

The motivation of this thesis is an ongoing case study based on a library of 95 different amino-acid sequences of 50 amino-acids, called pseudo-domains, which yield a full-length protein (or enzyme) of 200 amino-acids generated by assembling 4 pseudo-domains at a time (Chapter 1). The number of all the theoretically different full-length random enzymes to be screened is represent by all the permutations (with repetition) of 95 elements in 4 positions, which is equal to $95^4 = 8.1 \times 10^7$. Thus the search space composed by all the candidate enzymes is enormous. The aim of this study is to develop a method for finding enzymes with a biological functionality. In fact, the methodology might help to improve existing enzymes or to design novel ones for specific applications in different fields such as medicine and fine-chemical production. The problem we are facing can be considered as a discrete optimization problem and the main aim is to find the optimum of a function, or a good approximation of it, in a very large discrete search space where we do not have *a priori* knowledge about its principal features.

The first step of this thesis is to investigate the performance of some classical approaches to tackle our biological problem and its complexity. After transforming our discrete problem into a continuous one (Chapter 2) by recent developments of the well known statistic technique called Multidimensional scaling, we develop a 3-stage approach that deals with continuous variables and uses some concept from Response Surface Methodology. However, this method fails in the approximation of the response surface and it does not have good performance in terms of prediction. This failure is possibly due to the very nature of the classical methods devised for cases when a large number of experimental units were measured and a small number of features had to be considered, whereas we are in the presence of a so called "*large p, small n*" problem [39]. We give an overview of some the most important techniques for "*large p, small n*" problems in Chapter 3.

As a consequence of this failed attempt we have been forced to face the discrete nature of the optimization problem itself. Optimization problems that involve discrete variables are called *Combinatorial Optimization* (*CO*) problems (Chapter 4). In this thesis, we have investigated some so-called metaheuristic algorithms that serve the purpose, with the ultimate aim to test the possibility of exploiting bio-inspired algorithms combined with advanced statistical techniques to search in a discrete sequence space for a target structure. More specifically, in Chapter 4 we describe the Ant Colony Optimization (ACO) [21] [25] [22] approach. Ant algorithms are inspired by the observation of real ant colonies. In ACO algorithms, a colony of artificial ants (agents) cooperate in finding solutions to a difficult discrete minimization problem. A solution is expressed as minimum cost (shortest) path through the states of the problem in accordance with the problems constraints. Starting from an initial state selected according to some problem dependent criteria, each artificial ant builds a path. A single ant is able to build a path, but only the cooperation among all the agents of the colony concurrently building different paths is able to find high quality solutions. ACO is based on probabilistic matrices where the best path has higher probability to be chosen.

This thesis endeavours to define a new approach within Design of Experiments for optimization based on Evolutionary Model Based Experimental Design [29] [10] [5], that exploits a combination of approaches from Design of Experiments and metaheuristic algorithms to guide the exploration of the space. We propose two different approaches (Chapter 5):

– Model Based Ant Colony Design ($MACD$);

– Naïve Bayes Ant Colony Optimization ($NACO$).

which we now briefly describe.

1. MACD is based on the idea behind closed loop optimization [42] and the procedure boosts an optimization algorithm by a statistical model. More precisely, MACD combines:

   – $\mathcal{MAX} - \mathcal{MIN}$ Ant System [68]. $\mathcal{MM}$AS is coupled with a local search (Simulated Annealing [41] [13]) to increase the performance of the algorithm;

– linear regression model with binary predictive variables, which is estimated from observed data by the least squares method.

MACD couples real experimentation with simulated experiments. Solutions are generated by an algorithm in computer simulation, but their evaluation is achieved by a physical experiment. Evaluations are fed back to the simulative phase of the approach and its generation of subsequent solutions is a function of these. This enables the method to explore the complex relationships between input and output variables. The main advantage of this is that the system becomes more observable, since computer runs are generally easier and cheaper than measurements taken in a physical set-up, and the exploration can be carried out more thoroughly. Our method improves upon existing methods in two main directions:

(i) Thanks to the statistical model, we can simulate the problem and move in the search space as many times as we want, improving the solutions step by step;

(ii) The iterated refinement of the predictive model provides the optimization algorithm with predictive capability of the model resulting in an increased accuracy during the optimization process.

2. In order to learn the most about the system under study using the least number of trials, we tackle the problem using a Naïve Bayes Classifier [56] combined with Ant Colony Optimization. We have called this approach Naïve Bayes Ant Colony Optimization (NACO). NACO extracts the information from the data using the Naïve Bayes Approach and explores the search space by the ACO algorithm. Considering a sequence of elements (*i.e.* a string of letters), Naïve Bayes Approach identifies which elements affect mostly the response of the system for each position and the optimization algorithm selects the best path, *i.e.* the connection between the elements in the sequence in the set of possible candidate sequences. In this context, NACO improves upon the limits of the Naïve Bayes Classifier. In Naïve Bayes Classifier it is

impossible to understand the relations between the elements. The combination of ACO and Naïve Bayes Approach can avoid this problem. A path is composed by nodes and arcs. Considering the previous string of letters, nodes can be seen as letters and an arc connecting a letter to the next one can be seen as the relation that exists between the two letters in a sequence. In other words, ACO can capture the interaction existing between letters in a sequence. Moreover, an advantage of NACO is that it is not computationally intensive. This advantage allows the researcher to be fast in the creation and analysis of possible solutions.

Before starting the physical experimentation, we have tested the performance of our proposed methods on 3 different benchmark functions (Chapter 5). MACD and NACO have shown good results in terms of convergence, in fact the two approaches outperform standard algorithms in the search of the optimum. MACD has shown good results in 2 of the benchmark functions and satisfactory result in the third one. NACO has shown satisfactory results in all the benchmark functions and is comparable with other algorithms. The analysis of the performance of the Model Based Ant Colony Design and the Naïve Bayes Ant Colony Optimization have demonstrated that our approaches are able to reach good points in few iterations for the three benchmark functions (Chapter 5).

After the simulation study, we have applied MACD and NACO to the problem of Enzyme Engineering Design (Chapter 6). Starting from a set of 96 randomly chosen enzymes, we have performed 5 generations for each approach in a sequential way. The purpose is to study the evolution of the generations and the performance of the two techniques in a real application. MACD has demonstrated to move toward good regions of the search space, generation by generation the distribution of the response was moving to higher values. NACO has reached high values of the Score in few generations and has confirmed the ability to explore the search space in a exhaustive way. The two approaches have required a very small number of experimental points to reach the optimality region of the search space and have demonstrated the ability to tackle problems where a small number of experimental points are measured and a large number of parameters has to be considered.

From our simulative and experimental studies on the performance of MACD and NACO we have demonstrated that the two approaches are able to:

– learn about a complex system composed by several experimental input variables with patterns of interactions;

– reach an approximation of the optimum of the unknown functions with few iterations;

– explore a very large part of the search space. In the case of MACD , this happen because of being able to predict the response over a large part of the search space. In the case of NACO, because of being able to extract information on that would not be used otherwise;

– reduce the number of real experimentations in order to save money and time.

Moreover we were successful in creating an interactive process where the dialogue between design and laboratory experimentation at each generation has created a path in the combinatorial search space that leads toward a region of optimality.

Chapter 1

The Motivating Problem: Designing Enzyme Functionality

1.1 Enzyme Engineering Design

At the origin, the purpose that started this thesis was to find a methodological approach to tackle the problem of (re)designing enzyme functionality using secondary structure domains.

The interest in enzymes (re)design lies in the fact that enzymes are capable of promoting (*i.e.* catalyzing) chemical reactions and are responsible for the vast majority of chemical reactions occurring within a cell. One of the striking feature of enzymes is their specificity and effectiveness, indeed a particular enzyme is capable of recognizing a specific substrate among many closely related ones and is capable of promoting chemical reactions under mild conditions (pH 7, room temperature, 1 atmosphere). To this regard, enzymes are considered the most appealing catalysts to perform chemical reactions for human purposes such as synthesis of high-value compounds (*e.g.* pharmaceuticals) or biodegradable polymers (*e.g.* polylactic acid or polyhydroxyalkanoate).

Within this framework, several groups worldwide have developed a number of techniques to redesign enzymes and alter their functionality and specificity in order to exploit them for useful purposes. However, current techniques are not capable of redesigning enzyme functionality effectively. Thus, there is the urgent need to develop novel procedures for the smart exploration of the sequence space in order to identify the best enzyme variants with a reasonable effort.

1.2 Enzyme function and structure

Enzymes are a specific class of proteins, whose main function is to promote chemical reactions. Enzymes are capable to catalyze a broad range of reactions ranging from organic molecule synthesis such as antibiotic [28] to degradation of pollutants [59].

From the structural prospective, enzymes are linear polymers built from series of up to 20 different L-α-amino-acids arranged in a linear chain (primary structure), which folds into a number of transient states (secondary structure), to yield a well-defined three-dimensional structure (tertiary structure) (Figure 1.1). The three-dimensional shape of a given enzyme ultimately defines the enzymes function so that redesigning enzyme activity is ultimately linked to redesigning enzyme shape. Operatively, enzyme tertiary structure is determined by the secondary structure which is in turn determined by the linear arrangement of amino acids. Thus, redesigning enzyme function can be accomplished by redesigning either the enzyme primary or the secondary structure.

Summarysing, an enzyme P_i is a sequence of monomers from a set $A \equiv \{a_1, a_2, a_3, \ldots, a_{20}\}$ joint together to form a complex string (*i.e.* primary structure), which may differ in length, amino-acid composition and sequence. Each string is associated to a secondary and tertiary structure, which defines enzyme activity.

The main problem of enzyme engineering and design is the large experimental space defined by an enormous number of variables (*e.g.* amino-acid composition and order, 3D interactions among secondary domains, and other things) and their possible levels. In addition, variables usually interact in a

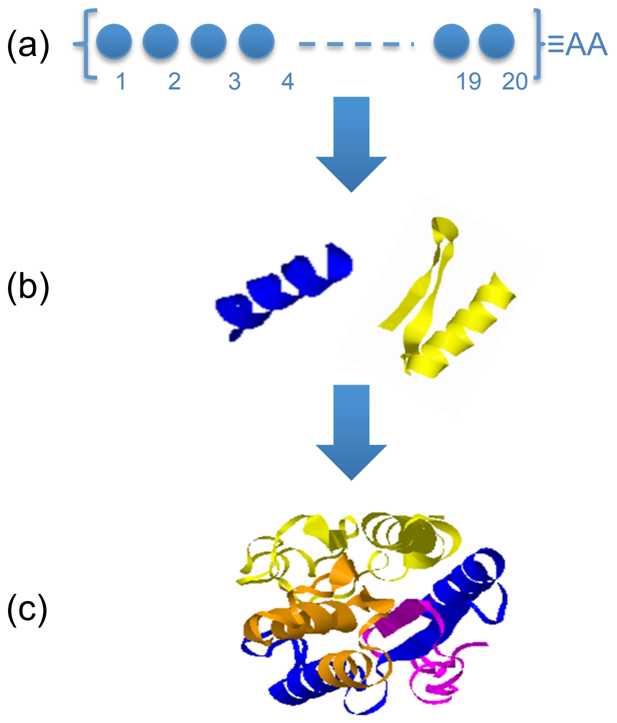


Figure 1.1: Structural prospective of a enzymes: (a) Primary Structure (b) Secondary Structure (c) Tertiary Structure.

non-linear way; with long range and epistatic effects. Furthermore, the number of experimental points that can be tested in a lab is severely limited by economic and time constraints.

1.3 Protein engineering and design: state-of-the-art

The redesign of enzyme specificity and performance has long tempted biochemists due to the enormous potential of enzymes as fine catalysts operating under mild conditions. Enzyme Engineering and Design can be seen as a walk through a multi-dimensional experimental space to find mutants with improved or novel properties. There are 3 general strategies for enzyme engineering:

– rational design (*RD*) [66];

– directed evolution (*DE*) [27] also known as irrational design or applied molecular evolution:

– high-throughput screening (*HTS*) [62].

Rational design (Figure 1.2(a)) relies on a detailed knowledge of the relationship between structure and function so that tailored, site-specific mutations can be performed to rationally alter the function of the target protein. Despite the recent success reported in literature [62], a comprehensive description of the structure-function relationship is rarely available, which severely limits the deployment of rational design strategies.

Conversely, directed evolution (Figure 1.2(b)) mimics natural evolution by means of iterative cycles of mutation-selection-amplification, which allows the simultaneous testing of a vast library of candidate mutants for a priori defined function without a priori knowledge. The great advantage of directed evolution techniques is that no prior structural knowledge of a enzyme is required, nor is it necessary to be able to predict what effect a given mutation will have on target enzymes function. There are two fundamental

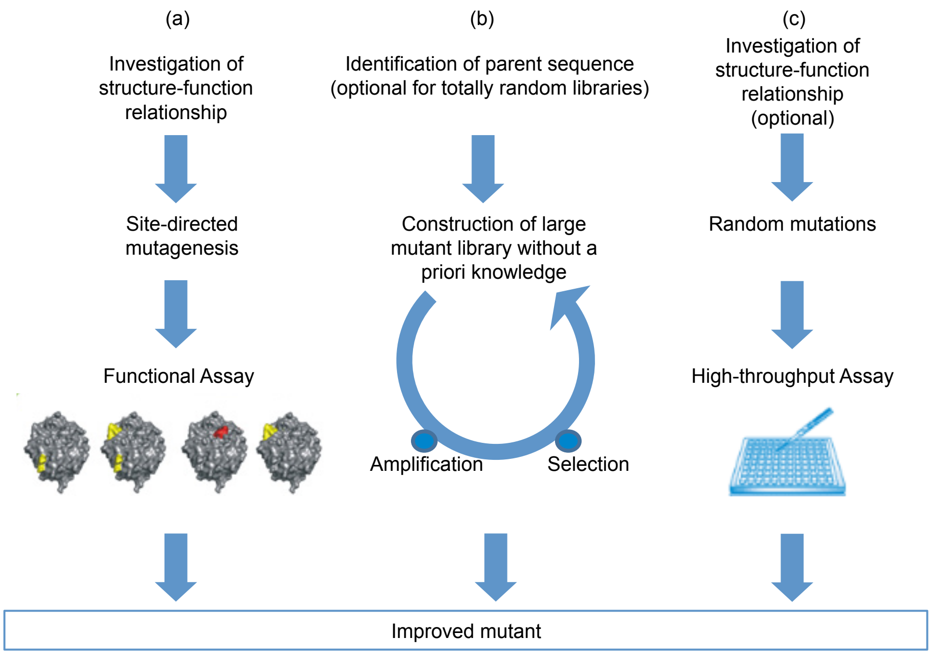

(a)
Investigation of structure-function relationship
Site-directed mutagenesis
Functional Assay
(b)
Identification of parent sequence (optional for totally random libraries)
Construction of large mutant library without a priori knowledge
Amplification
Selection
(c)
Investigation of structure-function relationship (optional)
Random mutations
High-throughput Assay
Improved mutant

Figure 1.2: Schematic representation of the main approaches to molecular design and engineering: (a) Rational Design (b) Direct Evolutions (c) High Throughput Screening.

requirements to carry out directed evolution: the first is the availability of physical link between the genotype and the phenotype. The second relies on the availability of a suitable screening procedure to enrich the initial enzyme population of those sequences satisfying the selection criteria.

Despite the numerous accomplishment [67], DE cannot be effectively exploited to re-design or engineer enzymes (*e.g.* enzymes embedded with a catalytic function) due to the lack of suitable selection methods.

The high throughput screening (Figure 1.2(c)) approach relies on the generation of a vast library of candidate mutants, that are individually tested for the desired function. Despite the recent success [69], screening procedures can explore only an infinitesimal number of mutants with respect to all the theoretical ones. Indeed, the number of theoretically possible mutants is

$$M_H = 19^H \frac{L!}{(L-H)!(H)!},$$

where M_H is the theoretical number of mutants for a given protein of length L assuming H mutations. It is easy to calculate that for an enzyme of only 200 amino acids assuming 3 mutations per mutant there are more than 10^{10} theoretically possible mutants.

Due to the limitations described above, there is the urgent need to develop novel procedures for the smart exploration of the sequence space in order to identify best mutants with reasonable effort. To tackle this problem, we propose to combine approaches from Design of Experiments and metaheuristic algorithms to guide the smart exploration of the sequence space.

The methodological approach adopted in this work relies on the (re)design of enzymes functionality using short amino-acid sequences rather than operating on individual amino acid at primary structure level.

Accordingly, we designed a library of 95 different amino-acid sequences of 50 amino-acids called pseudo-domains, that are subsequently assembled to yield a full-length protein of 200 amino-acids (*i.e.* candidate solution) generated by assembling 4 pseudo-domains at a time, according to the algorithms described in Chapter 5.

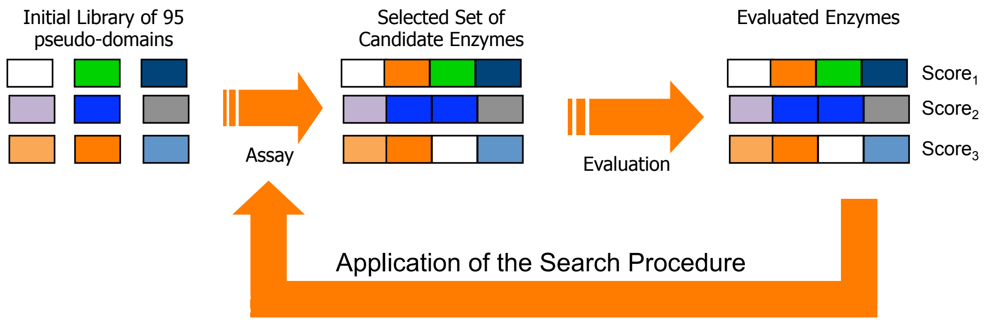

Initial Library of 95 pseudo-domains
Selected Set of Candidate Enzymes
Evaluated Enzymes
Assay
Evaluation
$Score_1$
$Score_2$
$Score_3$
Application of the Search Procedure

Figure 1.3: General framework of the procedure.

The possible number of theoretically different full-length random enzymes to be screened is represent by all the permutations (with repetition) of 95 elements in 4 positions are $95^4 = 8.1 \times 10^7$.

1.4 Designing, evaluating and scoring random pseudo-domains and candidate enzymes

The library of 95 different pseudo-domains were generated according to the methods described by [55], which yield 95 completely random pseudo-domains with no significant homology with extant enzymes.

Pseudo-domains similarity matrix calculation

Despite the difference in primary sequence of pseudo-domains, it is possible to elaborate a similarity metric which defines a 95×95 matrix describing the relative similarity between any two pseudo-domains. The first step is to calculate pseudo-domains secondary structure using PSIPRED software [52]. PSIPRED predicts whether a given pseudo-domains adopts an helix, coiled-coil or beta-sheet conformation. The typical output of the PSIPRED algorithm is presented in Figure 1.4.

The output reported in Figure 1.4 states that the enzyme under investigation adopts a coiled-coil conformation (C) at positions 1, 19 and 20; a beta-sheet conformation (E) at positions 2-5 and 14-18 and an helix confor-

```
Conf: Confidence (0=low, 9=high)
Pred: Predicted secondary structure (H=helix, E=strand, C=coil)
AA: Target sequence

# PSIPRED HFORMAT (PSIPRED V3.0)
```

Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
AA:	Y	H	C	T	Y	S	Q	S	E	P	G	G	G	K	T	Q	T	Y	S	C
Pred:	C	E	E	E	E	H	H	H	H	H	H	H	H	E	E	E	E	E	C	C
Conf:	9	1	3	3	2	0	1	3	8	9	9	9	7	1	1	6	8	8	8	2

Figure 1.4: PSIPRED output of a test protein.

mation at positions 6-12.

The second step is to use secondary structure profile of pseudo-domains calculated using PSIPRED to calculate the similarity among different pseudo-domains according to the method proposed by [19]. Briefly, the similarity of pseudo-domain was calculated using secondary structure prediction by aligning all pseudo-domains in a pair-wise fashion and calculating similarity score by a two-step procedure as follows:

1 - *Position-related score calculation:*

$s_{i,j} = \textbf{IF } (pred_{i,j} = pred_{i,r})\ confi_{i,j}\ \textbf{ELSE } \{0\}$

where $s_{i,,j}$ is the position-score of the *i-th* amino acid of the *j-th* pseudo-domain, $pred_{i,j}$ is the secondary prediction of the *i-th* amino-acid of the *j-th* pseudo-domain whereas $(pred_{i,r})$ is the secondary prediction of the *i-th* amino-acid at the same position in the *r-th* pseudo-domain with $r \neq j$. When the IF statement is satisfied, the output value is the correspondent confidence value $confi_{i,j}$ of the *i-th* amino acid of the *j-th* pseudo-domain, otherwise the output value is zero.

2 - *Global score calculation:*

$S_j = \sum_{i=1}^{50} s_{i,j}$

which is a summation of individual position-scores over the entire pseudo-domain length.

It is noteworthy that the pair-wise alignment generates a non-symmetric matrix. This is due to the fact that the IF statement output value is equal to the confidence value of the query sequence. As an example consider two pseudo-domains j and r which adopt the same predicted conformation at position $i = 34$ with confidence 6 and 9 respectively. When comparing the 34-*th* amino-acid of pseudo-domains r versus j the IF statement output value is 6, whereas it yields 9. When comparing the 34-*th* amino acid of pseudo-domains j versus r. Although equivalent, the two-halves of the similarity matrix are not the same. In according with the biologists, we chose the upper half as similarity matrix to be used in sequel of the thesis.

Candidate enzymes scoring function

The rationale beyond the method proposed relies in redesigning enzymes functionality starting from secondary domains to circumvent the limitations described in Paragraph 1.3. Accordingly, the scoring function adopted to rank candidate solutions relies on the evaluation of a distance metric between any given candidate solution and a desired target structure not included in the attainable search space derived from all the possible permutation of the 95 pseudo-domains in all positions (*ca.* 81mln candidate solutions). The scoring function employed was the one described by [18]. The theoretical range of the score, or response, is from 0 to 1000.

Combinatorial constraints to designing random enzymes

Candidate enzymes shall be experimentally characterized in terms of expression, solubility, structural features and enzymatic activity. In order to be experimentally tested, candidate enzymes shall respect the following biological restrictions:

i. The number of cysteine residues shall be no higher than 9 and different from 5 and 7;

ii. The percentage of coil shall not be higher than 70.

These constraints should be considered in the proposed approaches.

1.5 Relevance of the research

The present research is relevant both for statistics and the biological sciences.

From a methodological perspective, the ultimate aim of this work is to test the possibility of exploiting bio-inspired algorithms combined with advanced statistical techniques to search in a discrete sequence space for a target structure. We wish to define a new approach within Design of Experiments for optimization based on the Evolutionary Model Based Experimental Design that has been proposed in [29] [10] [5], which is able to solve problems where the number of variables vastly increases.

The issue of high dimensionality represents a challenging topic for statisticians with many problems that are still unsolved. The large number of variables and the scarcity of observations characterize the current state of real applications, therefore, new methods and theories need to be developed. The thesis endeavours to overcome classical methods in optimizing high dimensional systems, giving a new flexible tool for tackling complex scenarios.

With respect to the biological sciences, the relevance of this research is two-fold. First, this research addresses the problem of developing a novel method to effectively explore enzyme sequence space to improve or redesign enzyme functionality. The objective is to go beyond the state-of-the-art and circumvent the limitations of current enzyme engineering techniques that - despite the differences - rely on the random exploration of the sequence space. This approach is labour-intensive and time-consuming and represents a concrete bottle-neck to the development of improved enzymes.

Second, we start from a library of 95 non-natural completely random pseudo-domains with no significant homology to extant enzymes. The straightforward consequence is that the candidate enzymes produced starting from this dataset will be themselves novel and with no significant homology to extant enzymes. Thus, this approach allows the exploration of the sequence space that has not been sampled during the course of natural evolution [14] contributing to the age-old discussion in theoretical biology concerning contingency and determinism [51]; namely *are natural enzymes the optimal solu-*

tion found by natural evolution? or *rather natural enzymes represent simply a suitable solution, a sort of frozen accident*?

The debate between contingency and determinism is of paramount importance in theoretical biology, since it relates to the possibility that nature has explored only a tiny fraction of what is possible and attainable in the biological realm.

Chapter 2

An Approach to Optimization by Some Classical Statistical Methods

2.1 Introduction

In the previous chapter we present the nature of the problem. The aim is to find an optimum in a discrete search space. The method that we want to use has to be able to move in a large search space.

In this chapter, we investigate the performance of classical statistical methods in the face of our problem. For this purpose, we want to change a discrete problem into a continuous one and we develop a 3-stage method:

– We start with a similarity (or dissimilarity) matrix;

– We apply a recent development of Multidimensional Scaling (MDS) [46] [9] so that the set of "objects" we start with can be represented by points in a low dimensional space;

– We fit a polynomial regression to our data.

In our problem each enzyme is a sequence of 4 positions. For each position we can select an element (pseudo-domain) from a set of 95 objects. The discrete nature of the problem is evident. Our goal is to transform the 95 objects (that can be selected in each position) in points of an n-dimensional Euclidean space (*e.g.* 2-dimensional space). Therefore, the sequence with 4 positions can be seen as a value from $\mathbb{R}^{4n}$, *i.e.* $4 \times n$ real numbers that specify the coordinates of a point in a $(n \times 4)$-dimensional space. For this aim, we apply a recent method from MDS, called Scaling by MAjorizing a COmplicated Function (SMACOF) [15], to a 95×95 matrix that describe the relative similarity between any two objects (Chapter 1). Notice that, in this context, MDS is applied to convert the problem from a discrete to a continuous one. Traditionally, MSD is a visualization tool for reducing high dimensional data in low dimensions with the aim of discovering meaningful information obscured by the intrinsic complexity of the data.

For this point onwards, we can use polynomial regression with the purpose of exploring the response surface by Response Surface Methodology (RSM) [58]. RSM explores the shape of the dependence relation of the response on a set of a quantitative factors and uncovers the particular combination of factors levels that yields the maximum or minimum response.

In the first part of this chapter we introduce some basic concepts of Design of Experiments (DoE) [3], that will be applied later, we present the SMACOF and we describe RSM.

2.2 Design of Experiments

In the last decade, researches have devoted a lot of effort to increase their ability to perform complex experiments. A great deal of experimentation is an efficient method of learning about the world. In fact, in developing a scientific theory, testing a research hypothesis or getting insights into the process underlying an observable phenomenon, several questions may be addressed by conducting experiments. In conducting experiments, most of the system elements are supposed to be under the control of the investigator, who can then strongly affect the accuracy of the experimental result. The investigator plans (designs) the experiments, by deciding on what has to be experimentally evaluated and how the experiments should be conducted. The validity of the interpretation of the experimental results strongly depends on the elements selected for the analysis and on the laboratory protocols chosen to conduct the experimentation. In several research areas, such as biology, chemistry, or material science, experimentation is complex, extremely expensive and time consuming, so an efficient plan of experimentation is essential to achieve good results and avoid unnecessary waste of resources. The statistical theory that deals with this problem is the so called Design of Experiments (DoE) [3]. An accurate statistical design of the experiments is important also to tackle the uncertainty in the experimental results derived from systematic and random errors that frequently obscure the effects under investigation.

In DoE, fundamental importance is attached to the model relating the responses observed in the experiments to the experimental factors. The purpose of the experiments is often to find out about the model, including its adequacy. Frequently the model is then used to optimize the output.

An experimental design can be described as a selected set of experimental points where different compositions and different laboratory conditions are tested. Formally, an experimental design can be written as

$$X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)'$$

where n is the number of selected experimental points and each $\mathbf{x}_i$ is a p-vector

$$\mathbf{x}_i = (x_{i1}, \ldots, x_{ip}) \text{ with } i = 1, \ldots, n$$

describing the particular combination of p factors that are tested in that particular trial may yield the experimental results

$$\mathbf{y} = (y_1, y_2, \ldots, y_n).$$

In a common and schematic way the system under study may be represented in Fig. 2.1. In the figure, the $\mathbf{x}_i$ vector describe the i-*th* experimental point that is a combination of the relevant p factors that independently or in relations among them (interactions) can affect the result of the experimentation. In this scheme the factors z_{ui}, $u = 1, \ldots, v$ represent variables that can

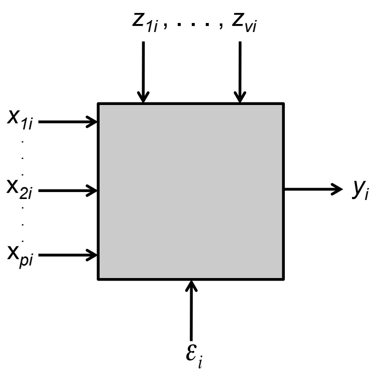


Figure 2.1: General representation of a system.

affect the results of the experimentation but are not under the control of the investigator, and the elements ϵ_i describe the experimental errors. Finally the result, or response of the i-*th* experimentation, is denoted by y_i.

Experiments can be designed to answer a variety of questions. In fact the investigator is asked to determine the number of experimental points (n) to test for achieving reliable results; how many factors should be considered (p) and how many levels (or the range of variation for continuous factors); and also which factor interactions should be investigated. In analysing and modelling the resulting data the investigator is further asked to infer which factor and factor interactions are the most influential on the responses; which combination can optimize the response value; which combination gives the smallest variability in the response; and finally, given the systematic and random errors in the experimentation, which level of uncertainty characterizes the estimation of relevant parameters and overall interpretation of the results.

In the case of discrete variable, the reference model is the so called Analysis of Variance (*ANOVA*). ANOVA is a collection of statistical models in which the observed variance in a particular variable is partitioned into components attributable to different sources of variation to the effect of testing equality of effects and existence of interactions.

When the main objective of experimentation is to optimize the response of the system, the *Response Surface Methodology* is commonly adopted.

2.2.1 Response Surface Methodology

The typical application of Response Surface Methodology (RSM) [58] is where several input variables potentially influence some performance measure or quality characteristic of a product or process. This performance measure or quality characteristic is called the response and is measured on a continuous scale.

Suppose for instance, that p variables taken at different levels $(x_1, x_2, \ldots, x_p)$ can give rise to a particular response. This response represents the output of the system and is measured by an identified variable Y. The dependence relation between the variables and the response can be described by

$$Y = f(x_1, x_2, \ldots, x_p) + \epsilon \tag{2.1}$$

where f may be a smooth function of $x_1, x_2, \ldots, x_p$ and ϵ represents a random noise in the observable response. The expected response $E(Y) = f(x_1, x_2, \ldots, x_p)$ is called *response surface*. In Fig. 2.2, the usually graphical representation of $E(Y)$ in the case of two variables, x_1 and x_2, is plotted.

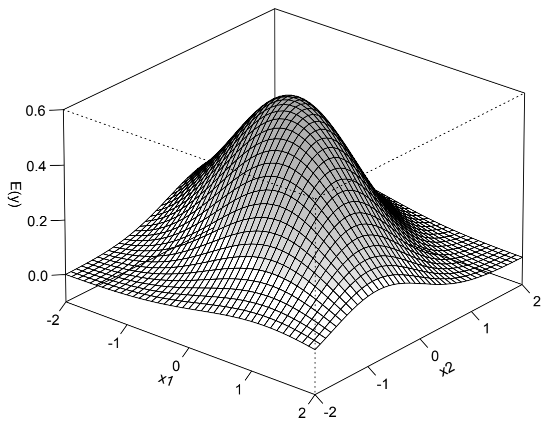


Figure 2.2: A response surface for two factor design.

Because the form of the true response function f is unknown we must approximate it. Usually, a polynomial model in some relatively small region of the independent variable space is appropriate. The general motivation for a polynomial approximation for the true response function f is based on the Taylor series expansion.

The simplest polynomial model to explore the space approximating the Function 2.1 is the first-order polynomial model,

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon \tag{2.2}$$

where p variables are supposed to affect the response in a linear way without interactions, for each k the parameters β_k measure their influence of variable k on y.

The first-order polynomial model is likely to be appropriate when we are interested in approximating the true response surface over a relatively small region of the independent variable space in a location where there is little curvature in f. Often the curvature in the true response surface is strong enough that the first-order model is inadequate. A second-order polynomial model will likely be required in these situations.

The second-order polynomial model is a flexible model to describe experimental data in which nonlinear terms are present. The nature of the response surface depends on the signs and magnitudes of the coefficients in the following model:

$$Y = \beta_0 + \sum_{i=1,\ldots,p} \beta_i x_i + \sum_{i=1,\ldots,p} \beta_{ii} x_i^2 + \sum_{i<j} \beta_{ij} x_i x_j + \epsilon. \tag{2.3}$$

The approach of least squares estimates for the β parameters is then generally used and the adequacy of the fitted surface is evaluated with the ANOVA methodology.

In the sequel of the chapter we will use these concepts of RSM that are related to linear regression analysis.

2.3 Multidimensional Scaling

Another important technique that we use in this part of the work is the Multidimensional Scaling. From a general point of view, *multidimensional scaling* (MDS) is a set of methods for discovering hidden structures in multidimensional data. Based on a dissimilarity matrix derived from variables measured on objects (points) as input entity, these dissimilarity measures are mapped on a low dimensional (typically two or three dimensions) spatial representation.

The traditional way of performing MDS is referred to as classical scaling [71] which is based on the assumption that the dissimilarities are precisely Euclidean distances without any additional transformation.

Starting from an $N \times N$ dissimilarity matrix ($\Delta = [\delta_{ij}]$), where N is the number of points (objects) and δ_{ij} is the dissimilarity between point i and j we can apply MDS. The dissimilarity matrix (Δ) should agree with the following constraints:

– symmetry ($\delta_{ij} = \delta_{ji}$);

– nonnegativity ($\delta_{ij} \geq 0$);

– zero diagonal elements ($\delta_{ii} = 0$).

The objective of MDS techniques is to construct a configuration of the given data in a low dimensional Euclidean space, while each distance between a pair of points in the configuration is approximately equal to the corresponding dissimilarity value. The output of MDS algorithms could be represented as an $N \times L$ configuration matrix X, whose rows represent data points x_i ($i = 1, ..., N$) in an L-dimensional space.

To evaluate how well the given points are configured in the L-dimensional space we can use suggested objective functions of MDS, for instance the Stress [46] function, defined as follows:

$$\sigma(X) = \sum_{1<j\leq N} w_{ij}(\delta_{ij}(X) - \delta_{ij})^2 \qquad (2.4)$$

where $1 < j \leq N$, $\delta_{ij}(X)$ is equal to the (i, j)-th element of the X matrix and w_{ij} are weights that in the sequel will be put equal to 1.

The MDS problem could be considered as a non linear optimization problem, which minimizes the Stress function in the process of configuring L-dimensional mapping from the set of objects into L-dimensions ($\mathbb{R}^L$).

2.3.1 MAjorizing a COmplicated Function (SMACOF)

MAjorizing a COmplicated Function ($SMACOF$) [15] is an iterative majorization algorithm to solve MDS problem with Stress criterion. The iterative majorization procedure of the SMACOF could be thought of as Expectation-Maximization (EM) [53] approach. Although SMACOF has a tendency to find local minima due to his hill-climbing attribute, it is still a powerful method since it is guaranteed to decrease the Stress (σ) criterion monotonically. In the following part of this section we briefly explain how SMACOF works.

Consider a $N \times N$ matrix Δ of dissimilarities based on observed data. Δ is symmetric, non-negative, and has zero diagonal. The problem we solve is to locate $i = 1, \ldots, n$ points in a low dimensional Euclidean space in such a way that the distances between the points approximate the given dissimilarities δ_{ij}. Thus we want to find a $N \times L$ configuration X such that $\delta_{ij}(X) \approx \delta_{ij}$, where

$$\delta_{ij}(X) = \sqrt{\sum_{s=1}^{l} (x_{is} - x_{js})^2}.$$

Considering the Equation 2.4, we follow [17], so the Stress criterion can be decomposed as:

$$\begin{aligned} \sigma(X) &= \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}\delta_{ij}^2 + \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}\delta_{ij}^2(X) - 2\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}\delta_{ij}\delta_{ij}(X) = \\ &= \eta_\delta^2 + \eta^2(X) - 2\rho(X). \end{aligned}$$

From [15] and [17], we can write

$$\sigma(X) \leq 1 + \mathbf{tr}\ X^\top V X - 2\mathbf{tr}\ X^\top B(Y) Y = \tau(X, Y).$$

In the case of $w_{ij} = 1$, V is equal to $N(I - \frac{ee^\top}{N})$ where $e = (1, \ldots, 1)^\top$ is a vector whose length is N. Y is a supporting point which is a $N \times P$ matrix of configurations and $B(Y)$ has elements equal to $-\frac{\delta_{ij}}{\delta_{ij}(Y)}$ when $i \neq j$. 0 if $\delta ij(Y) = 0$ and $i \neq j$. $-\sum_{i \neq j} b_{ij}$ if $i = j$ where b_{ij} is the (i, j)-th element of the B matrix.

SMACOF iteratively minimizes the Stress function until a certain limit is reached.

2.4 Results of the SMACOF

In this section we present the results obtained with SMACOF on the similarity matrix S introduced in Chapter 1.

First of all we transform the similarity matrix to dissimilarity matrix using.

$$\Delta = 1 - S.$$

Therefore, we apply the SMACOF to reduce the 95×95 Δ matrix in a 95×1 configuration matrix X, whose rows represent each pseudo-domains x_i $(i = 1, 2, 3, ..., 95)$ in $1-$dimensional space.

In Table 2.1, the 95×1 configuration matrix X containing the $\delta_{ij}(X)$ is reported. The pseudo-domains are ordered based on the new configuration. SMACOF minimizes the Stress function in such a way to reduce the configuration matrix. The minimum value of the Stress function is 0. The amount of Stress is used for judging the goodness of fit of the SMACOF solution: a small Stress value indicates a good fitting solution, whereas a high value indicates a bad fit. In this case, the value of the Stress function is 0.24976.

The value is not really small and we can not consider this configuration a good one.

PsD	X	PsD	X	PsD	X	PsD	X	PsD	X
90	-1.139	32	-0.560	57	-0.156	11	0.276	4	0.726
47	-1.075	27	-0.540	53	-0.150	43	0.302	6	0.732
10	-1.047	17	-0.536	85	-0.104	5	0.351	13	0.745
3	-1.012	34	-0.480	28	-0.102	69	0.363	59	0.747
42	-0.999	63	-0.460	64	-0.096	23	0.370	93	0.768
20	-0.921	67	-0.442	37	-0.057	66	0.380	68	0.780
21	-0.891	72	-0.441	29	-0.050	14	0.382	31	0.785
35	-0.878	75	-0.427	81	-0.013	38	0.408	49	0.851
19	-0.860	88	-0.403	65	-0.007	89	0.432	76	0.865
7	-0.849	50	-0.391	36	0.034	73	0.465	46	0.937
24	-0.815	30	-0.381	41	0.069	84	0.503	15	1.011
51	-0.759	87	-0.370	60	0.091	52	0.520	61	1.047
39	-0.753	95	-0.345	92	0.093	56	0.521	54	1.057
8	-0.710	70	-0.283	48	0.098	77	0.601	12	1.153
2	-0.684	16	-0.278	71	0.140	62	0.605	25	1.201
79	-0.680	9	-0.270	78	0.155	22	0.628		
45	-0.656	18	-0.244	82	0.156	86	0.631		
1	-0.621	80	-0.221	94	0.196	26	0.657		
40	-0.600	58	-0.176	55	0.202	83	0.669		
91	-0.574	74	-0.161	44	0.259	33	0.700		

Table 2.1: Domains ordered using the $1-$dimensional representation. PsD stands for pseudo-domains and X is the 95×1 configuration matrix

Anyway, we study the quality of the 1-dimensional configuration by examining the response (Score) values for groups of enzymes where only one pseudo-domain changes. Considering three constant positions we can suppose that the response is affected only by the position where the pseudo-domains are free to change.

In tables 2.2 and 2.3, we select two groups of pseudo-domains that result to be near in the matrix X. We notice that the neighbouring pseudo-domains have a response with high fluctuation. In accordance with biological considerations, we were expecting to find a different behaviour of the response namely that a group of neighbouring pseudo-domains would affect the response in a similar way. In other words, we were expecting a smooth behaviour of the response, that it is not confirmed by our empirical analysis of the 1-dimensional configuration where from a high peak, the response drops to a minimum and, suddenly, increases up to a new peak, namely a high value of the score. We remind that the range of the score, or response, is from 0 to 1000 (see Chapter 1). Hence, we decided to move from a 1-dimensional configuration to a 2-dimensional one.

$Pos1$	$Pos2$	$Pos3$	$Pos4$	$Score$
8	64	75	83	568
8	37	75	83	490
8	29	75	83	686
8	81	75	83	578

Table 2.2: Position 2 is the free position. Group of 4 consequent pseudo-domains.

$Pos1$	$Pos2$	$Pos3$	$Pos4$	$Score$
8	73	75	83	561
8	84	75	83	714
8	52	75	83	541
8	56	75	83	611
8	77	75	83	541
8	62	75	83	614

Table 2.3: Position 2 is the free position. Group of 6 consequent pseudo-domains.

Then, we consider a 95×2 configuration matrix X, whose rows represent each pseudo-domain x_i $(i = 1, 2, 3, ..., 95)$ in a $2-$dimensional space. With a 95×2 configuration matrix X, the value of the Stress function reached by the minimization procedure of SMACOF is 0.12521. In this case the Stress value is smaller with respect to the previous one then this configuration better fits the 95×95 Δ matrix. Furthermore, we compare the 1-dimensional configuration and the 2-dimensional configuration using the Shepard diagram. The Shepard diagram displays the relationship between the dissimilarities and the distances of the point configuration. The ideal location for the points in a Shepard diagram is a monotonically increasing line describing the so-called disparities, the optimally scaled dissimilarities. Less spread in this diagram implies a good fit. In Fig. 2.3 we can see the Shepard diagram for the 1-dimensional configuration and the 2-dimensional configuration. In the case of the 1-dimensional configuration (Fig. 2.3 (a)), we notice that the point are spread around the line. Instead, in Fig. 2.3 (b) the points are more close to the monotonically increasing line. This result suggests us to consider the 2-dimensional configuration as the final configuration.

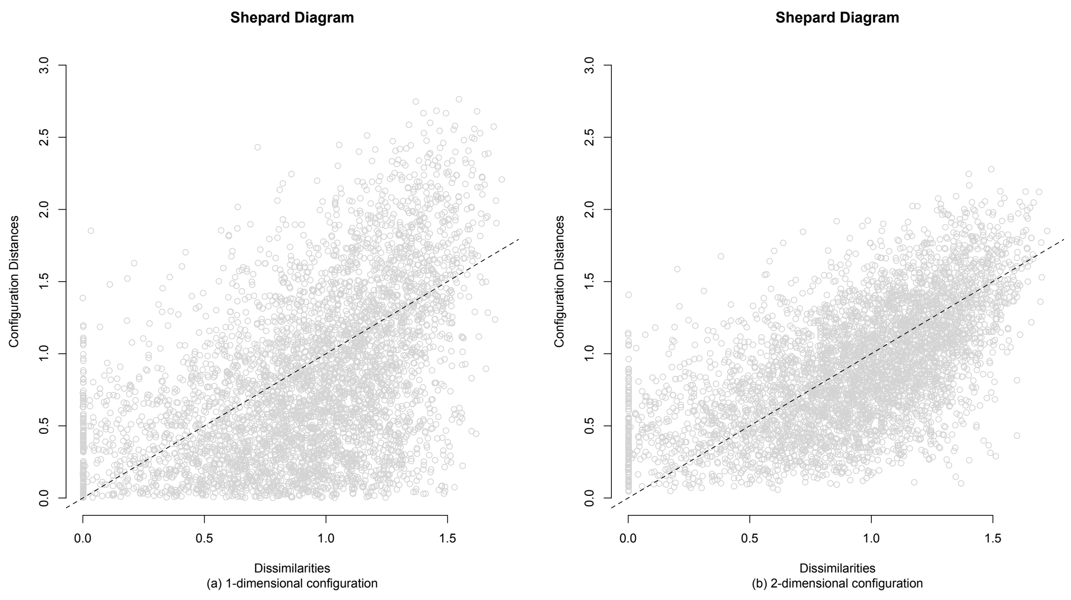


Figure 2.3: 1-dimensional configuration vs 2-dimensional configuration.

The configuration assumed by the pseudo-domains is shown in figure 2.4.

It is possible to notice the presence of some clusters in the right part of the plot. We suppose that a cluster is formed by pseudo-domains with comparable secondary structure. Therefore, considering groups of enzymes where only one pseudo-domain changes, we suppose that pseudo-domains in a cluster do not have a different effect on response because they share common biological features. In the plot it is possible to notice some clusters as $[20, 8, 88, 72, 95, 39]$, $[7, 67, 27, 18]$, $[35, 58]$, $[52, 90, 42]$ e $[28, 81]$. The remaining pseudo-domains do not form evident significant clusters. Some exception are present, for example $[33, 86]$. Using this representation some

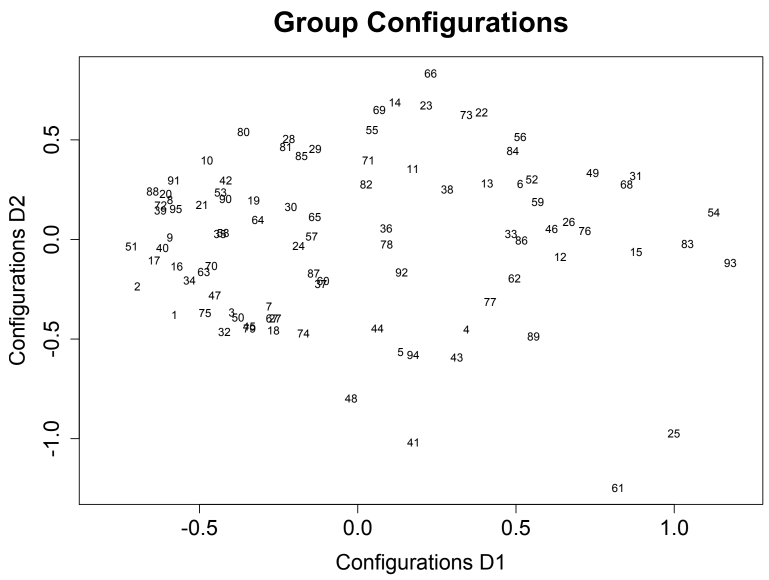


Figure 2.4: 2−dimensional configuration.

pseudo-domains are considered totally different, as 51, 93, 41 e 66.

$Pos1$	$Pos2$	$Pos3$	$Pos4$	$Score$
8	20	75	83	551
8	8	75	83	602
8	88	75	83	552
8	95	75	83	545
8	39	75	83	549

Table 2.4: Position 2 is the free position. Cluster [20, 8, 88, 72, 95, 39] is considered.

As shown from tables 2.4, 2.5 and 2.6 with the 2−dimensional representation the variation of the response is minimum in each cluster. In table 2.4, we consider the cluster composed by the pseudo-domains [20, 8, 88, 72, 95, 39] and we notice that the response is not affected too much changing the pseudo-domains in the sequence. In tables 2.5 and 2.6 not all the pseudo-domains in the clusters are reported because the enzymes with those pseudo-domains have been not evaluated. The trend of the response confirms the biological considerations that have been done at the beginning of this analysis.

$Pos1$	$Pos2$	$Pos3$	$Pos4$	$Score$
8	27	75	83	521
8	67	75	83	533

Table 2.5: Cluster $[7, 67, 27, 18]$ is considered.

$Pos1$	$Pos2$	$Pos3$	$Pos4$	$Score$
8	28	75	83	568
8	81	75	83	578

Table 2.6: Cluster $[28, 81]$ is considered.

From our analysis, we conclude that the 2-dimensional configuration is a satisfactory representation of the 95 pseudo-domains. The Multidimensional scaling, in particular the SMACOF approach, allows us to transform our discrete problem into a continuous one. In fact, we can represent the 95 pseudo-domains in a 2-dimensional Euclidean space. Therefore, the sequence (enzyme) composed by 4 positions can be seen as a value from $\mathbb{R}^8$ that consists of 8 real numbers that specifies the coordinates of a point in 8-dimensional space.

2.5 Regression Analysis

In this section we present the results obtained applying polynomial regression to the new representation of our problem. With this representation we consider as input variables of the model the 8 coordinates that compose an enzyme and the response is the Score explained in Chapter 1.

We start our analysis with a linear model:

$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_8 x_8 + \epsilon,$

with $-0.7149 \le x_i \le 1.1776$ $(i = 1, 3, 5, 7)$ and $-1.2468 \le x_j \le 0.8341$ $(j = 2, 4, 6, 8)$.

We estimate the unknown parameters β_i using the least squares approach and we fit the model on a data set composed by 96 random chosen enzymes. The resulting model is summarized in Table 2.7

	$Estimate$	$Std.Error$	$Lower$	$Upper$
Intercept	372.385	6.644	359.584	385.186
x_1	-55.775	13.302	-82.215	-29.335
x_2	-5.071	15.388	-35.657	25.515
x_3	35.969	13.464	9.207	62.731
x_4	74.271	16.195	42.081	106.461
x_5	8.584	14.053	-19.349	36.517
x_6	-58.389	18.446	-95.054	-21.724
x_7	66.096	11.746	42.729	89.463
x_8	29.174	14.965	-0.572	58.920

Table 2.7: Resulting Linear Model. Lower and Uppers refer to the 95% confidence intervals for each regression parameters.

In Table 2.7, the value for each β_i, the standard errors and the 95% confidence intervals for each estimations are shown. We notice that the confidence interval for β_8 is really wide in the sense that the upper limit is about 100 times larger that the lower limit and we are not really confident what the exact effect of growth on input variables x_8 is.

Hence, we determine if there is a linear relationship between the response variable y and a subset of the regression variables. In general, we use the test for significance of regression. The appropriate hypotheses are $H_0 : \beta_1 = \beta_2 = \ldots = \beta_k = 0$ and $H_1 : \beta_i \neq 0$ for at least on i.

SV	SS	DF	MS	F_0	$p-value$
Regression	339374.3	8	42421.79	11.193	9.352×10^{-11}
Residual	329731.3	87	3790.015		
Total	669105.6	95			

Table 2.8: Test for Significance of Regression. SV stands for Source of Variation, SS for Sum of Squares, DF for Degree of Freedom and MS for Mean Square.

In Table 2.8 is shown the analysis of variance and if we select an $\alpha = 0.05$ then we reject $H_0 : \beta_1 = \beta_2 = \ldots = \beta_k = 0$ in fact the $p-value$ for F_0 is considerable smaller than α.

Furthermore, we investigate if some input variables can be dropped from the model. We use the test on individual regression coefficients where the null hypothesis is $H_0 : \beta_i = 0$. If $H_0 : \beta_i = 0$ is not rejected, then this indicates that x_i can be deleted from the model. Using the t-test and an $\alpha = 0.05$ we notice that input variables x_2 ($p - value = 0.743$) and x_5 ($p - value = 0.543$) can be dropped from the model. The same resulting model is reached also using a forward selection based on the minimization of the Akaike Information Criterion (AIC).

Hence, we obtained a new model with 6 input variables (x_1, x_3, x_4, x_6, x_7, x_8) plus the intercept. The model is summarized in Table 2.9.

	Estimate	*Std.Error*	*Lower*	*Upper*
Intercept	372.561	6.581	59.485	385.637
x_1	-55.374	13.033	-81.270	-29.478
x_3	33.638	12.746	8.312	58.964
x_4	71.500	15.538	40.626	102.374
x_6	-57.193	18.177	-93.310	-21.076
x_7	65.711	11.478	42.904	88.518
x_8	27.571	14.618	-1.475	56.617

Table 2.9: Resulting Linear Model. Lower and Uppers refer to the 95% confidence intervals for each regression parameters.

In this case F_0 is equal to 15.1 and the $p - value$ for F_0 is 7.538×10^{-12}. As in the previous case, we reject the null hypothesis. In this case, for all the individual regression coefficients we reject the hypothesis $H_0 : \beta_i = 0$. Now, a further investigation is to understand how well the model fits the data. We use the *coefficient of determination* R^2 that is a measure of the amount of reduction in the variability of y obtained by using the regressor variables x_1, x_3, x_4, x_6, x_7, x_8 in the model. We calculate the adjusted R^2. In this case, the linear model explains about 47.1 % of the variability observed in the Score. However, we use our estimated model on 96 experimental points (enzymes) to predict the value of 10 points not included in the initial data set, available at the moment of the analysis. We use this new data set to check the ability of the model to predict the response of the system. In Tab. 2.10, it is possible to see the predictions obtained using the last model.

In Figure 2.5 is shown the real Score of each enzyme against the predic-

	Pos1	*Pos2*	*Pos3*	*Pos4*	*Score*	*Prediction*
Enzyme1	62	59	80	24	517	333.6
Enzyme2	8	22	11	88	417	408.1
Enzyme3	79	22	11	58	534	403.0
Enzyme4	2	59	80	57	470	403.6
Enzyme5	24	22	59	1	515	382.5
Enzyme6	14	74	11	13	256	341.0
Enzyme7	8	54	11	37	342	418.9
Enzyme8	24	40	13	90	508	321.2
Enzyme9	77	59	76	22	578	422.8
Enzyme10	18	29	27	13	510	472.5

Table 2.10: Comparison between real score and predicted one for the considered polynomial model.

tions. Moreover the 95 % prediction interval is plotted.

This result does not seem to be satisfactory in our case. This is probably due, from a biological point of view, because the pseudo-domains have a strong interactions between them. These interactions are not considered from our model. We decide to include in the model also all the interactions between the input variables. The model has the following formal form:

$$Y = \beta_0 + \sum_{i=1,\dots,8} \beta_i x_i + \sum_{i<j} \beta_{ij} x_i x_j + \epsilon. \tag{2.5}$$

The interaction between the variables are 24 because we do not consider the interaction between coordinates ($i.e.$ x_1, x_2) that identify 95 pseudo-domains in the same position ($i.e.$ first position) in the sequence (enzyme). Anyway only few of them are significant in the model. We report the model after a forward selection based on the minimization of the AIC and the fitted model (from now we call this model M1) is summarized in Table 2.11.

We notice that the 95% confidence intervals for each interactions is wide so we are not really confident on the effect of interactions in the response. We analyse the variance (Table 2.14) and we do the test of significance of regression. As in the previous case, we accept the alternative hypothesis, $H_1 : \beta_i \neq 0$ for at least one i. The R^2_{adj} is equal to 0.5563 so we do not have

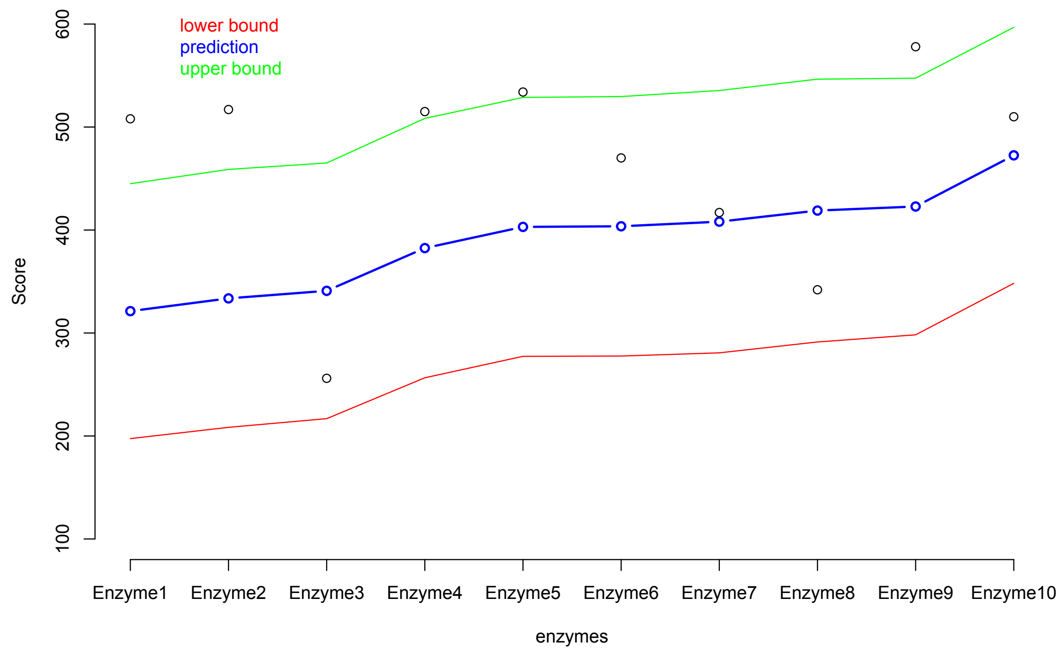


Figure 2.5: The real Score of each enzyme is shown against the predicted Score. The black dots represent the real score. The model is the one with only the significant principal effects.

a big incrementation with respect to the previous model.

Anyway, looking at the confidence intervals, we notice that some coefficients are not significant.

We decide to restart the analysis, starting from the model in the form 2.5, and drop all the non significant coefficients (from now we call this model M2). The resulting model is shown in Table 2.13. In this case, the R^2_{adj} is equal to 0.5260.

As in the previous analysis, we use both estimated models on 96 experimental points (enzymes) to predict the value of the same 10 points used before. Also in this case, the aim of this analysis is to check the ability of the models to predict the response of the system. The results seems to be not satisfactory. In Tab. 2.15, it is possible to see that the predictions obtained

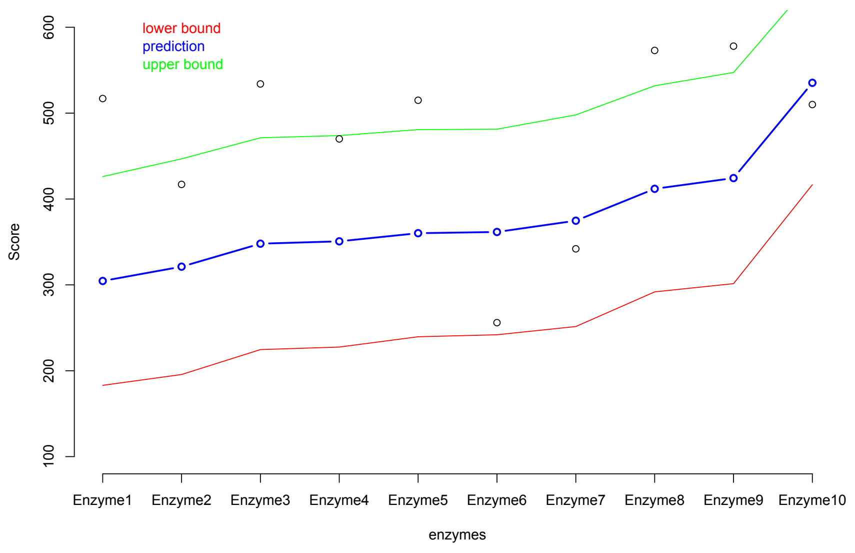
	Estimate	*Std.Error*	*Lower*	*Upper*
Intercept	376.679	6.525	363.694	389.664
x_1	-45.134	12.765	-70.537	-19.731
x_2	-15.130	14.795	-44.573	14.313
x_3	37.926	13.022	12.011	63.841
x_4	73.751	15.260	43.383	104.119
x_5	15.231	13.169	-10.976	41.438
x_6	-77.216	18.740	-114.510	-39.922
x_7	57.213	11.370	34.586	79.840
x_8	31.923	14.023	4.016	59.830
x_1:x_7	-67.165	23.939	-114.805	-19.525
x_2:x_3	-57.002	30.007	-116.718	2.714
x_2:x_4	79.009	47.227	-14.976	172.994
x_3:x_6	-58.128	40.556	-138.837	22.581
x_4:x_6	-128.176	44.633	-216.998	-39.356
x_4:x_7	56.458	26.353	4.014	108.902
x_5:x_8	29.174	28.857	-109.516	5.338

Table 2.11: M1: resulting Linear Model with significant interaction between input variables. Lower and Uppers refer to the 95% confidence intervals for each regression parameters.

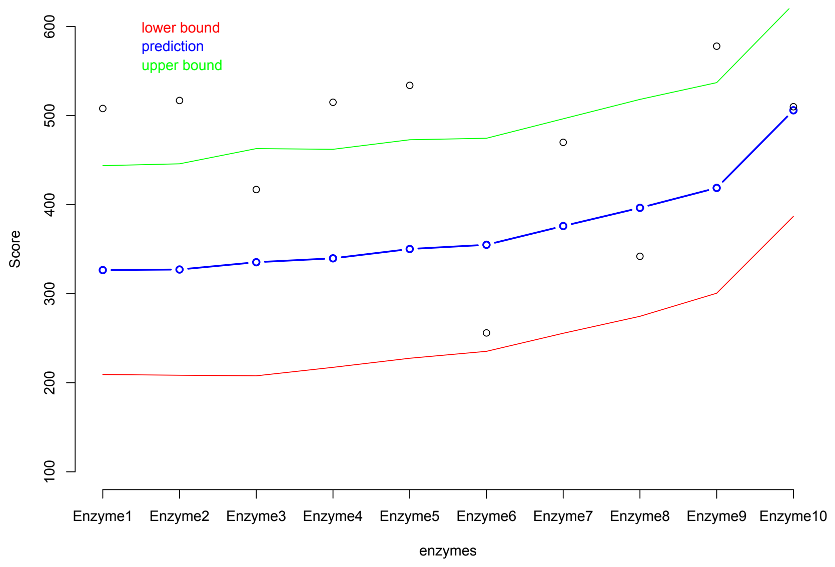
using both models are not good.

From Figure 2.6, we can notice that the 50% of the predicted score is inside the bounds, anyway some of the points are really near to the upper or lower bound. The other 50% are over the upper bound, we can suppose that the models overestimate the influence of the input variables on the response. With bounds closer to the predicted values probably the prediction could be worse with the exception of two enzymes.

The limited amount of data limits the reliability of the models. Another indicator is R^2_{adj}, its small value suggests us that the regression models are not good. We have done a preliminary analysis also with a second order polynomial model but all the quadratic terms were not significant so we have decided to stop this analysis. The few experimental points and the non linearity of the phenomena under study reduce the reliability of the models.



(a)



(b)

Figure 2.6: The real Score of each enzyme is shown against the predicted Score obtained by (a) M1 and (b) M2. The black dots represent the real score.

SV	SS	DF	MS	F_0	$p-value$
Regression	419091.1	15	27939.41	8.94	1.028×10^{-11}
Residual	250014.5	80	3125.181		
Total	669105.6	95			

Table 2.12: M1: test for Significance of Regression. SV stands for Source of Variation, SS for Sum of Squares, DF for Degree of Freedom and MS for Mean Square.

	$Estimate$	$Std.Error$	$Lower$	$Upper$
Intercept	374.643	6.269	362.181	387.105
x_1	-46.232	12.672	-71.423	-21.041
x_3	27.346	12.452	2.592	52.100
x_4	65.401	15.117	35.350	95.453
x_6	-58.387	17.817	-93.806	-22.968
x_7	60.632	11.290	38.188	83.076
x_8	32.843	14.013	4.986	60.700
x_1:x_7	-55.077	24.296	-103.376	-6.778
x_4:x_6	-117.593	44.573	-206.201	-28.985
x_4:x_7	47.061	26.899	-6.412	100.534

Table 2.13: M2: resulting Linear Model. Lower and Uppers refer to the 95% confidence intervals for each regression parameters.

2.6 Some Conclusions

In this chapter we describe the possibility to face the Enzyme Engineering Design using classical statistical approaches.

For this purpose we develop a 3-stage method based on some biological consideration and two well-known statistical methods. We show how it is possible to transform a discrete problem to a continuos one using Multidimensional Scaling. In fact we demonstrate that it is possible to represent each pseudo-domain as a value from $\mathbb{R}^2$.

This result allows us to tackle the problem using a linear regression model. We fit different models to a dataset of 96 randomly chosen enzymes and we notice that this approach is not satisfactory in terms of prediction and reliability. We conclude that a more thorough investigation of statistical

SV	SS	DF	MS	F_0	$p - value$
Regression	381994.0	9	42443.77	12.71	1.275×10^{-12}
Residual	287111.7	86	3338.5087		
Total	669105.6	95			

Table 2.14: M2: test for Significance of Regression. SV stands for Source of Variation, SS for Sum of Squares, DF for Degree of Freedom and MS for Mean Square.

	$Pos1$	$Pos2$	$Pos3$	$Pos4$	$Score$	$M1 : Pred$	$M2 : Pred$
Enzyme1	62	59	80	24	517	304.6	327.2
Enzyme2	8	22	11	88	417	321.2	335.4
Enzyme3	79	22	11	58	534	348.0	350.2
Enzyme4	2	59	80	57	470	350.7	376.0
Enzyme5	24	22	59	1	515	360.2	339.8
Enzyme6	14	74	11	13	256	361.6	354.9
Enzyme7	8	54	11	37	342	374.7	396.4
Enzyme8	24	40	13	90	508	327.4	326.6
Enzyme9	77	59	76	22	578	424.4	418.8
Enzyme10	18	29	27	13	510	535.3	506.0

Table 2.15: Comparison between real score and predicted one for M1 (*M1:Pred*) and M2 (*M2:Pred*).

models for discrete problems is necessary.

For this reason, the next part of this thesis is devoted to a study of more accurate models for high dimensional spaces and the concepts of Combinatorial Optimization, a well-known technique in the presence of discrete variables.

Chapter 3

Statistical Models for High Dimensional Problems

3.1 The “*large p, small n*“ Problem

In the last decade, modern scientific technology is providing a class of complex problems that typically involve data that are high dimensional. These complex experiments involve a vast number of variables, a high dimensional search space and a large number of economical constraints that limit the ability of classical statistical techniques to tackle the problems.

For most of the time, the primary motivation of statistical studies has been to find solutions when a large number of experimental units were measured and a small number of features had to be considered. Nowadays the situation is changing. More and more frequently the variables involved in a problem reach high numbers. Moreover, limited budgets reduce the possibilities to experiment several combinations of modalities for the variables. It is very common that a vast number of variables and a large experimental space are involved, and only few experimental points can be tested and evaluated.

Modern applications of statistical theory and methods are devoted to this new problem. They can involve extremely large datasets, often with an enormous number of measurements on each of a comparatively small number of experimental units. New methodology has emerged in response and papers that illustrate a number of these recent developments [47] [38] [54] are present in the litaterature.

If, informally, we let p denote the dimension of what is “unknown“ and let n denote the cardinality of what is known, the key scenarios to be investigated can be described as “*large p, small n*“ or in some case as “*large p, large n*“; the theory for the former scenario would assume that p goes to infinity faster than n and for the latter would assume that p and n go to infinity at the same rate [39].

In practice, n will generally correspond to the number of experimental units on which data are available; for p, however, there are at least two interpretations, with a strong relation between them. The first interpretation is the measure of complexity of the model to be fitted to the data. However, that is often determined by the dimension of the data as given by the number of variables recorded for each experimental unit.

In our case we shall assume that we are in the case that n, number of experimental units, is small and p, number of parameters involved in the experiment, is high. As we can understand, the number of all the possible combinations of the p parameters and their levels can be enormous, creating an incredibly complex search space.

In this contest we need statistical models and techniques able to get as much information as possible from the few data points. In the literature there are models that assume that the number of really influential parameters k, is much smaller than the nominal number p involved in the experiment, these are for instance the additive models, the “Least Absolute Shrinkage and Selection Operator“ (LASSO) models and others.

Bayesian Network and Naïve Bayes Network are two other efficient techniques in the case of high dimensionality.

In this chapter we are going to present an overview of some the most important techniques for “*large p, small n*“ problems. In the sequel of this thesis we will apply only one of them, namely a modified version of the Naïve Bayes Classifier.

3.2 A First Step: Ridge Regression

Consider a set of data in the form of measurements on n individuals, $\bar{x}_i$, y_i; $i = 1, ..., n$, where $\bar{x}_i$ is a set of predictors and y_i is a response. A convenient notation, using vector and matrix, is possible to represent the model for the complete set of n data pairs,

$$y = X\beta + \epsilon \ , \tag{3.1}$$

where the $n \times 1$ vector contains the response, the vector β contains the p parameters except for σ^2, the $n \times 1$ vector ϵ contains the error and the $n \times p$ design matrix X completes the model.

The standard way of estimating the unknown slope and intercept in β is to use least-squares approach and obtain

$$\hat{\beta} = arg \min_{\boldsymbol{\beta}} \sum_i (y_i - \beta_1 - \beta_2 x_i)^2,$$

which means that $\hat{\beta}$ is the minimizer of the sum of squares function on the right hand side. In the general vector-matrix notation, this case can be written in terms of Euclidean distance $|\cdot|$, as

$$\hat{\beta} = arg \min_{\boldsymbol{\beta}} |y - X\beta|^2.$$

$\hat{\beta}$ satisfies

$$X^\top X\hat{\beta} = X^\top y,$$

and

$$\hat{\beta} = (X^\top X)^{-1} X^\top y,$$

the explicit formula in the second equation being available provided that the matrix $X^\top X$ can be inverted.

There is another important interpretation of $\hat{\beta}$. Our assumptions mean that y is drawn form $N_n(X\beta, \sigma^2 I)$, in which N_n denotes an n-variate multivariate Gaussian distribution, with $X\beta$ as the vector of means and $\sigma^2 I$ as the covariance matrix, and where I is the $n \times n$ indentity matrix, then the density function for y is then

$$p(y|X, \beta) = \{\sqrt{2\pi\sigma^2}\}^{n/2} exp\{-\frac{|y - X\beta|^2}{2\sigma^2}\}.$$

The data provide y and X. When viewed as a function of the parameters, this is now called the likelihood function, and,

$$\hat{\beta} = arg \max_{\boldsymbol{\beta}} p(y|X, \beta).$$

Thus, $\hat{\beta}$ is the so-called maximum likelihood estimator of β.

The usual estimation procedure for the unknown β is unbiased and has minimum variance in the class of unbiased linear estimators. This estimation procedure is a good one if $X^\top X$ is nearly a unit matrix. If $X^\top X$ is not nearly to a unit matrix, the least square estimations are sensitive to a number of limitations. These limitations are due to the non linearity of the phenomena under study and they can reduce the reliability of the true model. Then the least squares estimations often do not make sense when put into the contest of the physics, chemistry, and engineering of the process which is generating the data.

A possible solution of the previous problems is to consider a transformation of the $X^\top X$ matrix.

A. E. Hoerl first suggested in 1962 [36] that to control the inflation and general instability associated with the least squares estimates, one can use

$$\begin{aligned} \hat{\beta}^* &= (X^\top X + kI)^{-1} X^\top Y; k \geq 0 \qquad (3.2) \\ &= W X^\top Y. \qquad (3.3) \end{aligned}$$

The positive scalar k is called ridge parameter or regularization constant and the family of estimates given by $k \geq 0$ has many mathematical similarities with the portrayal of quadratic response functions [36]. For this reason, estimation and analysis around (3.2) has been labeled ridge regression [37].

The relationship of a ridge estimate to an ordinary estimate is given by the alternative form

$$\begin{aligned}\hat{\beta}^* &= [I + k(X^\top X)^{-1}]^{-1}\hat{\beta} \\ &= Z\hat{\beta}.\end{aligned} \tag{3.4}$$

Some properties of $\hat{\beta}^*$, W, and Z are

(i) Let $\xi_i(W)$ and $\xi_i(Z)$ be the eigenvalues of W and Z, respectively. Then

$$\xi_i(W) = 1/(\lambda_i + k),$$

$$\xi_i(Z) = \lambda_i/(\lambda_i + k), \tag{3.5}$$

where λ_i are the eigenvalues of $X^\top X$. These results follow directly the definition of W and Z in (3.3) and (3.4) and the solution of the characteristic equations $|W - \xi I| = 0$ and $|Z - \xi I| = 0$.

(ii)

$$Z = I - k(X^\top X + kI)^{-1} = I - kW. \tag{3.6}$$

The relationship is verified by writing Z in the alternative form $Z = (X^\top X + kI)^{-1}X^\top X = WX^\top X$ and multiplying both sides of (3.6) on the left by W^{-1}.

(iii) $\hat{\beta}^*$ for $k \neq 0$ is shorter than $\hat{\beta}$, *i.e.*,

$$(\hat{\beta}^*)^\top(\hat{\beta}^*) < \hat{\beta}^\top\hat{\beta}, \tag{3.7}$$

Let us show this result. By definition $\hat{\beta}^* = Z\hat{\beta}$. From its definition and the assumptions on $X^\top X$, Z is clearly symmetric positive definite. Then the following relation holds:

$$(\hat{\beta}^*)^\top(\hat{\beta}^*) < \xi^2_{max}(Z)\hat{\beta}^\top\hat{\beta},$$

But $\xi_{max}(Z) = \lambda_1/(\lambda_1 + k)$ where λ_1 is the largest eigenvalue of $X^\top X$ and (3.7) is established. From (3.5) and (3.6) it is seen that $Z(0) = I$ and that Z approaches 0 as $k \to \infty$.

For an estimate $\hat{\beta}^*$ the residual sum of squares is

$$\phi^*(k) = (Y - X\hat{\beta}^*)^\top(Y - X\hat{\beta}^*),$$

which can be written in the form

$$\phi^*(k) = Y^\top Y - (\hat{\beta}^*)^\top X^\top Y - k(\hat{\beta}^*)^\top(\hat{\beta}^*).$$

The expression shows that $\phi^*(k)$ is the total sum of squares less the regression sum of squares $\hat{\beta}^*$ with a modification depending upon the squared length of $\hat{\beta}^*$.

We can say that an estimation based on the matrix $[X^\top X + kI]$, $k \geq 0$ rather than on $X^\top X$, is a procedure that can be used to help circumvent many of the difficulties associated with usual least squares estimates. In particular, the procedure can be used in the case of non linearity of the particular set of data being considered, and it can be used to obtain a point estimate with a smaller mean square error.

3.3 Least Absolute Shrinkage and Selection Operator (LASSO)

Ridge Regression is a continuous process that shrinks coefficients and hence it is more stable: however, it does not set any coefficient to 0 therefore it does not give an easily interpretable model.

Tibshirani in the 1996 [70] proposed a new method for estimation in linear models. The method minimizes the residual sum of squares subject to the sum of the absolute values of the coefficients being less than a constant.

Because of the nature of this constraint it tends to produce some coefficients that are exactly 0 and hence gives interpretable models.

Starting from the set up introduced in Section 3.2, we assume that the x_{ij} are standardized so that $\sum_i x_{ij} = 0$, $\sum_i x_{ij}^2 = 1$.

Letting $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_p)^\top$, the lasso estimate $(\hat{\alpha}, \hat{\beta})$ is defined by

$$(\hat{\alpha}, \hat{\beta}) = arg\min[\sum_{i=1}^{N}(y_i - \alpha - \sum_j \beta_j x_{ij})^2], \tag{3.8}$$

subject to $\sum_j |\beta_j| \leq t$.
Here $t \geq 0$ is a tuning parameter. Now, for all t, the solution for α is $\hat{\alpha} = \bar{y}$.

The parameter $t \geq 0$ controls the amount of shrinkage that is applied to the estimates. Let $\hat{\beta}^0$ be the full least squares estimates and let $t_0 = \sum |\hat{\beta}_i^0|$. Values of $t < t_0$ will cause shrinkage of the solutions towards 0, and some coefficients may be exactly equal to 0.

The motivation for the LASSO came from a proposal of Breiman in 1993 [12].

Breiman's *non-negative garotte* minimizes

$$\sum_{i=1}^{N}(y_i - \alpha - \sum_j c_j \hat{\beta}_j^{ols} x_{ij})^2, \tag{3.9}$$

subject to $c_j \geq 0$, $\sum_j c_j \leq t$.
The garotte starts with the ordinary least square (OLS) estimates and shrinks them by non-negative factors whose sum is constrained. A drawback of the garotte is that its solution depends on both the sign and magnitude of the OLS estimates. In overfit or highly correlated settings, where the OLS estimates behave poorly, the garotte may suffer as a result. In contrast the LASSO avoids the explicit use of the OLS estimates.

Another important point is that the lasso estimate is a non-linear and non-differentiable function of the response values even for a fixed value of t, it is difficult to obtain an accurate estimate of its standard error. One approach is via the bootstrap: either t can be fixed or we may optimize over t for each bootstrap sample. Fixing t is analogous to selecting a best subset, and then using the least squares standard error for that subset.

The LASSO approach is useful when the number of p is high with respect to the available data, and overall in the situation when there is sparsity on the $n \times p$ design matrix.

3.4 Elastic Net

The LASSO has shown success in many situations but it has some limitations. In the $p > n$ case, the LASSO selects at most n variables before it saturates, it means that $p - n$ variables are not consider in the models. Then, important information is missed. This seems to be a limiting feature for a variable selection method. Moreover, if there is a group of variables among which the pairwise correlations are very high, then the LASSO tends to select only one variable from the group and does not care which one is selected. Last limitation is that for usual $n > p$ situations, if there are high correlations between predictors, it has been empirically observed that the prediction performance of the LASSO is dominated by ridge regression [70].

Zou and Hastie in the 2005 proposed a new regularization technique called *elastic net* [72]. The elastic net simultaneously does automatic variable selection and continuous shrinkage, and it can select groups of correlated variables.

First we have to introduce the concept of naïve elastic net. Suppose that the data set has n observations with p predictors. Let $y = (y_1, ..., y_n)^\top$ be the response and $X = (x_1|...|x_p)$ be the model matrix, where $x_j = (x_{1j}, ..., x_{nj})^\top$, $j = 1, ..., p$, are the predictors. After a location and scale transformation, we can assume that the response is centred and the predictors are standardized,

$$\sum_{i=1}^{n} y_i = 0, \qquad \sum_{i=1}^{n} x_{ij} = 0, \quad and \quad \sum_{i=1}^{n} x_{ij}^2 = 1, \qquad for \quad j = 1, 2, ..., p.$$

For any fixed non negative λ_1 and λ_2, Zou and Hastie define the naïve elastic net criterion

$$L(\lambda_1, \lambda_2, \beta) = |y - X\beta|^2 + \lambda_2 |\beta|^2 + \lambda_1 |\beta|_1, \tag{3.10}$$

where

$$|\beta|^2 = \sum_{j=1}^{p} \beta_j^2 \qquad |\beta|_1 = \sum_{j=1}^{p} |\beta_j|.$$

The naïve elastic net estimator $\hat{\beta}$ is the minimizer of equation (3.10):

$$\hat{\beta} = arg \min_{\beta} [L(\lambda_1, \lambda_2, \beta)].$$

This procedure can be viewed as a penalized least squares method. Let $\alpha = \lambda_2/(\lambda_1 + \lambda_2)$; then solving $\hat{\beta}$ on equation (3.10) is equal to:

$$\hat{\beta} = arg \min_{\beta} |y - X\beta|^2, \qquad subject \quad to \quad (1-\alpha)|\beta|_1 + \alpha|\beta|^2 \leq t,$$

for some t. $(1-\alpha)|\beta|_1 + \alpha|\beta|^2$ is called elastic net penalty, which is convex combination of the LASSO and ridge penalty. When $\alpha = 1$, the naïve elastic net becomes a ridge regression. For all $\alpha \in [0, 1)$, the elastic net penalty function is singular (without first derivative) at 0 and it is strictly convex for all $\alpha > 0$, thus having the characteristics of both the LASSO and ridge regression.

With the parameters (λ_1, λ_2) the naïve elastic net solution is

$$\hat{\beta}_1^{(nen)} = \frac{(|\hat{\beta}_i^{(OLS)}| - \lambda_1/2)_+}{1 + \lambda_2} sgn[\hat{\beta}_i^{(OLS)}], \tag{3.11}$$

where $\hat{\beta}^{(OLD)} = X^\top y$ and $z = (|\hat{\beta}_i^{(OLS)}| - \lambda_1/2)_+$ denotes the positive part of the equation, which is z if $z > 0$ otherwise 0.

Naïve elastic net does not perform satisfactory unless it is very close to either ridge regression or the LASSO. An improvement is possible doing a scaling transformation of the coefficient in the following way. Given data (y, X), penalty parameter (λ_1, λ_2) and augmented data (y_*, X_*), the naïve elastic net solves a lasso type problem

$$\hat{\beta}^* = arg \min_{\beta^*} |y^* - X^*\beta^*|^2 + \frac{\lambda_1}{\sqrt{(1-\lambda_2)}}|\beta^*|_1.$$

The elastic net estimates $\hat{\beta}$ is now defined by

$$\hat{\beta}^{elastic} = \sqrt{(1-\lambda_2)}\beta^*,$$

thus

$$\hat{\beta}^{elastic} = (1+\lambda_2)\hat{\beta}^{(nen)}.$$

Such a transformation preserves the variable selection property of the naïve elastic net and is the simplest way to undo shrinkage.

In conclusion, the elastic net is a generalization of the LASSO, which has been shown to be a valuable tool for model fitting and feature extraction. It produces a sparse model with good prediction accuracy, while encouraging a grouping effect.

3.5 Sparse Additive Model (SpAM)

Starting from nonparametric regression it is possible to relax the assumption made by a linear model and to create a model more suitable in high dimensions. Nonparametric regression is defined as

$$y_i = f_j(x_i) + \epsilon_i,$$

where f is a general smooth function.

A more accurate solution is proposed by Hastie and Tibshirani [35] in 1999 that introduces a class of additive models of the form

$$y_i = \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i, . \tag{3.12}$$

This additive combination of univariate functions, one for each covariate X_j, is less general, but can be more interpretable and easier to fit; in particular, and additive model can be estimated by using a co-ordinate descent Gauss-Seidel procedure, called backfitting [35]. An extension of the additive models is the functional ANOVA model

$$y_i = \sum_{1\leq j\leq p} f_j(x_{ij}) + \sum_{j<k} f_{jk}(x_{ij}, x_{ik}) + \sum_{j<k<l} f_{jkl}(x_{ij}, x_{ik}, x_{kl}) + ... + \epsilon_i, \quad (3.13)$$

which allows interactions among the variables. Additive models only have good performance when the number of variables p is not large relative to the sample size. So their usefulness is limited in the high dimensional setting.

A possible solution is the use of Sparse Additive Models (SpAM) [63] that combine idea from sparse linear modeling and additive nonparametric regression in such a way as to extend the advantages of the first kind of models to the additive, nonparametric setting. SpAM is a method for fitting the models that is effective when the number of covariates is larger that the sample size.

The underlying model is the same as in (3.12), but it is impose a sparsity constraint on the index set $\{j : f_j \neq 0\}$ of functions f_j that are not indenticaly zero. This helps to simultaneously encourage smoothness of each component and sparsity across components. As it is possible to understand, the success of this method depends on the initial estimates of component functions f_j.

The SpAM estimation procedure allows the use of arbitrary nonparametric smoothing techniques, and in the case where the underlying component functions are linear, it reduces to the LASSO.

The procedure for fitting the Sparse Additive Model is based on a coordinate descent algorithm derived as follow.

Algorithm 3.5.1: SpAM PROCEDURE($data(x_i, y_i), \lambda$)

```
procedure SpAM BACKFITTING ALGORITHM($(x_i, y_i), \lambda$)
  Initialize  $\hat{f}_j = 0$,  for  $j = 1, ..., p$
  repeat
    compute : $R_j = y - \sum_{k \neq j} \hat{f}_k(x_k)$
    estimate : $P_j = E(R_j | x_j)$
    comment: by smoothing $P_j = S_j R_j$
    estimate : $\hat{s_j} = (1/n) \sum_{i=1}^{n} P_j^2(i)$
    threshold : $\hat{f}_j = [1 - \lambda / \hat{s_j}]_+ P_j$
    comment: soft thresholding
    centre : $\hat{f}_j \leftarrow \hat{f}_j - mean(\hat{f}_j)$
  until convergence
  comment: for each $j = 1, ..., p$
  return $(\hat{f}_j, \hat{m}(x_i) = \sum_j \hat{f}_j(x_{ij}))$
```

To underline that S_j is a linear smoother, such as local linear or kernel smoother. Another important point of the SpAM Backfitting Algorithm is the estimation of $\hat{s_j}$ that is equal to $\frac{1}{\sqrt{n}} |\hat{P}_j| = \sqrt{E(P_j^2)}$.

Moreover the first two steps in the iterative algorithm are the usual backfitting procedure, the remaining steps carry out functional soft thresholding. This algorithm can be seen as a functional version of the coordinate descent algorithm for solving the LASSO.

The SpAM approach can be extended to nonparametric logistic regression for classification.

3.6 Another approach: Bayesian Theory

Considering a parameterized family of probability density function for continuos distribution, or a probability mass functions in the case of discrete distributions, we can define the $f(x|\theta) = f(x_1, ..., x_n|\theta)$ as the joint distribution of the data, where $x = (x_1, ..., x_n)$ stands for a vector of observations $(x_1, ..., x_n)$ and θ the parameter of the function. When we treat the unknown parameter θ as a random variable with distribution $\pi(\theta)$ over the parameter space we can use Bayesian Inference. The $\pi(\theta)$ is called *prior distribution* for θ and represents our degree of belief about θ before we see any data. Having seen the data, it is possible to update our degree of belief using Bayes calculus so that we changed into the *posterior distribution* for θ; all inference procedures are based on this posterior distribution.

Beginning with our prior distribution on θ which we summarize by a (discrete or continuos) probability distribution: $\pi(\theta)$, we then observe some relevant data $(x_1, ..., x_n)$ whose sampling distribution depends on θ. The sampling distribution is the distribution of the data given θ, $f(x|\theta)$. Then, by Bayes calculus we obtain the posterior distribution for θ as:

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{f(x)}$$

where $f(x) = \int f(x|\theta)\pi(\theta)d\theta$ is the marginal distribution of x.

Starting from these basic concepts, the rest of the chapter is dedicated to two important methodologies in Bayes Theory: Bayesian Network and Naïve Bayes Classifier.

3.6.1 Bayesian Network

Bayesian Network (BN) is used to represent knowledge about an uncertain domain and it belongs to the family of probabilistic *graphical models.*

BNs corresponds to a specific graphical model structure known as a *directed acyclic graph.* These graphs enable an effective representation and computation of the joint probability distribution over random variables [61].

The structure of a direct acyclic graph is defined by two sets: the set of

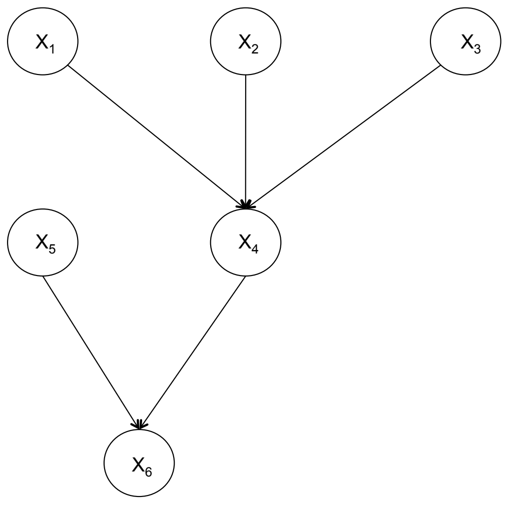


Figure 3.1: This graph represents a directed acyclic graph or Bayesian Network. Each node is a random variable and each arc has his own direction. There are not cyclic on the graph. The node X_6 can be called *"son"* of node X_4 and X_4 can be called *"parent"* of node X_6.

nodes (vertices) and the set of directed edges. The nodes represent random variables and the edges represent direct dependence among the variables. In particular, an edge from node X_i to node X_j represents a probabilistic dependence between the corresponding variables, in other words the facts that X_i happens changes the probability of X_j. Thus, the arrow indicates that a value taken by variable X_i influences the value taken by variable X_j. Node X_i is then referred to as a *parent* of X_j and, similarly, X_j is referred to as a *child* of X_i.

Another important definition is that the node X_j may be said to be a descendant of the node X_i if there is a direct path between X_i and X_j, otherwise X_j is said to be a nondescendant of X_i. We can say that a sequence of nodes $[X_0, ..., X_n]$ is a path if and only if, $\forall\ i$, $1 \leq i \leq n$, there is a direct arc between X_{i-1} and X_i.

The network is defined by a pair $B = \langle G, \Theta \rangle$ where G is the directed acyclic graph whose nodes $X_1, X_2, ..., X_n$ represents random variables, and whose edges represent the direct dependencies between these variables. The graph G encodes independence assumptions, by which each variable X_i is independent of its nondescendents given its parents in G. The second component Θ denotes the set of parameters of the network. This set contains the parameter $\theta_{x_i|\pi_i} = P_B(x_i|\pi_i)$ for each realization x_i of X_i conditioned on π_i, the set of parents of X_i in G. According by B defines a unique joint probability distribution over the set of random variables: namely:

$$P_B(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P_B(X_i|\pi_i) = \prod_{i=1}^{n} \theta_{X_i|\pi_i}$$

Summarizing, a Bayesian Network is an acyclic graph that represents a joint probability distribution over a set of random variables.

Given a Bayesian Network that specifies the joint probability distribution in a factored form, one can evaluate all possible inference queries by marginalization [61].

Two types of inference support are often considered: *predictive support* for node X_i, based on evidence nodes connected to X_i through its parent nodes (also called *top-down reasoning*). With top-down reasoning we intend a reasoning from symptoms to cause. This reasoning occurs in the opposite direction to the network arcs. Instead, *diagnostic support* for node X_i through its children nodes (also called *bottom-up reasoning*). Bottom-up reasoning is reffered to a reasoning from new information about causes to new belief about effects, following the directions of the network arcs [44].

Another inference support is the so called *approximate inference* methods. These methods are often used in the literature, such as *Monte Carlo* sampling that gives gradually improving estimates as sampling proceeds [34]. A variety of standard *Markov Chain Monte Carlo* methods, including the *Gibbs sampling* and the *Metropolis-Hastings algorithm*, are used for approximate inference [60].

Baysian Networks are used in different applications such as machine learning, text mining, bioinformatics and cellular networks. Moreover, Baysian networks can be used, even in the case of missing data, to learn the causal relationships and gain an understanding of the various problem domains and to predict future events.

3.6.2 Naïve Bayes Classifier

The Naïve Bayes Approach [56] is a classification procedure based on Bayes rule, that assumes the attributes $X_1, ..., X_n$ are all conditionally independent of one another, given Y. The value of this assumption is that it dramatically simplifies the representation of $P(X|Y)$, and the problem of estimating it from the data in order to estimate our model.

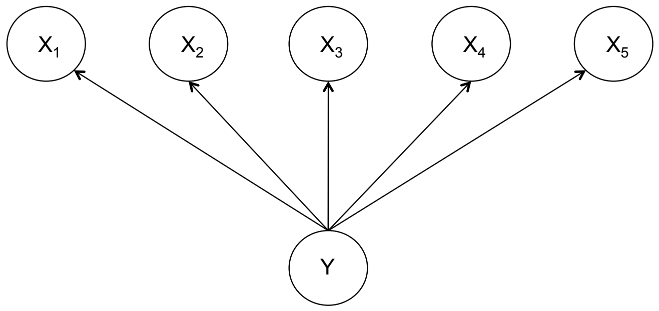


Figure 3.2: This graph represents a graph based on the Naïve Bayes approach. Each node is conditionally independent of one another, given Y.

Consider the case where $X = \langle X_1, ..., X_n \rangle$, we have

$$P(X_1, X_2, ..., X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

This equation follows directly from the definition of conditional independence. Starting from this point, it is possible to understand how the Naïve Bayes Approach works. Assuming that Y is any discrete valued variables, and the attributes $X_1, ..., X_n$ are any discrete or real valued variables, the goal of Naïve Bayes method is to train a classifier that will output the probability distribution over possible values of Y, for each new instance X that we want to classify.

The probability that Y will take on its k th possible value, according to the Bayes Rule, is

$$P(Y = y_k|X_1, ...X_n) = \frac{P(Y = y_k)P(X_1, ..., X_n|Y = y_k)}{\sum_j P(Y = y_j)P(X_1, ..., X_n|Y = y_j)}$$

where the sum is taken over all possible value y_j of Y. Assuming that X_i

are conditionally independent given Y, we can write

$$P(Y = y_k|X_1, ...X_n) = \frac{P(Y = y_k) \prod_i P(X_1, ..., X_n|Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i|Y = y_j)} \tag{3.14}$$

Equation (3.14) is the base for the Naïve Bayes Classifier. Given a new instance $X^{new} = \langle X_1, ..., X_n \rangle$, it is possible to calculate the probability that Y will take on any given value. If we want to know the most probable value of Y, we obtain the Naïve Bayes Rule:

$$Y \leftarrow arg \max_{y_k} \frac{P(Y = y_k) \prod_i P(X_1, ..., X_n|Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i|Y = y_j)}$$

In the case of discrete values, in the Naïve Bayes Approach it is necessary to estimate two sets of parameters. So, when the n input attributes X_i each take on J possible discrete values, and Y is a discrete variable taking on K possible values, the first set of parameters to be estimate are

$$\theta_{ijk} \equiv P(X_i = x_{ij}|Y = y_k)$$

for each X_i, each of its possible values x_{ij} and each possible value y_k of Y. Note there will be nJK such parameters, and also that only $n(J-1)K$ of these are independent, given that they must satisfy $\sum_i \theta_{ijk} = 1$ for each pair (i, k).

In addition, it is necessary to estimate parameters that define the prior probability over Y

$$\pi_k \equiv P(Y = y_k).$$

Here, there are K parameters, $(K-1)$ of which are independent.

To estimate these parameters, it is possible to use either maximum likelihood estimates or using maximum *a posteriori* probability (MAP) estimates, augmenting the observed data with a prior distributions over the values of the parameters.

This approach reduces the complexity for learning Bayesian classifier by making a conditional independence assumption that dramatically reduces the number of parameters to be estimated when modeling $P(X|Y)$.

It is often used in the biological field. Kohonen at al. [43] proposed a Naïve Bayes Classifier for protein function prediction. In this case, the Naïve Bayes approach is used as a tool for annotating proteins on the basis of amino-acids motifs, cellular localization and protein-protein interactions. The authors applied the Naïve Bayes model in order to provide probabilistic predictions, and to enable a computationally efficient approach to data integration.

Chapter 4

Combinatorial Optimization and Metaheuristics

4.1 Combinatorial Optimization

When we are speaking about an optimization problem, theoretical as well as practical, often we refer to the need of finding the best configuration of a set of variables to achieve a specific goal. Optimization problems can be divided mainly into two different categories: those where we consider real-valued variables and those where we consider discrete variables. Optimization problems can concern also a combination between the two types but those are less frequent.

Considering only discrete variables, we find a class of problems called *Combinatorial Optimization* (*CO*) problems. CO problems refer to problems where the solution is an object from a finite, or possibly countable infinite, set. Typically, this object is an integer number, a subset, a permutation of discrete elements or a graph structure [8].

Definition 4.1 *A* Combinatorial Optimization *problem $P = (S, f)$ can be defined by*

- *a set of variables $X = \{x_1, ..., x_n\}$;*

- *variable domains $D_1, ..., D_n$;*

- *constraints among variables;*

- *an* objective function f *to be minimized (or maximized), where* $f : D_1 \times ... \times D_n \to \mathbb{R}^+$*;*

The set of all possible feasible assignments is

$$S = \{s = \{(x_1, v_1), ..., (x_n, v_n)\} | v_i \in D_i, \text{s satisfies all the constraints}\}.$$

where v_i indicates the value of the variable x_i, $\forall i$. S can be called also *search space* and each element in the search space is considered as a candidate solution. A solution $s^* \in S$ is one that minimizes the objective function, it means that $f(s^*) \leq f(s) \forall s \in S$. s^* is a global optimal solution of the problem $P = (S, f)$ and the set $S^* \subseteq S$ is the set of globally optimal solutions.

Traveling Salesman Problem (TSP) [49], Quadratic Assignment Problem (QAP) [31], Timetabling and Scheduling Problems are typical Combinatorial Optimization problem. CO problems arise also from biology and biochemistry. All of them are really important from a practical point of view therefore many algorithms have been developed. These algorithms can be classified in:

- *complete* algorithms;

- *approximate* algorithms;

The first category identifies algorithms that are guaranteed to find for every finite size instance of a problem an optimal solution in bounded time. The second one sacrifices the guarantee of finding optimal solutions for the sake of getting good solutions. The main advantage of the approximate methods is to significantly reduce the amount of time for finding a solution.

Another classification within the approximate algorithms is to consider them separated in: *constructive* methods and *local search* methods. The constructive approach creates solutions from scratch by adding components, to an initially empty partial solution set, until a solution is complete. They are fast but they often return solutions of inferior quality when compared to local search algorithms. The latter start from some initial solution and iteratively tries to replace the current solution by a better one in an appropriately defined neighborhood of the current solution [8]. We can say that a neighborhood of a point is a set containing the point where you can move that point some amount without leaving the set. More formally, a neighborhood is defined as follows:

Definition 4.2 *A* neighborhood structure *is a function $N : S \to 2^S$ that assigns to every $s \in S$ a set of neighbors $N(s) \subseteq S$. $N(s)$ is called a neighborhood of s.*

We intend a neighbor as a modification of the starting point to another one that, for some specific distances, is considered near.

Now it is possible to define what a locally minimal solution is.

Definition 4.3 *A* locally minimal solution *with respect to a neighborhood structure N is a solution $\hat{s}$ such that $\forall s \in N(\hat{s}) : f(\hat{s}) \leq f(s)$. $\hat{s}$ is called a strict locally minimal solution if $f(\hat{s}) < f(s)$ $\forall s \in N(\hat{s})$ and $s \neq \hat{s}$.*

One of the first approaches to solving combinatorial optimization problems was the Branch and Bound (BB) algorithm [48] that is an algorithm requiring a systematic enumeration of all candidate solutions, where a large number of candidate solutions are discarded by using upper and lower estimated bounds of the quantity being optimized. BB algorithm obtains high quality results but it requires effort that grows exponentially with problem size. In the last 27 years a new class of more efficient algorithms has emerged, the so called *metaheuristics*.

4.2 Metaheuristic Algorithms

Metaheuristic methods are a new set of algorithms based on the combination of heuristic approaches in high level frameworks aimed at efficiently and effectively exploring the search space. Metaheuristic algorithms are widely used in different fields with good results.

This set of algorithms includes, but is not restricted to, Ant Colony Optimization (ACO) [26], Evolutionary Computation (EC) [2] including Genetic algorithms (*GA*) [33], Iterated Local Search (*ILS*) [50], Simulated Annealing (*SA*) [41], and Tabu Search (*TS*) [32].

There are different definitions for the term metaheuristic but all the definitions share common features that can be summarized as follows. Metaheuristic algorithms are strategies that guide the search process; the final goal of each approach is to efficiently explore the search space in order to find optimal solutions or, at least, near-optimal solutions. Metaheuristics are composed with different techniques that can range from simple local search to complex learning processes. This kind of algorithms rely on probabilistic decisions made during the search, moreover these approaches include mechanisms to avoid getting stuck in confined areas of the search space. Metaheuristics are a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems, domain-specific knowledge can be included in the process and can help to reach the optimum solution.

In short, metaheuristics are high level strategies for exploring search spaces by using different methods. These methods are a balance between *diversification* and *intensification*.

Diversification means the exploration of the search space, instead the term intensification refers to the exploitation of the accumulated search experience. The use of diversification and intensification is really important in the search process because it allows one to quickly identify regions in the search space with high quality solutions and, also, not to waste too much time in some regions of the search space which either have already been explored or do not provide high quality solutions.

Different metaheuristics apply search strategies that depend on the philosophy of the metaheuristic itself.

4.3 Different Metaheuristic Strategies

It is possible to classify metaheuristic algorithms depending on the characteristics selected to differentiate among them. The most important ways of classifying metaheuristic is [8]:

- *Nature-inspired* or *Non-nature Inspired*;
- *Population-based* or *Single-point Search*;
- *Dynamic* or *Static Objective Function*;
- *One* or *Various Neighborhood Structures*;
- *Memory Usage* or *Memory-less Methods.*

Nature-inspired and *Non-nature Inspired.* This classification is based on the origin of the algorithm. When the search process of the algorithm is *bio-inspired* and, for example, it imitates elements form the social behavior of some physical species it can be classified into the Nature-inspired algorithms. It this class we can individuate algorithms such as Particle Swarm Optimization [40] and Ant Colony Algorithm. The Non-nature inspired algorithms can be, for instance, Iterated Local Search or Tabu Search.

Population-based and *Single-point Search.* When algorithms share the property of describing a trajectory in the search space during the search process (*trajectory methods*) and the trajectories work on single solutions, the algorithms are called Single-point searches. Single-point search techniques encompass local search-based metaheuristic such as Variable Neighborhood Search (*VNS*) [57] or Iterated Local Search. Population-based algorithms perform search processes which describe the evolution of a set of points in the search space.

Dynamic and *Static Objective Function.* Some algorithms keep the objective function given in the problem representation constant while others modify it during the search. The modification of the objective function helps to escape from local minima because the search landscape is modified by trying to incorporate information collected during the search process.

One and *Various Neighborhood Structures.* When metaheuristcs use a set of neighborhood structures to have the possibility to diversify the search by swapping between different landscapes, they are classified as Various Neighborhood Structure Techniques. An example is the Variable Neighborhood Search. Other algorithms do not change the fitness landscape topology in the course of the algorithm.

Memory Usage and *Memory-less Methods*. Memory-less Algorithms perform a Markov process, as the information they exclusively use to determine the next action is the current state of the search process. In the case of Memory Usage Techniques we have to distinguish between the use of short term and long term memory. The first usually stores only the recently performed moves, visited solution or decision taken. The second is usually an accumulation of synthetic parameters about the search.

In the next part of the chapter we will focus our attention on the definition of Population-based and Single-point Search.

4.4 Single-point Search Techniques

The class of Single-point Search Techniques are characterized by a *trajectory method*, it means the search phase is based on trajectory in the search space. In other words, the search process guided by a trajectory method can be seen as the evolution in time of a discrete dynamical system [6]. Starting from an initial solution the algorithm describes a trajectory in the search space. In the simplest case, a trajectory can be composed of two parts: a *transient* phase followed by an *attractor*. An attractor can be a fixed point, a cycle or a complex attractor.

The trajectory is an important feature also because it gives information about the behaviour of the algorithm and its ability to tackle the problem under study.

A classical example of Single-point Search Method is the Simulated Annealing [41] [13].

4.4.1 Simulated Annealing

Simulated Annealing (*SA*) is one of the oldest metaheuristics and it has a strategy to escape from local minima. SA was first introduced as a Combinatorial Optimization tool by Kirkpatrick et al. [41] and Cerny [13].

This algorithm is inspired by the annealing process of metals and glass, which assumes a low energy configuration when cooled with an appropriate cooling schedule. The basic idea behind SA is to modify the local search in order to accept, in probability, worsening solutions. The general framework of SA is the following:

Algorithm 4.4.1: SA PROCEDURE(*Search Space N, function f*)

```
procedure SA ALGORITHM($N, f$)
  $Generate\_an\_initial\_solution:\ S$
  $Initialize\_the\_parameter:\ T$
  repeat
    $Generate: S',\ S' \in N(S)$
    if $f(S') < f(S)$
      then $S \leftarrow S'$
      else $accept\_that:\ S \leftarrow S',\ with\_probability:\ P = e^{-\frac{f(S')-f(S))}{T}}$
    $Update:\ T$
  until (end condition)
  return $(\hat{S})$
```

Generally, the algorithm starts from an initial solution and by initializing the so called temperature parameter T. It is important to underline that T decreases during the search phase, thus at the beginning of the search the probability of accepting uphill moves is high and it gradually decreases. We can say that in the first steps the algorithm is doing an exploration of the search space and, when the "temperature" T starts to decrease, the algorithm concentrates its effort to converge to a (local) minimum.

In other words, SA, with respect to local search, only accepts partial neighborhood exploration and implements the intensification/diversification strategy by means of the annealing (decrease) of parameter T [64].

SA has been applied to several Combinatorial Optimization problems, such as Quadratic Assignment Problem (QAP) and the Job Shop Scheduling (JSS) [30]. Nowadays it is used as a component in metaheuristics, rather than applied as stand-alone search algorithm.

4.5 Population-based Search Techniques

When an algorithm considers a set (*i.e.* a population) of solutions rather than a single solution it is called a Population-based approach. These algorithms provide a natural, intrinsic way for the exploration of the search space while dealing with a population of solutions. Moreover, the final performance depends strongly on the way the population is manipulated. In Combinatorial Optimization the most studied population-based methods are Evolutionary Computation (*EC*) and Ant Colony Optimization (*ACO*).

In the Evolutionary Computation approach, a population of potential solutions (*i.e.* individuals) is modified by *recombination* and *mutation* operators. *Recombination* or crossover operator recombines two or more solutions to produce new possible solutions. A *Mutation* or *modification* operator causes a change in a solution obtaining a new one.

In ACO a colony of artificial *ants* is used to construct solutions guided by the *pheromone* trails and heuristic information.

We shall now describe this algorithm in more details.

4.5.1 Ant Colony Optimization (*ACO*)

Ant Colony Optimization is a metaheuristic approach proposed by Dorigo [21] [25] [22]. In the course of this section, we keep close to the description as given in [23].

Ant algorithms are inspired by the observation of real ant colonies. Social insects, as ants, live in colonies and their behaviour is directed more to the survival of the colony as a whole than to that of a single individual component of the colony. Social insects are really interesting for the high structuration level that their colonies can achieve. In the case of ant colonies it is the foraging behaviour, and, in particular, the way in which ants can find the shortest paths between food sources and their nest.

When ants are walking from the food sources to the end and vice versa,

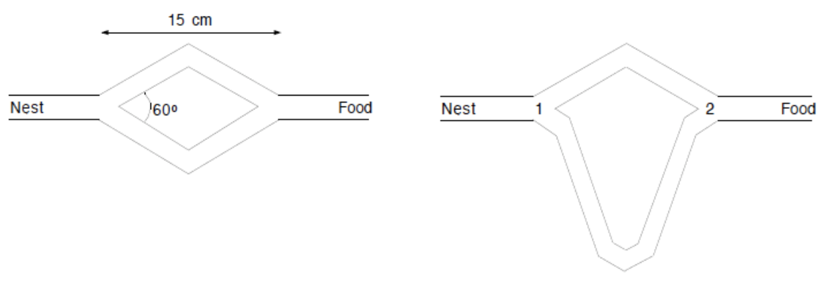


Figure 4.1: Experimental setup for the double bridge experiment. (a) Branches have equal length. (b) Branches have different length.

they deposit on the ground a special substance called *pheromone*, in such a way to form a pheromone trail. Each ant can smell this substance and, when choosing its way, the ant chooses, in probability, paths marked by a strong pheromone concentrations.

The pheromone trail allows the ants to find their way to come back to the food source or to the nest.

In other words, when more paths are available from the nest to a food source, a colony of ants may be able to exploit the pheromone trails left by the individual ants to find the shortest path from the nest to the source and vice versa. This behaviour is possible to demonstrate experimentally.

One important study was designed and run by Deneubourg et al. [20]. This experiment used a double bridge connecting a nest of ants of the Argentine ant species *I. humilis* and a food source. The experiments were run varying the length of the two branches of the double bridge. More precisely, in the first experiment the bridge had two branches with the same length (see Fig. 4.1a).

At the start, the ant were left free to walk in the double bridge from the nest to the source, and the amount of ants walking in the two bridges were observed. The final result was that, although in the initial phase random choices occurred, eventually all the ants used the same branch [26]. This is because in the initial part of the experiments no pheromone on the two branches was present. Hence, the ants did not have any preference with respect to which branch to choose and they selected the branches with the same probability. Yet, because of random fluctuations, a few more ants will select one branch over the other. While walking, ants deposit pheromone, so a larger amount of ants in one branch results in a larger amount of pheromone on that branch; this quantity of pheromone stimulates ants to choose that branch again, and so on until finally the ants converge to one single path. This result is an example of self-organizing behaviour of the ants.

In the second experiment (see Fig. 4.1b), one branch was double the other. In this case, after some time all the ants chose to use only the short branch. When ants start to move in the double brindge, they choose the path randomly because the two branches appear identical to them ants. Now, because one branch is shorter than the other, the ants choosing the short branch are the first to reach the food and to start their return to the nest. At this point, when they choose between the short and the long branches, the higher level of pheromone on the short branch will bias they decision. Therefore, pheromone starts to accumulate faster on the short branch, which will be used by all the ants.

It is interesting to note that a single ant gives only a very small contribution but it is the ensemble of ants which presents the *shortest path finding* behaviour [23]. Another important point is that ants perform this specific behaviour using a simple form of indirect communication mediated by pheromone lying, known as *stigmergy*.

The model inspired by ants foraging behaviour is and interesting model for artificial multi agent systems applied to the solution of difficult optimization problems.

Differences between ACO and real ants' behaviour

In Ant Colony Optimization (ACO) algorithm a colony of artificial ants (*agents*) cooperate in finding a solutions to difficult discrete optimization problem. Artificial ants are an abstraction of real ants and, on the other hand, they have been enriched with some capabilities which do not find a natural counterpart.

The first similarity is that, as real ant colonies, ant algorithms are composed of a populations, or colonies, of concurrent and asynchronous entities globally cooperating to find a good solution for the problem under study. As in the reality, artificial ants modify some aspects of their environment. While real ants deposit pheromone while they are walking, artificial ants change some numeric information locally stored in the problem's state they visit. This information is the ant's current history/performance and can be read/written by any ant. This numeric information can be called *artificial pheromone trail.* ACO algorithm uses another real aspect of real ant colonies, more specifically of the real pheromone. Real pheromone evaporates over time, in ACO an evaporation mechanism is implemented. The pheromone evaporation allows the ant colony to slowly forget its past history so that it can direct its search towards new directions.

Artificial and real ants share a common task: to find a shortest path joining the nest to the destination food sites. In the algorithm it is the ability to find the path with the minimum cost from an origin to an end state. This is possible because artificial ants, as real ants, build a solution applying a probabilistic way to move through adjacent states. The policy by which the ants choose the direction is a function of both the *a priori* information represented by the problem specification and of the local modifications in the environment induced by past ants.

ACO metaheuristc has also some characteristics of its own. Artificial ants live in a discrete world and their moves consist of transitions from discrete states to discrete states, moreover they have an internal state. This state contains the memory of the past actions. In the algorithm the pheromone deposited by the ant is a function of the quality of the solution found. Another difference between artificial ants and real ants is that artificial ants' timing in pheromone laying is problem dependent, and often does not reflect ants' behaviour.

The last difference is that ACO algorithms, to improve the overall system, can be enriched with extra capabilities like local optimization, backtracking and so on, that cannot be found in real ants.

ACO framework

In ACO algorithms each "ant" builds a solution starting from an initial state selected according to some problem dependent criteria. A solution is expressed as minimum cost (shortest) path through the states of the problem in accordance with the problem's constraints. A single ant is able to build a solution but only the cooperation among all the agents of the colony concurrently building different solutions is able to find high quality solutions.

Each ant builds a solution by moving through a sequence of states. Moves are selected by applying a stochastic local search policy directed by:

- Ant private information: the ant internal state or memory. The ant's internal state stores information about the ant past history which can be used to carry useful information on the solutions or part of them;

- Pheromone trails accumulated by all the ants from the beginning of the search process and *a priori* (heuristic) problem-specific information.

The combination of available pheromone and heuristic values defines *ant-decision tables*, that is, probabilistic tables used by the ants's decision policy to direct their search towards the most interesting regions of the search space. An important point to underline is that the stochastic component of the move choice decision policy and the pheromone evaporation avoid early stagnation of all the ants in a part of the search space.

In this section, a high level description of the ACO metaheuristic is reported in pseudocode, divided in three parts. This pseudocode is taken from

[23].

Algorithm 4.5.1: ACO PSEUDOCODE - PART 1()

```
procedure ACO METAHEURIST()
  while termination_criterion_not_satisfied
    do
      schedule_activities
        ants_generation_and_activity();
        pheromon_evaporation();
        daemon_actions();
      end schedule_activities
  end while
end procedure
```

Algorithm 4.5.2: ACO PSEUDOCODE - PART 2()

```
procedure ANTS GENERATION AND ACTIVITY()
  while available_resources
    do
      schedule_the_creation_of_a_new_ant();
      new_active_ant();
  end while
end procedure
```

Algorithm 4.5.3: ACO PSEUDOCODE - PART 3()

```
procedure NEW ACTIVE ANT()
  comment: ant lifecycle
  $initialize\_ant()$;
  $M = update\_ant\_memory()$;
  while $current\_state \neq target\_state$
    do
      $A = read\_local\_ant\_routing\_table()$;
      $P = compute\_transition\_probabilities(A, M, problem\_constraints)$;
      $next\_state = apply\_ant\_decision\_policy(P, problem\_constraints)$;
      $move\_to\_the\_next\_state(next\_state)$;
      if $online\_step\_by\_step\_pheromone\_update$
        then
          $deposit\_pheromone\_on\_the\_visited\_arc()$;
          $update\_antrouting\_table()$;
      end if
  end while
  if $online\_delayed\_pheromone\_update$
    then
      $evaluate\_solution()$;
      $deposite\_pheromone\_on\_all\_visited_arcs()$;
      $updating\_ant\_routing\_table()$;
  end if
  $die()$
  end procedure
```

The *daemon_action()* refers to actions such as local optimization procedures. It is optional and it is suggested in the case of missing heuristic information.

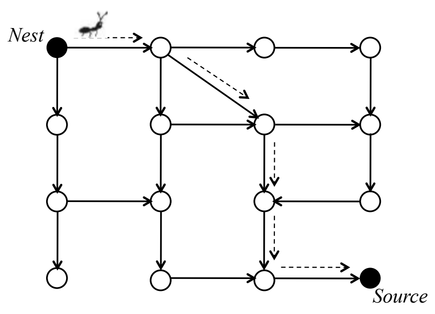


Figure 4.2: Representation of a possible graph were ants move, from the nest to the source.

Ant System (*AS*) and the Traveling Salesman Problem (*TSP*)

To understand better how ACO algorithm works, in this section we will present one of the most important application of ACO: the *travelling salesman problem* (TSP) [49].

The following is a general definition of the TSP. Consider a set N of nodes, representing cities, and a set E of arcs fully connecting the nodes N. Let d_{ij} be the length of the arc $(i, j) \in E$, that is the distance between cities i and j, with $i, j \in N$. The TSP is the problem of finding a minimal length Hamiltonian circuit on the graph $G = (N, E)$, where an Hamiltonian circuit of graph G is a closed tour visiting once and only once all the $n = |N|$ nodes G, and its length is given by the sum of the lengths of all the arcs of which it is composed.

How ACO algorithm works in the TSP problem is explained through the *Ant System* (AS) [24], the first version of this approach. In this case artificial ants build solutions, which in the TSP are tours, by moving on the problem graph from one city to another. The maximum number of iteration that the algorithm is allowed to do is t_{max}. During each iteration m ants build a tour executing n steps in which a probabilistic rule is applied, as we shall show below. In other words, when in node i the ant chooses the node j to move to, and the arc (i, j) is added to the tour under construction. The algorithm is repeated until the ant has completed its tour.

In the AS algorithm, the pheromone can be deposited in different ways either while building a solution or after the ants have built a complete tour.

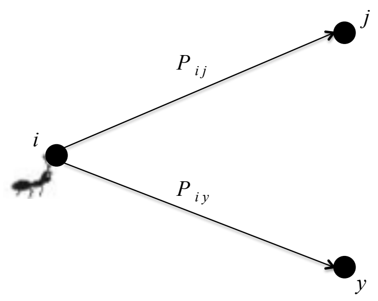


Figure 4.3: Ant, in the node i, chooses the next node (j or y) in according with the probability in the arcs, such as P_{ij} and P_{iy}

Our description is concentrated on the second way to update the pheromone.

After ants have built their tours, each ant deposits pheromone on the pheromone trail variables associated to the visited arcs to make the visited arcs become more desirable for future ants. The pheromone trail $\tau_{ij}(t)$ associated to arc (i, j) represents the desirability of choosing city j when in city i. The quantity of pheromone deposited in the arcs is proportional to the quality of the solutions, this choice helps to direct the search towards good solutions.

Each ant has a memory of all the visited cities and is called *tabu list.* The memory is used to define, for each ant k, the set of cities that an ant located on city i still has to visit.

The ant-decision table $A_i = [a^i_j(t)]_{|N_i|}$ of node i is obtained by the composition of the pheromone trail values with heuristic values as follow:

$$a_{ij} = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i} [\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta} \qquad \forall j \in N_i$$

where $\tau_{ij}(t)$ is the amount of pheromone trail on arc (i, j) at time t, $\eta_{i,j} = 1/d_{ij}$ is the heuristic value of moving from node i to node j, N_i is the set of neighbors of node i, and α and β are two parameters that control the relative weight of pheromone trail and heuristic information.

The probability with which an ant k chooses to go from city i to city $j \in N^k_i$ while building its tour at the t-th algorithm iteration is:

$$p_{ij}^k(t) = \frac{a_{ij}(t)}{\sum_{i \in N_i^k} a_{ij}(t)}$$

where $N_i^k \subseteq N_i$ is the set of nodes in the neighborhood of node i that ant k has not visited yet.

After all ants have completed their tour, pheromone evaporation on all arcs is applied, and then each ant k deposits a quantity of pheromone $\triangle\tau_{ij}^k(t)$ on each arc that it has used:

$$\triangle\tau_{ij}^k(t) = \begin{cases} 1/L^k(t) & \text{if (i, j)} \in T^K(t) \\ 0 & \text{if (i, j)} \notin T^K(t) \end{cases}$$

where $T^k(t)$ is the tour done by ant k at iteration t, and $L^k(t)$ is its length. Obviously $\triangle\tau_{ij}^k(t)$ depends on how well the ant has performed: the shorter the tour done, the greater the amount of pheromone deposited.

In pratice, the pheromone is updated in the following way:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \triangle\tau_{ij}(t)$$

where $\triangle\tau_{ij}(t) = \sum_{k=1}^{m} \triangle\tau_{ij}^k(t)$, m is the number of ants at each iteration and $\rho \in (0, 1]$ is the pheromone trail decay coefficient.

AS algorithm demonstrates really good solutions in term of quality and convergence and it is the first ACO algorithm and the base for many improvement of this method.

$\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS)

Ant System showed to be an efficient method to tackle hard combinatorial optimization problems but it was rather poor in the presence of high number of cities in the TSP (or variables). Researches on ACO have demostrated that a strong exploitation of the best solutions found during the search can help to improve the performance of the algorithm. Another important point to achieve better performance is to combine the previous approach with a mechanism to avoid premature stagnation of the search.

In 2000, Stützle and Hoos [68] have presented the $\mathcal{MAX}-\mathcal{MIN}$ Ant

System ($\mathcal{MM}$AS), an Ant Colony Optimization algorithm derived from Ant System.

They have demostrated that $\mathcal{MM}$AS is able to reach a strong exploitation of the search space by adding pheromone only to the best solution during the pheromone trail update. Moreover they applied a simple method for limiting the strenghts of the pheromone trails that effectively avoids premature convergence of the search. In what follows, the most important features of $\mathcal{MM}$AS are studied.

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System differs with respect to the AS in three key aspects:

(i) only a single ant deposits pheromone after each iteration. It can be the ant that reached the best solution in the current iteration (*interation-best* ant) or the one which found the best solution from the beginning of the trial (*global-best* ant);

(ii) the range of possible pheromone trails on each solution component is limited to an interval $[\tau_{min}, \tau_{max}]$;

(iii) the pheromone trails is deliberately initialized to τ_{max}.

Point (i) helps to exploit the best solutions found during an iteration or during the run of the algorithm. Then, only a single ant is used to update the pheromone trails after each iteration. Now, the pheromone trail update rule is:

$$\tau_{ij}(t+1) \leftarrow (1-\rho)\tau_{ij}(t) + \triangle\tau_{ij}^{best}$$

where $\triangle\tau_{ij}^{best} = 1/f(s^{best})$ and $f(s^{best})$ is the solution cost of either the iteration-best (s^{ib}) or the global-best solution (s^{gb}). Using s^{gb}, the search may concentrate too fast on this solution limiting the founding of other solutions. This situation can be avoided applying the s^{ib} since the iteration-best solutions may differ iteration to iteration allowing to reinforce a larger number of solution components. It is possible to use a mixed approach, for example using s^{ib} as a standard approach for updating the pheromone and using s^{gb} onlyevery fixed number of iterations.

Using these ways to update the pheromone trail, solutions elements which frequently occur in the best found solutions get a large reinforcement.

The second point restricts the range of possible values for the probability of choosing a specific arc, this helps to avoid early stagnation. For the same reason, in the $\mathcal{MAX} - \mathcal{MIN}$ Ant System, the pheromone trails are updated using a proportional mechanism: $\triangle\tau_{ij}(t) \propto (\tau_{max} - \tau_{(i,j)}(t))$. This mechanism is called *trail smoothing mechanism* and is useful when some pheromone trails are close to τ_{max} while most of the others are close to τ_{min}. It is shown by [68] that by limiting the influence of the pheromone trails it is possible to avoid the relative differences between the pheromone trails from becoming to high during the iterations of the algorithm.

The last point (iii), require that the pheromone trails are initialized to τ_{max}. In other words, after the first iteration all pheromone trails correspond to $\tau_{max}(1)$. This is possible by setting $\tau(0)$ to some arbitrarily high value. After the first iteration, the pheromone will be set to $\tau_{max}(t)$.

This way of initializing the pheromone allows us to increase the exploration of solutions during the first iterations of the algorithm. Moreover the probability to select a solutions evolve more slowly, and hence, the exploration of solutions is favoured.

$\mathcal{MM}$AS achieves a strongly improved performance compared with other versions of ACO's algorithm by exploiting more in deeph the best solutions found during the search, and by directing ants' search to very high quality solutions and by avoiding premature convergence of the ants' search.

This concludes our presentation of the optimization algorithms that we are going to use for the remaining of the thesis.

Chapter 5

Evolutionary Model Based Experimental Design

5.1 Some Initial Considerations

Our problem is characterized by high dimensionality of the search space, due to the very large number of elements to be selected, the number of different ways in which elements can be composed, the different laboratory protocols and the network of potential interactions between elements.

A possibility is to combine approaches from Design of Experiments and metaheuristic algorithms to guide the exploration of the space. Some examples of this approach can be found in the literature.

For example in Koukouvinos et al. [45] a hybrid simulated annealing genetic algorithm (SAGA) is used for generating Optimal Designs. The hybrid SAGA combines features such as the power of the Genetic Algorithm (GA) and the speed of a local optimizer such as Simulating Annealing (SA), merging the previous metaheuristics into a powerful hybrid optimization algorithm. This class of hybrid metaheuristics has enabled the authors to build optimal designs.

In our case, the ultimate aim is to test the possibility of exploiting bio-inspired algorithms combined with statistical techniques to search in a discrete sequence space for a target structure. We wish to define a new approach for optimization based on Evolutionary Model Based Experimental Design that has been proposed by [29] [10] [5]. The Evolutionary Model Based Experimental Design must be able to:

– reach the optimum of unknown functions with few iterations of the algorithm;

– explore a very large part of the search space;

– reduce the number of real experimentations in such a way as to save money and time.

5.2 Model Based Ant Colony Design (MACD)

5.2.1 Basic idea

As already pointed out, in our biological problem the direct experimental evaluation of potential solutions is the only option to know the performance of an enzyme but the experimentation is costly and time consuming.

We propose a method that couples real experimentation with simulated experiments. We base our idea on the concept of closed loop evolution [42] where the solutions are evaluated in the real world by conducting a physical experiments and the creation of new candidate solutions is operated in a simulation setting.

Closed loop optimization deals at the same time with concepts from statistics and metaheuristic algorithms. In general, the purpose of the approach is optimization, rather than global modeling. The convergence of these techniques was proposed for the first time by Box [11] who incorporated replication of experiments and considered how to minimize the effects of nuisance factors. The prevalence in closed loop optimization of batch experiments, where many solutions can be evaluated in parallel, suggests the use of population based approaches (see Chapter 4).

An outline of the closed loop optimization is given in Fig. 5.1. Solutions are generated by an algorithm in computer simulation, but their evaluation is

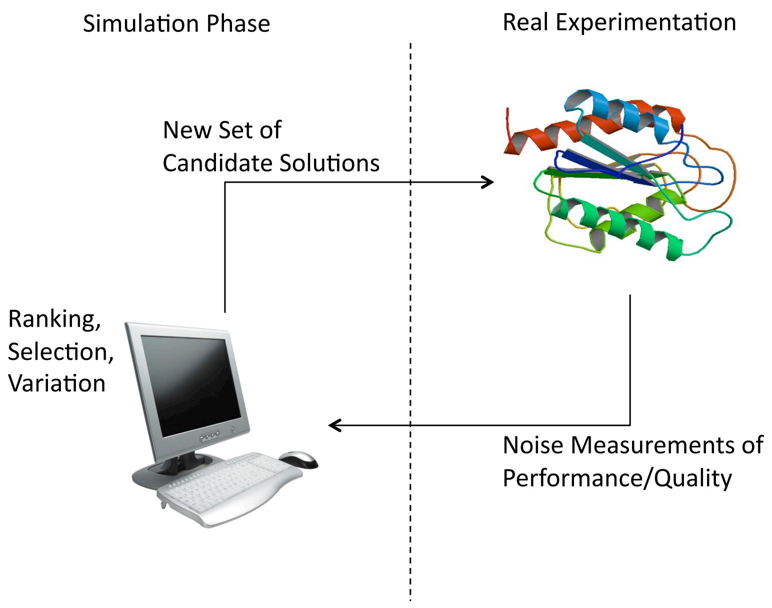

Simulation Phase
Real Experimentation
New Set of Candidate Solutions
Ranking, Selection, Variation
Noise Measurements of Performance/Quality

Figure 5.1: A outiline of closed loop optimization

achieved by physical experiment. Evaluations are fed back to the simulative phase of the approach and its generation of subsequent solutions is a function of these. Thus the process has the form of a closed loop, being at least partially sequential.

Closed loop optimization has different features and limits that can be summarized as follows:

– fitness function design: it is the process that transforms a real system into a focused programme of optimization based on defined measures of quality or performance. A possible solution is to use a simulator as in the theory of *Computer Experiments*. This has led to the use of surrogate models (emulators), *i.e.* simpler models which represent a valid approximation of the original simulator. These emulators are statistical interpolators built from the simulated input-output data. Predictions at untried points, most useful in the case of expensive simulations, are made by the surrogate models [4].

– *evaluations of experiments*: the total amount of evaluations available is often restricted below what is confortable for obtaining optimal or close to optimal solutions. The combination of a predictive model, estimated from a first population of real experimental runs, and an efficient optimization algorithm to search in the space of candidate solutions during the simulative phase is an approach able to explore the search space although in a predictive way.

– *noise, uncertainty and uncontrolled factors*: experimental measurements are often noisy due to factors in the environment that are not involved in the optimization process. In the Design of Experiments, noise is estimated and accounted for by using *replication*, whilst disturbances to estimation caused by nuisance factors are mitigated using blocking, and *randomization*. When dealing with optimization, these concepts are important to be considered in the construction of the solutions.

– *population size* (a set of candidate solutions): in a optimization approach, the population size is usually a free parameter of the algorithm. In closed loop scenarios, the offspring population size may be largely dictated by details of the experimental setup. For example, if chemicals are to be tested in a 96-well plate (a standard piece of laboratory equipment), then 96 may be the maximum population size.

– *constraints*: the real world nature of experimental problems means that finding an appropriate representation and set of constraints can be challenging. It is important to consider a set of constraints that are not overly restrictive and still allow innovative solutions to be found, whilst maintaining feasibility.

Starting from these concepts we develop our methodology, the Model Based Ant Colony Design. The method is described in the following section.

5.2.2 The Approach

Model Based Ant Colony Design (MACD) is based on the idea behind closed loop optimization and the procedure boosts an optimization algorithm by a simulator (strictly speaking an emulator), in our case a statistical model. More precisely, MACD combines:

– $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS, see Chapter 4). $\mathcal{MM}$AS is coupled with a local search. We select the Simulated Annealing (SA) as the local search method because it has demonstrated highest performance in the simulation study (see Section 5.5);

– linear regression model with binary predictive variables, which is estimated from the data by the least squares method. This model does not include interactions between variables.

The following steps summarize our procedure:

1. Randomly generate and evaluate an initial population (size N, in our application $N = 96$. 96 corresponds to the dimension of the well-plate.) of $m - tuples$. With $m - tuples$ we intend a sequence of 4 pseudo-domains forming an enzyme so $m = 4$;

2. Estimate the predictive statistical model based on the population of the available $m - tuples$ or enzymes;

3. Select a new set of N $m - tuples$ (in the application $m = 4$) by the solution construction process implemented in the $\mathcal{MM}$AS. For this purpose, we create a graph where each node represents a specific pseudo-domain. A solution is a path with length 4 composed of 4 nodes connected by arcs. In the biological application, a node corresponds to a pseudo-domain and an arc to the connection between the pseudo-domain i in position k and the pseudo-domain j in position $k + 1$;

4. Identify the best predicted $m - tuple$ and use it to start a local search by the Simulated Annealing. Make a prediction of the response value using the fitted statistical model;

5. If the predictive response value of the new solution is larger than the one selected in Step 4, the new solution replaces the old one in the population obtained at Step 2;

6. The probability matrices are updated (see Chapter 4);

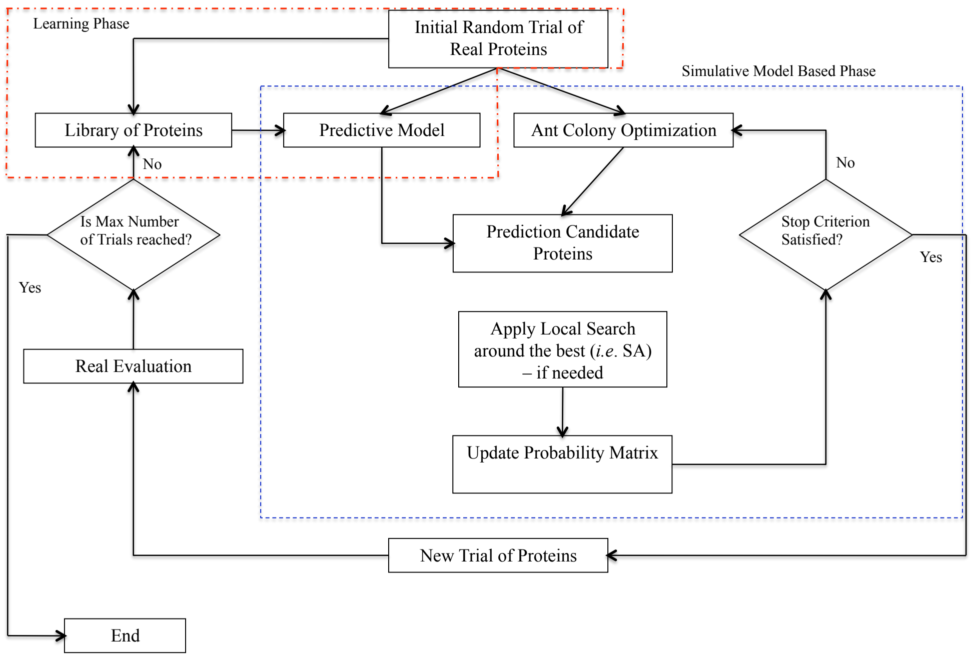

Learning Phase
Initial Random Trial of Real Proteins
Simulative Model Based Phase
Library of Proteins
Predictive Model
Ant Colony Optimization
No
No
Is Max Number of Trials reached?
Prediction Candidate Proteins
Stop Criterion Satisfied?
Yes
Yes
Apply Local Search around the best (i.e. SA) – if needed
Real Evaluation
Update Probability Matrix
New Trial of Proteins
End

Figure 5.2: Model Based Ant Colony Design.

7. Repeat steps from 3 to 6 until the stop criterion is satisfied. In our case, we stop after $T = 100$ iterations. When the stop criterion is satisfied, the last set of $m - tuples$ proposed by the approach is chosen as the new set of candidate solutions to be tested;

8. The new set of candidate solutions is evaluated and included in the set of the $m - tuples$ that have been already evaluated;

9. Repeat the steps from 2 to 8 for a fixed number of experimental generations.

The procedure describe above improves upon existing methods in two main directions:

1. Thanks to the statistical model, we can simulate the problem and move in the search space as many times as we want, hopefully improving the solutions step by step;

2. The iterated refinement of the predictive model provides the optimization algorithm with predictive capability of the model resulting in an increased accuracy during the optimization process.

5.3 Naïve Bayes Ant Colony Optimization (NACO)

5.3.1 Basic idea

In our problem, the experiments are costly and time consuming thus empirical evidence provided by well-designed experiments is crucial to reach a satisfactory result.

In an adaptive experiment, the data gathered from earlier experiment batches is used to improve the next experiments in order to be maximally informative in a properly defined sense. In this framework, sometimes referred to as Adaptive Design Optimization (ADO), the experimenter has some degree of control over what experimental points to investigate.

In this setting, one important point is how to learn the most about the system under study using the least number of trials. This is non trivial when:

– The goal is to learn about a complex system composed by several experimental input variables with patterns of interactions;

– Experiments are costly or time-consuming and each input-output pair provides only little information about the whole system.

In this work, we tackle the ADO problem using a Naïve Bayes Classifier combined with Ant Colony Optimization. Our strategy calculates which elements affect mostly the response of the system for each position and uses this information to help the metaheuristc algorithm to choose the next set of candidate solutions.

The Naïve Bayes Approach (see Chapter 3) has a strong assumption and it assumes that the attributes $X_1, ..., X_n$ are all conditionally independent of one another, given the response Y. It has the advantage to simplify the representation of the probability o X given Y but with the Naïve Bayes Classificator it is not possible to understand the relations between the attributes.

In our problem it could be a strong limitation because we know that domains have significant interactions between them in a protein (see Chapter 1).

The combination of ACO and Naïve Bayes Approach can avoid this problem. ACO is based on probabilistic matrices where the best path has higher probability to be chosen. A path is composed by nodes and arcs. In our problem nodes can be seen as pseudo-domains and an arc connecting a pseudo-domain to the next one can be seen as the relation that exists between the two pseudo-domains. Then, ACO implicity implies the sequential relationship between pseudo-domains. In fact, the response of an enzyme depends on the pseudo-domain and its position. Naïve Bayes Ant Colony Optimization (NACO) improves upon the limits of the individual techniques enabling us to deal with the very large experimental space of the possible solutions.

5.3.2 The Approach

Naïve Bayes Ant Colony Optimization (NACO) is an optimization algorithm based on the combination between Ant Colony Optimization and Naïve Bayes Classifier. NACO extracts the information from the data using the Naïve Bayes Approach and explores the search space by the ACO algorithm. At the same time, the better pseudo-domains are individuated in each position and the interactions between positions are indentified.

The following steps summarize the NACO approach:

1. Random generation and evaluation of an initial population;

2. Individuation of the *Iteration Best Solution*;

3. Calculation of the Naïve Bayes Classifier on the available solutions evaluated ($N = 96$). The Naïve Bayes Classifier is applied on each position of the sequence on desirable values of the response. At each iteration it focuses on values of the response greater than a certain threshold $\gamma \in \mathbb{R}$;

4. Use of the similarity matrix as heuristic information;

5. The probability matrix is updated using the information extracted in points 2, 3 and 4;

6. Selection of the next population of candidate solutions using the principle of Ant Colony Optimization;

7. The new set of candidate solutions is experimentally evaluated and included in the set of solutions that has been already evaluated;

8. If stop criterion is reached, then stop. Otherwise repeat points from 2 to 7;

In Fig. 5.3 it is shown how the similarity matrix is used as heuristic information. For each pseudo-domain a measure of similarity is calculated (see Chapter 1) with respect to all the other $n-1$ pseudo-domains. Following the identification of the *Iteration Best Solution*, the weight of each pseudo-domain is increased proportionally to its similarity with the best pseudo-domain identified for a given position. The rationale behind this is that once a good pseudo-domain is identified for a given position, the algorithm exploits the surrounding experimental space using the similarity matrix.

Fig. 5.4 clarifies point number 5. At iteration t, agents move over the graph according to the best paths identified in the previous steps (Fig. 5.4 (a)). Following candidate solution evaluation, the *Iteration Best Solution* is individuated and the corresponding pheromone path is updated (Fig. 5.4 (c)). At this point, using the Naïve Bayes Classifier the best pseudo-domains for each position are identified, namely those that anticipate to yield a fitness higher than a suitable chosen fitness treshold $\gamma \in \mathbb{R}$ (Fig. 5.4 (b)). For any given arc connecting pseudo-domain i with pseudo-domain j, the weight λ_{ij} is increased according to the Naïve Bayes Classifier. The set of $\{\lambda_{ij}\}$ is called *Naïve Information*. Now, the ant-decision table $A_i = [a^i_j(t)]_{|N_i|}$ of node i will be obtained by the composition of the pheromone trail values with heuristic values and with *Naïve Information* as follows:

$$a_{ij} = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta [\lambda_{ij}]^\delta}{\sum_{l \in N_i} [\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta [\lambda_{ij}]^\delta]} \qquad \forall j \in N_i$$

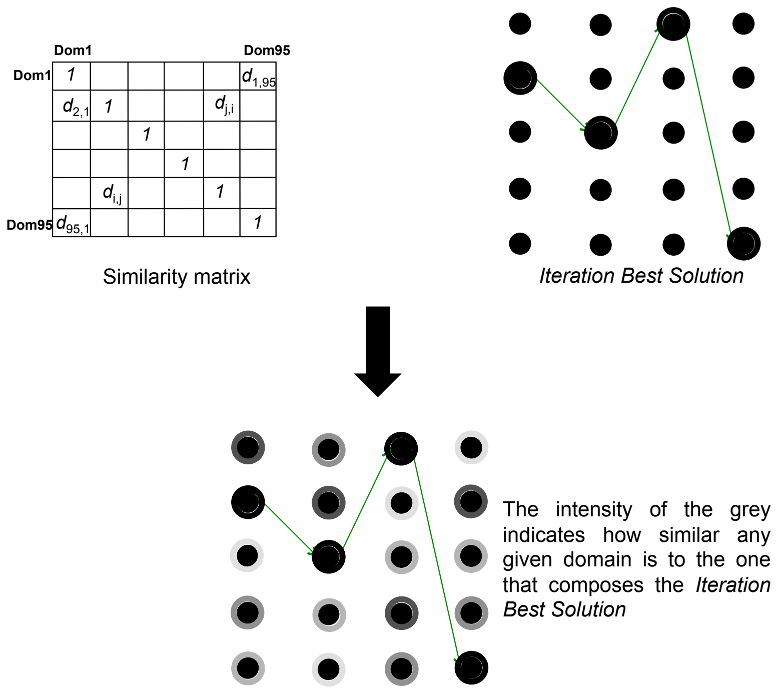

Dom1
Dom95
Dom1
1
$d_{1,95}$
$d_{2,1}$
1
$d_{j,i}$
1
1
$d_{i,j}$
1
Dom95
$d_{95,1}$
1
Similarity matrix
Iteration Best Solution
The intensity of the grey indicates how similar any given domain is to the one that composes the Iteration Best Solution

Figure 5.3: The Similarity Matrix is used as a Heuristic Information.

where $\tau_{ij}(t)$ is the amount of pheromone trail on arc (i, j) at time t, $\eta_{i,j} = 1/d_{ij}$ is the heuristic value of moving from node i to node j, λ_{ij} is the *Naïve Information* on arc (i, j) at time t. N_i is the set of neighbors of node i, and α, β and δ are three parameters, chosen by the experimenter, that control the relative weight of pheromone trail, heuristic information and *Naïve Information* .

The probability with which an ant k chooses to go from domain i to domain $j \in N_i^k$ while building its tour at the t-th algorithm iteration is:

$$p_{ij}^k(t) = \frac{a_{ij}(t)}{\sum_{i \in N_i^k} a_{ij}(t)}$$

where $N_i^k \subseteq N_i$ is the set of nodes in the neighborhood of node i that ant k has not visited yet.

In the context of Enzyme Engineering and Design, NACO will extract information from few data and it will individuate the best connection between elements (*i.e.* pseudo-domains) in a sequence, which is an enzyme.

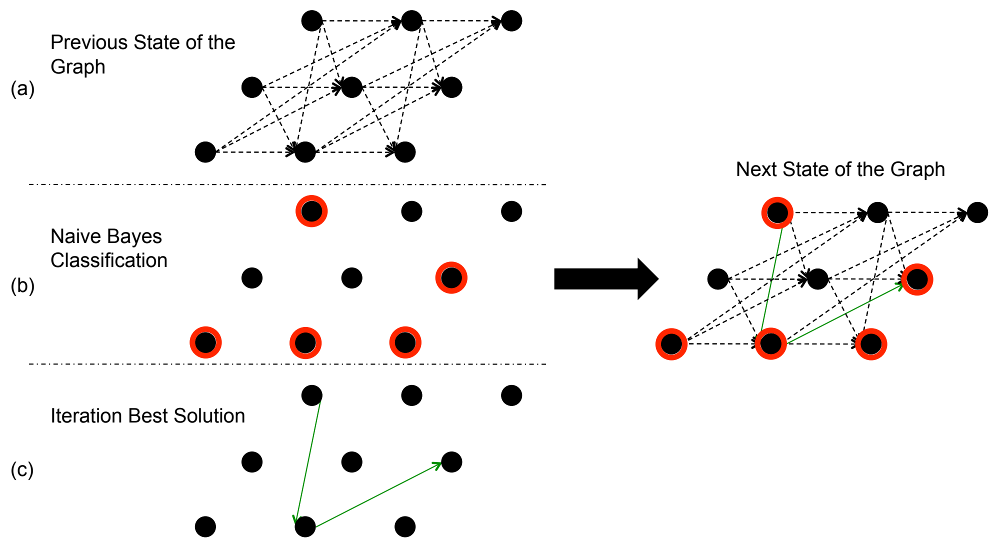


Figure 5.4: Updating Phase of the Naïve Bayes Ant Colony Optimization.

5.4 Methodological Issue

In this section, we describe the development of two Evolutionary Experimental Designs with the ultimate aim to aid the exploration of the enzyme combinatorial sequence space to identify a functional enzyme from a large library. In this perspective, we have designed the search algorithms bearing in mind that candidate enzymes shall be experimentally characterized in terms of expression, solubility, structural features and enzymatic activity. In order to be experimentally tested, candidate proteins shall respect the following biological restrictions already introduced in Chapter 1:

i. The number of cysteine residues shall be no higher than 9 and different from 5 and 7;

ii. The percentage of coil shall not be higher than 70.

These constraints have been implemented in the search algorithm. Step by step, a special function is dedicated to check if a candidate solution respects the biological restrictions and if it can be tested.

As benchmark functions we choose three mathematical models, which are described below.

Polynomial regression model (PRM)

This structure is described by a polynomial regression model with 380 main effects (*i.e.* the effects of one of the i, with $i = 1, \ldots, 95$, pseudo-domains in each j, with $j = 1, \ldots, 4$, different positions) and 18 interactions between pairs of variables and 12 interactions among triplets. The interactions between pairs of variables and triplets are obtained considering the best 3 pseudo-domains for each position and combining them in pairs and triplets. This model represents a enzyme fitness landscape dominated by strong interactions (*i.e.* epistasis), which occurs when the effect of one pseudo-domain depends on the presence of another [7]. This kind of fitness landscapes is characterized by ruggedness and local optima, and may range from "Mt. Fujiyama landscape"(5.5(a)) or "Smooth landscape"(5.5(b)) to highly rugged "Badlands'landscape"(5.5(c)).

The resulting simulative Polynomial regression model closely resembling a "Smooth landscape"is formalized as follows:

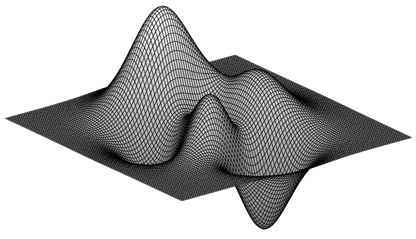
$$\begin{aligned}
y \quad = \quad & \sum_{i=1}^{95}\sum_{j=1}^{4}\beta_{ij}x_{ij} + \alpha_1 x_{2,1}x_{95,2} + \alpha_2 x_{2,1}x_{49,3} + \alpha_3 x_{2,1}x_{95,4} + \alpha_4 x_{95,2}x_{49,3} \\
+ \quad & \alpha_5 x_{95,2}x_{95,4} + \alpha_6 x_{49,3}x_{95,4} + \alpha_7 x_{1,1}x_{93,2} + \alpha_8 x_{1,1}x_{48,3} + \alpha_9 x_{2,1}x_{94,4} \\
+ \quad & \alpha_{10} x_{93,2}x_{48,3} + \alpha_{11} x_{93,2}x_{94,4} + \alpha_{12} x_{48,3}x_{94,4} + \alpha_{13} x_{3,1}x_{94,2} + \alpha_{14} x_{3,1}x_{50,3} \\
+ \quad & \alpha_{15} x_{3,1}x_{1,4} + \alpha_{16} x_{94,2}x_{50,3} + \alpha_{17} x_{94,2}x_{1,4} + \alpha_{18} x_{50,3}x_{1,4} + \delta_1 x_{2,1}x_{95,2}x_{49,3} \\
+ \quad & \delta_2 x_{2,1}x_{95,2}x_{95,4} + \delta_3 x_{2,1}x_{49,3}x_{95,4} + \delta_4 x_{95,2}x_{49,3}x_{95,4} + \delta_5 x_{1,1}x_{93,2}x_{48,3} \\
+ \quad & \delta_6 x_{1,1}x_{93,2}x_{94,4} + \delta_7 x_{93,2}x_{48,3}x_{94,4} + \delta_8 x_{1,1}x_{48,3}x_{94,4} + \delta_9 x_{3,1}x_{94,2}x_{50,3} \\
+ \quad & \delta_{10} x_{3,1}x_{94,2}x_{1,4} + \delta_{11} x_{94,2}x_{50,3}x_{1,4} + \delta_{12} x_{3,1}x_{50,3}x_{1,4} \qquad (5.1)
\end{aligned}$$

where the coefficients β_{ij} (where $i = 1, \ldots, 95$ and $j = 1, \ldots, 4$) are:

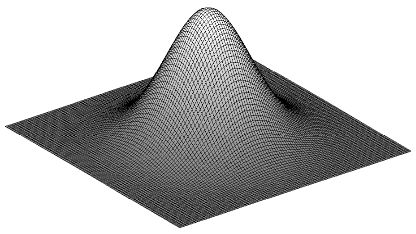
β_{i1} 95 real numbers equally spaced between $-30, \ldots, 30$;

β_{i2} 95 real numbers equally spaced between $-20, \ldots, 20$;

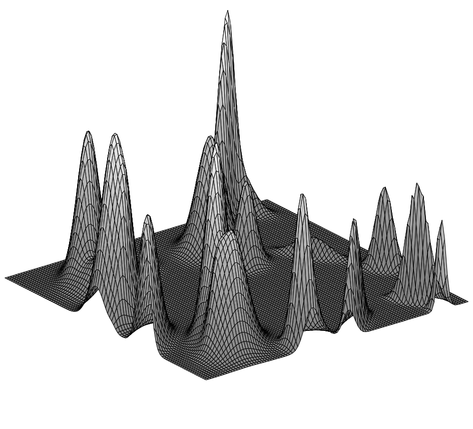
β_{i3} 95 real values obtained from the evaluation of a parabolic function $-10z^2 + z + 30$ with z in $-10, \ldots, 10$;



(a)



(b)



(c)

Figure 5.5: Enzyme fitness landscape type.

β_{i4} 95 real values obtained from the evaluation of a parabolic function $10z^2 + z - 30$ with z in $-10, \dots, 10$;

α, δ coefficients of interactions among pairs and triplets of the three best x_i for each j.

Furthermore each x_{ij} represents a specific pseudo-domain i in the position j. x_{ij} is equal to 1 if the pseudo-domain is in the considered sequence otherwise it is 0. The optimal solution is $(1, 95, 49, 95)$ with the response value equal to 184.961.

Polynomial sparse regression model (PSRM)

The second formal structure to generate data represents the situation where some elements for each position j, in the enzyme sequence, highly influence the response of the system and the others are close to 0. This model closely represents an experimental enzyme fitness landscape where the majority part of enzyme sequences do not possess any function (zero fitness) whereas rare functional enzymes are tightly clustered together [1]. The values of the coefficients determine the shape of the landscape.

The resulting simulative model is:

$$\begin{aligned}
y &= \sum_{i=1}^{95}\sum_{j=1}^{4}\beta_{ij}x_{ij} + \alpha_1 x_{54,1}x_{7,2} + \alpha_2 x_{54,1}x_{63,3} + \alpha_3 x_{54,1}x_{16,4} + \alpha_4 x_{7,2}x_{63,3} \\
&+ \alpha_5 x_{7,2}x_{16,4} + \alpha_6 x_{63,3}x_{16,4} + \alpha_7 x_{48,1}x_{17,2} + \alpha_8 x_{48,1}x_{91,3} + \alpha_9 x_{48,1}x_{76,4} \\
&+ \alpha_{10} x_{17,2}x_{91,3} + \alpha_{11} x_{17,2}x_{76,4} + \alpha_{12} x_{91,3}x_{76,4} + \alpha_{13} x_{12,1}x_{20,2} + \alpha_{14} x_{12,1}x_{84,3} \\
&+ \alpha_{15} x_{12,1}x_{47,4} + \alpha_{16} x_{20,2}x_{84,3} + \alpha_{17} x_{20,2}x_{47,4} + \alpha_{18} x_{84,3}x_{47,4} + \delta_1 x_{54,1}x_{7,2}x_{63,3} \\
&+ \delta_2 x_{54,1}x_{7,2}x_{16,4} + \delta_3 x_{54,1}x_{63,3}x_{16,4} + \delta_4 x_{7,2}x_{63,3}x_{16,4} + \delta_5 x_{48,1}x_{17,2}x_{91,3} \\
&+ \delta_6 x_{48,1}x_{17,2}x_{76,4} + \delta_7 x_{17,2}x_{91,3}x_{76,4} + \delta_8 x_{48,1}x_{91,3}x_{76,4} + \delta_9 x_{12,1}x_{20,2}x_{84,3} \\
&+ \delta_{10} x_{12,1}x_{20,2}x_{47,4} + \delta_{11} x_{20,2}x_{84,3}x_{47,4} + \delta_{12} x_{12,1}x_{84,3}x_{47,4} \qquad (5.2)
\end{aligned}$$

As before, x_{ij} equal to 1 when the pseudo-domain is in the considered sequence and in a specific position otherwise it is 0. In this case the relevant elements and their coefficients, drawn from a normal distribution $N(35, 10)$, are described in table 5.4.

Table 5.1: *Non-zero domains*

$j = 1$	$j = 2$	$j = 3$	$j = 4$
18 (40.131)	53 (36.991)	63 (42.866)	1 (34.591)
67 (32.378)	23 (38.578)	37 (25.229)	16 (49.442)
54 (43.981)	71 (31.034)	78 (31.903)	76 (40.864)
16 (27.829)	17 (44.619)	14 (34.850)	80 (37.748)
86 (35.693)	37 (35.352)	32 (26.392)	47 (38.896)
85 (37.257)	77 (37.346)	44 (29.733)	57 (32.174)
12 (41.074)	87 (34.551)	92 (32.147)	93 (35.515)
74 (28.378)	20 (39.471)	91 (37.414)	26 (36.408)
41 (29.302)	39 (33.131)	84 (35.480)	59 (32.628)
48 (43.378)	7 (46.291)	12 (32.061)	13 (32.710)

The values of the coefficients of the non-zero elements are shown in brackets

The optimal solution is $(54, 7, 63, 16)$ with the response value equal to 232.426.

Discrete Rosenbrock Function

In mathematical optimization, the Rosenbrock function is a non-convex function used as a performance test problem for optimization algorithms introduced by Rosenbrock [65].

The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. To converge to the global minimum, however, is difficult.

The Rosenbrock function is defined by

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{5.3}$$

It has a global minimum at $(x, y) = (1, 1)$ where $f(x, y) = 0$. A different coefficient of the second term is sometimes given, but this does not affect the position of the global minimum.

A more involved variant is

$$f(x) = \sum_{i=1}^{N-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2] \quad \forall x \in \mathbb{R}^N \tag{5.4}$$

This variant has been shown to have exactly one minimum for $N = 3$ (at $(1, 1, 1)$) and exactly two minima for $4 \leq N \leq 7$.

In our case, $N = 4$ and each x is a vector of 4 elements. Each element in x can assume 95 real numbers equally spaced between $-5, \ldots, 5$.

5.5 Simulative Study

In this section we study the performance of MACD and NACO compared with different versions of $\mathcal{MAX} - \mathcal{MIN}$ Ant System. All the algorithms are implemented in R. All algorithms have applied fitness functions (5.1), (5.2) and (5.4) to determine the quality of candidate solutions and to test their performance. The following five approaches are run for 100 simulations:

- $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS);
- $\mathcal{MAX} - \mathcal{MIN}$ Ant System with Iterative Improvement Local Search (Local-$\mathcal{MM}$AS);
- $\mathcal{MAX} - \mathcal{MIN}$ Ant System with Simulated Annealing (SA-$\mathcal{MM}$AS);
- Model Based Ant Colony Algorithm (MACD);
- Naïve Ant Colony Optimization (NACO).

The first three algorithms have the following settings:

The MACD has two different settings, one for the Simulative Model Based Phase (S-MB phase) and one for the Learning and Experimental Phase (L-E phase), as shown in table 5.4.

	N	Gen.	ρ	Heuristic	Local Seach	Pred. Model
$\mathcal{MM}$AS	96	53	0.80	No	No	No
Local-$\mathcal{MM}$AS	96	27	0.80	No	I.I.	No
SA-$\mathcal{MM}$AS	96	12	0.80	No	SA	No

Table 5.2: Paramenter Setting of the three standard algorithms. N is the population size, Gen. the number of generations, ρ the evaporation factor, Pred. Model stands for Predictive Model. I.I. is the Iterative Improvement Local Search and SA is Simulating Annealing.

	Steps	Iteration	p_0	α
SA	2	190	0.80	0.95

	Neighborhood
I.I.	204

Table 5.3: Paramenter Setting of the two Local Searches. I.I. stands for Iterative Improvement Local Search and SA stands for Simulating Annealing

The parameter settings for NACO are shown in Table 5.5. From Table 5.5, it is possible to understand that in the simulative case studies the NACO approach does not use any heuristic information. In the real case, this approach will include also the Similarity matrix presented in chapter 1. This information will allow the method to better perform in the search phase and to be faster in the search of a possible optimum.

Before to start the analysis of the algorithms, we test the performance of the methods on the three benchmark functions. We apply a non parametric analysis of variance based on the comparison of the medians. We compare the distributions of the max values obtained from each methods using the Kruskal-Wallis test [16]. The non parametric test compares between the medians of samples to determine if the samples have come from different methods. We select an $\alpha = 0.05$. Looking at the resulting p-values for each analysis we refuse the null hypothesis and we conclude that the behaviour of the responses is significantly different between the different methods.

S-MB Phase						
	N	Gen.	ρ	Heuristic	Local Seach	Pred. Model
SA-$\mathcal{MM}$AS	96	100	0.80	No	SA	Yes

	N	Gen.
L-E Phase	96	11

Table 5.4: Paramenter Setting of the Model Based Ant Colony Design. L-E Phase is refered to the real experimentation.

	N	Gen.	ρ	Heuristic	β	δ	threshold $\gamma \in \mathbb{R}$
NACO	96	11(53)	0.80	No	1	2	80%

Table 5.5: Paramenter Setting of the Naïve Bayes Ant Colony Optimization. N is the population size, Gen. the number of generations (11 in the case of PRM and PSRM, 53 in the case of Discrete Rosenbrock Function), ρ the evaporation factor. No heuristic information is used. β is the weight for the pheromone. δ controls the weight of *Naïve Information* and γ is considered by the Naïve Bayes Classifier.

Using the Polynomial Regression Model (PRM) as the benchmark function, $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) is not able to find good solution for this test case (Fig. 5.6). Clearly a local search technique to replace the heuristic information is needed. We introduce two different local search algorithms: (i) Iterative Improvement Algorithm and (ii) Simulated Annealing. The first modification of the $\mathcal{MM}$AS shows better results with respect to the standard version but there is still room to improve the performance of the algorithm. In fact using the Simulated Annealing as the local search we are able to reach better results. In Fig. 5.6 (b) we can see that the median of the distribution is higher for the $\mathcal{MM}$AS with the Simulated Annealing. We can say that the SA-$\mathcal{MM}$AS has a better chance of reaching higher solutions with respect to the other versions.

In Fig. 5.7 the behaviour of the three algorithms is confirmed with the second test case. In this situation, the ability of SA-$\mathcal{MM}$AS is more evident, moreover it is able to find higher score values. In the last iterations, from iterations 4000 to 5000, the performance of SA-$\mathcal{MM}$AS and Local-$\mathcal{MM}$AS

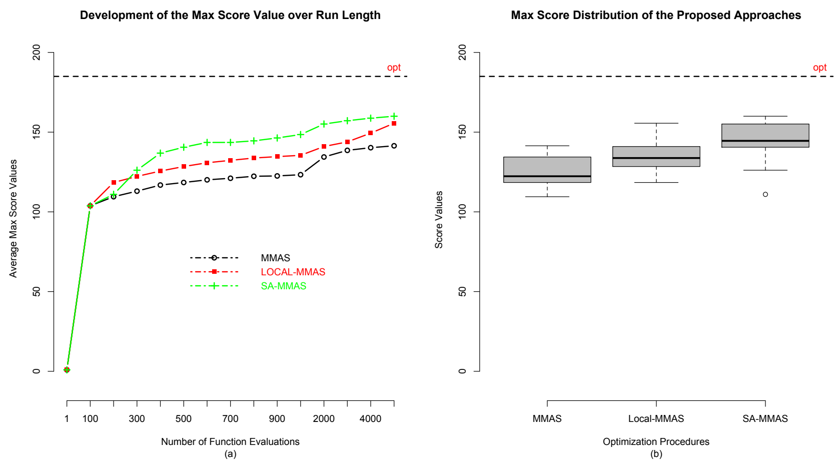


Figure 5.6: Performance comparison between the algorithms without the improvement. The candidate solutions are evaluated with the Polynomial Regression Model. (a) shows the development of the average max score value (calculated on 100 simulations) over run length and (b) Distribution (based on 100 simulations) through boxplot representation.

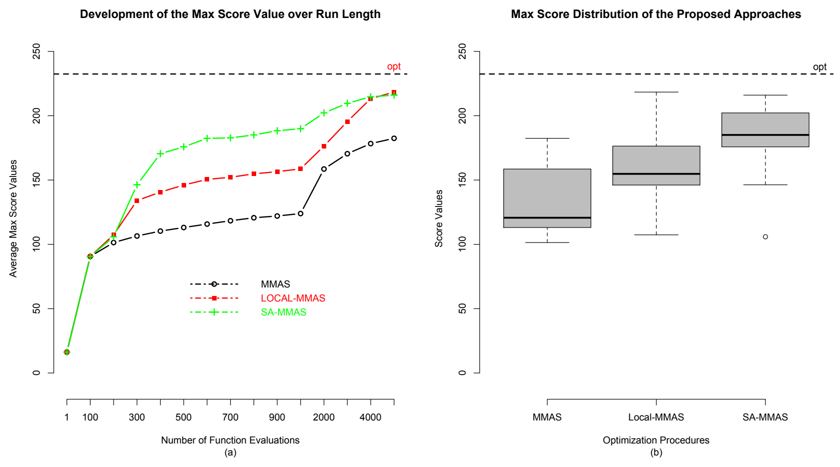


Figure 5.7: Performance comparison between the algorithms without the improvement. The candidate solutions are evaluated with the Polynomial Sparse Regression Model. (a) shows the development of the average max score value (calculated on 100 simulations) over run length and (b) Distribution (based on 100 simulations) through boxplot representation.

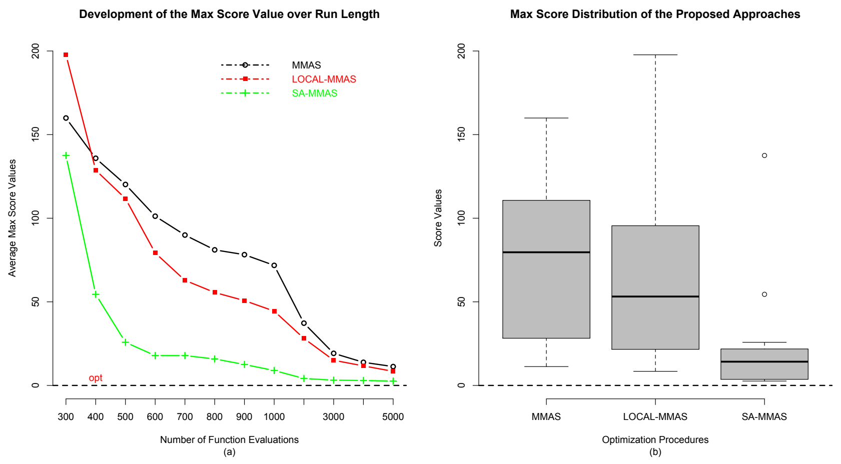


Figure 5.8: Performance comparison between the algorithms without the improvement. The candidate solutions are evaluated with the Discrete Rosenbrock Function. (a) shows the development of the average max score value (calculated on 100 simulations) over run length and (b) Distribution (based on 100 simulations) through boxplot representation.

are the same but it is likely that SA-$\mathcal{MMAS}$ finds better solutions in the first 1000 function evaluations in comparison with all the other approaches.

In Fig. 5.8 is shown the behaviour of the three algorithms using the Discrete Rosenbrock function as benchmark function. In this case the optimization is aimed to minimize the function. The function evaluations 1 and 100 are not represented. As in the previous test cases, $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MMAS}$) is not able to find a good solution. The Local-$\mathcal{MMAS}$ shows good results with respect to the standard version but, also in this case, SA-$\mathcal{MMAS}$ is able to find higher response values.

Looking at these results, we have decided to use MACD with the more powerful version of the $\mathcal{MMAS}$ tested.

In the following pages we show the results of MACD and Naïve Ant Colony Optimization compared with $\mathcal{MMAS}$ and SA-$\mathcal{MMAS}$. Also in this case, we test the performance of the methods on the three benchmark functions. We apply the Kruskal-Wallis test [16] with an $\alpha = 0.05$ and, looking at the resulting p-values for each analysis, we refuse the null hypothesis. The behaviour of the responses is significantly different between the different

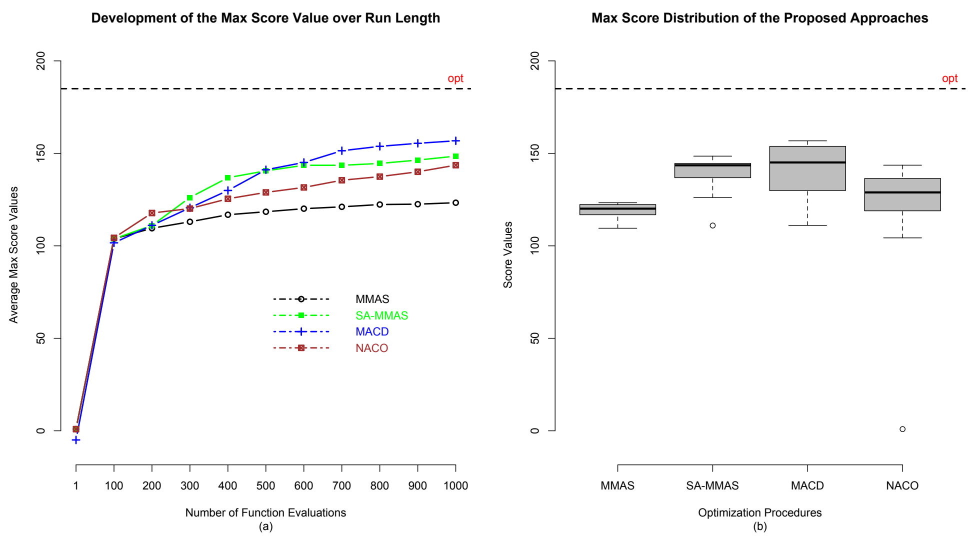


Figure 5.9: Performance comparison between the Model Based Ant Colony Design, the Naïve Ant Colony Optimization, the $\mathcal{MM}$AS and the SA-$\mathcal{MM}$AS . The candidate solutions are evaluated with the Polynomial Regression Model. (a) shows the development of the average max score value (calculated on 100 simulations) over run length and (b) Distribution (based on 100 simulations) through boxplot representation.

methods.

The algorithms are run for at most 1000 function evaluations, for the Polynomial Regression Model (PRM) and the Polynomial Sparse Regression Model (PSRM). In the case of Discrete Rosenbrock Function, we decide to allow the algorithms to run for 5000 function evaluations for the non linear nature of the function.

We apply a non parametric statistical hypothesis test for assessing whether two independent samples of observations have equally large values, namely Mann-Whitney U test [16] (also called the Mann-Whitney-Wilcoxon or Wilcoxon rank-sum test). Also in this case, we compare the difference of the max values reached by the different approaches. More precisely, the null hypothesis is that the distributions of both samples are equal and the alternative hypothesis is that there is a location shift in one sample, then we can interpret a significant Mann-Whitney-Wilcoxon test as showing a significant difference in medians. In Table 5.6, the p-value for each test is shown. We select an

$\alpha = 0.05$. We conclude that MACD and NACO obtain better results with respect to $\mathcal{MM}$AS but they do not have a significant difference in median with respect to SA-$\mathcal{MM}$AS.

X	Y	$p - value$
MACD	SA-$\mathcal{MM}$AS	0.09578
MACD	$\mathcal{MM}$AS	0.03795
NACO	SA-$\mathcal{MM}$AS	0.13854
NACO	$\mathcal{MM}$AS	0.02365

Table 5.6: Mann-Whitney U test. Alternative Hypothesis: X has a significant difference in median with respect to Y. $\alpha = 0.05$

Anyway, in Fig. 5.9 we can see that at 500 function evaluations the performance of the SA-$\mathcal{MM}$AS and MACD are almost the same. From the 500$-th$ evaluation, it is likely that MACD finds better solutions with respect to the other algorithms. NACO is not performing well with respect to SA-$\mathcal{MM}$AS and MACD but it is competitive with $\mathcal{MM}$AS. Anyway it is showing satisfactory results. In the first function evaluations our approaches are not able to reach good results due to the limited amount of data available for the learning phase of the predictive model and the clustering part of the NACO. After 400 function evaluations the predictive model starts to be more reliable, NACO needs more data to extract sufficient information to boost the performance of the algorithm.

In Fig. 5.9 (b) the range of the Max Score Distribution of the MACD is higher with respect to SA-$\mathcal{MM}$AS. We presume that it is due to the improvement of the score that, at each step, it is higher thanks to the Simulative Model Based Phase (see Fig. 5.2). NACO got a smaller range of the score.

With the Polynomial Sparse Regression Model (PSRM) the performance of MACD is more evident. NACO is showing a better behaviour in this case with respect to the previous one. In Fig. 5.10, MACD has a better chance of finding the a good solutions from the 500$-th$ evaluations onward. In this case too, the first function evaluations are important for the learning phase of the predictive model. In NACO the response value is constantly increasing iteration by iteration reaching the same results obtained by the

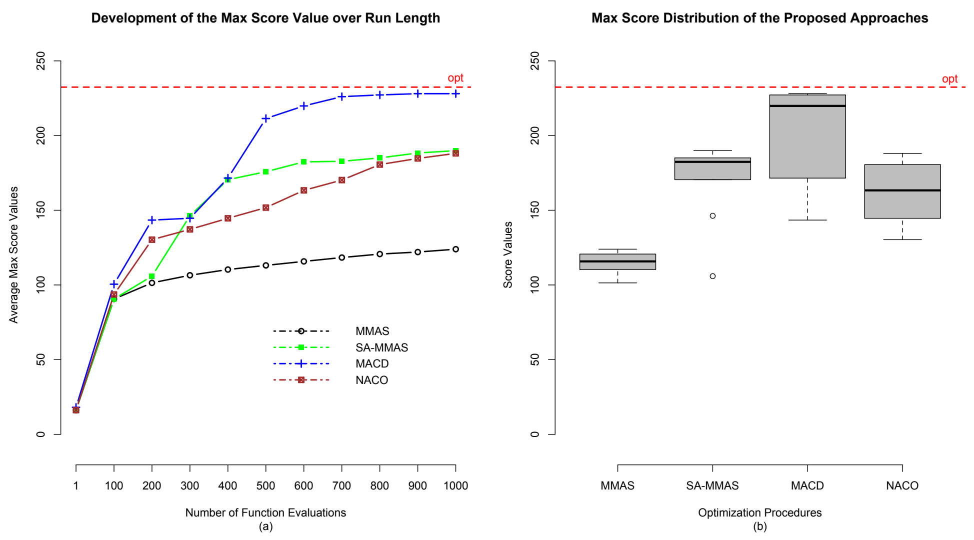


Figure 5.10: Performance comparison between the Model Based Ant Colony Design and the $\mathcal{MM}$AS and the SA-$\mathcal{MM}$AS . The candidate solutions are evaluated with the Polynomial Sparse Regression Model. (a) shows the development of the average max score value (calculated on 100 simulations) over run length and (b) Distribution (based on 100 simulations) through boxplot representation.

SA-$\mathcal{MM}$AS. From these preliminary results, MACD seems to be the most promising approach. In fact, in Fig. 5.11 is shown that MACD is able to reach the optimum of the PSRM after only 200 function evaluations.

Also with the PSRM, we apply the Mann-Whitney U test. In Table 5.7, the p-values are shown. MACD has a significant difference in median with respect to $\mathcal{MM}$AS and SA-$\mathcal{MM}$AS.

X	Y	$p - value$
MACD	SA-$\mathcal{MM}$AS	0.05367
MACD	$\mathcal{MM}$AS	0.00354
NACO	SA-$\mathcal{MM}$AS	0.09978
NACO	$\mathcal{MM}$AS	0.00261

Table 5.7: Mann-Whitney U test. Alternative Hypothesis: X has a significant difference in median with respect to Y. $\alpha = 0.05$

In the case of Discrete Rosenbrock Function the behaviour of MACD and NACO is different with respect to the previous two function. In Fig.

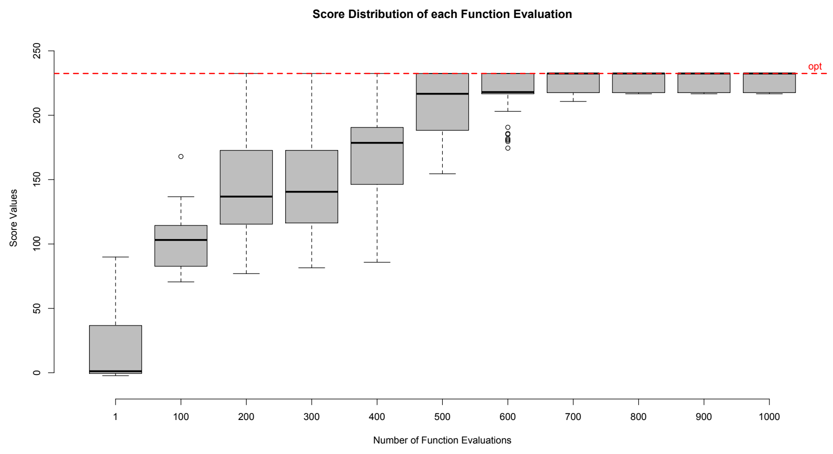


Figure 5.11: Max Score Distribution (based on 100 simulations) of the Model Based Ant Colony Design among the 1000 function evaluations. The candidate solutions are evaluated with the Polynomial Sparse Regression Model.

5.12 the best algorithm is the hybrid version of $\mathcal{MM}$AS coupled with the Simulated Annealing. Anyway, NACO reaches satisfactory results and it has better performance with respect to the standard version of $\mathcal{MM}$AS and, on average, it is able to minimize the function decreasing the response value until a good approximation of the minimum. MACD shows good performance but it has some limitations to reach good results due to the non linearity of the considered function. In fact, the predictive model used in the Simulative Phase of the algorithm is a linear one so it is not able to recognize correctly the possible relations between variables.

Applying the Mann-Whitney U test, we can say that MACD, NACO and the $\mathcal{MM}$AS have comparable performance and they do not show significant different in median.

The analysis of the performance of the Model Based Ant Colony Design and the Naïve Ant Colony Optimization demonstrates that our approaches are able to reach good solutions in few iterations in the three case studies, overall in the case of MACD. The presence of the predictive model and a further investigation of the surface, obtained from the model, using Simulating Annealing help MACD to reach good results. In fact, MACD is able to perform better in the two first case studies, NACO reaches better results in

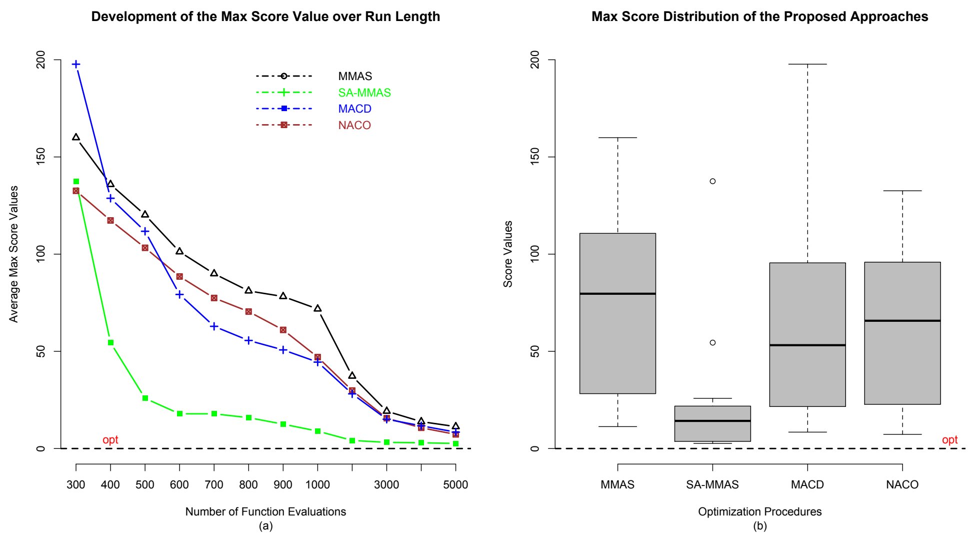


Figure 5.12: Performance comparison between the Model Based Ant Colony Design, the Naïve Ant Colony Optimization, the $\mathcal{MM}$AS and the SA-$\mathcal{MM}$AS . The candidate solutions are evaluated with the Descrete Rosenbrock Function. (a) shows the development of the average max score value (calculated on 100 simulations) over run length and (b) Distribution (based on 100 simulations) through boxplot representation.

the last case (in terms of minimum value reached). NACO is suffering the absence of heuristic information and, during the experimentation, it will be able to reach better results. In the case of MACD, the nature of the predictive model used in the Simulative Phase, limits the ability of the approach to optimize the Discrete Rosenbrock Function. SA-$\mathcal{MM}$AS seems to be a good compromise in all the functions but this optimization algorithm is not usable in the real experimentation due to the presence of a local search. In fact, the real experimentation is based on a parallel evaluation of 96 enzymes and It is not allow to test an enzyme at each step. These two features of the experimentation do not permit to use a local search approach that it is based on a single-point search.

The experimentation will test the ability of the two approaches in a real contest and to understand if they are good solutions in the case of complex function. In the next chapter, we will present the results obtained with the two methods on the motivating problem and we will analyze the time elapsed to generate a new set of candidate solutions for the MACD and NACO.

Chapter 6

Results

6.1 Introduction

In this thesis, we have proposed and investigated the performance of new methods to tackle complex problems. We now apply our proposed solutions to the problem of Enzyme Engineering Design.

In this context, we develop a novel biological approach to create enzymes, whose main function is to promote chemical reactions (see Chapter 1).

During the experimentation, the number of enzymes that can be tested each time is 96. This number is chosen in accordance with the dimension of the well-plate. According to the biologists, the procedure can be competitive with classical biological techniques if and only if it is able to find the optimal or a good enzyme in, at most, 5 generations. In this chapter, we present the results obtained in these 5 generations.

The initial set of enzymes are randomly chosen in the experimental space. Randomness (instead of prior knowledge) allows the exploration of the space in areas not anticipated by prior knowledge but where interesting new effects may possibly reside. This initial design, or the first generation of experimentation, has been conducted and responses observed. All the approaches start from this initial set of data.

The first part of this chapter is dedicated to a preliminary analysis of the starting set of enzymes, then we present the results obtained by Model Based

Ant Colony Design and Naïve Bayes Ant Colony Optimization.

6.2 Preliminary Analysis

We now show the results obtained from the first generation of enzymes.

In Fig. 6.1, we can see the distribution of the score. The main descriptive statistics are summerized in Table 6.1.

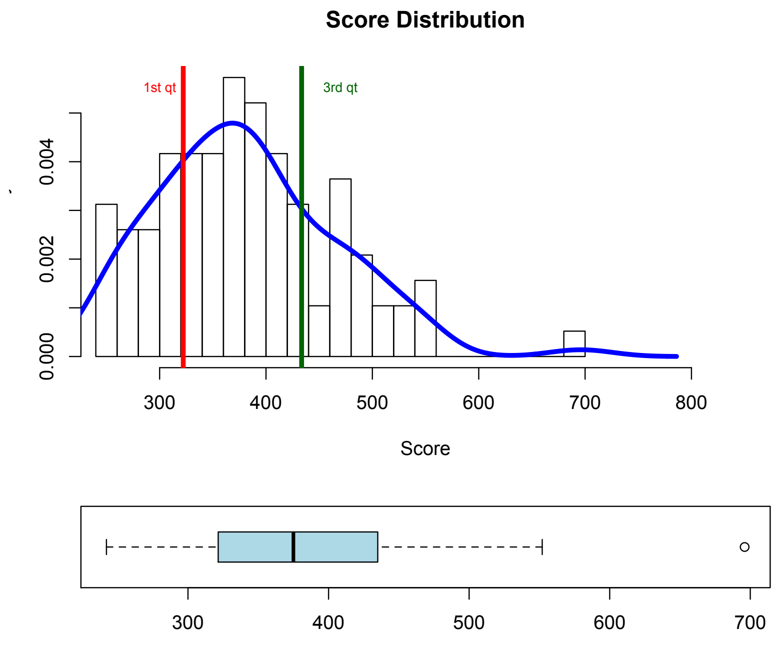


Figure 6.1: Score Distributions for Generations 1.

Gen.	*Min.*	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max*	*St. Dev.*
1	242.0	322.2	375.0	381.4	433.5	696.0	26.9

Table 6.1: Descriptive Statistics of Generation 1

In general, the values reached by the score are not really high and, in Fig. 6.1, we can see an asymmetrical distribution around the mean value except for an outlier, the enzyme composed by pseudo-domains $(79, 29, 2, 22)$ with a score of 696. However not all pseudo-domains were tested. In fact, pseudo-domains number 19, 22, 23, 26 and 51 do not appear in any position in any sequence. This could be a problem in the next step of the approaches. The proposed method must be able to move in the search space including areas where there is no information whatsoever.

We know that we are interested in maximizing the response, so we can find some pseudo-domains in each position of the enzymes, which are present in experimental trials with high system response values. In fact, we can identify some pseudo-domains in each position that can be part of candidate enzymes that are more likely to yield higher response.

In Figure 6.2, as an example, we present the pseudo-domains in position 1. The analysis can be extended to all the other positions. The pseudo-domains that appeared in experiments whose response values exceed the 3^{rd} quantile threshold are marked by a green vertical column and they are shown in the left part of the plot. In the right part, we can see the pseudo-domains appearing in experiments whose response values did not exceed the 1^{st} quantile threshold. We can immediately highlight that some of them are present only in experiments with higher response values ($e.g$ pseudo-domains labelled as 24, 39, 88) and others appear both in high values and in low values of the response ($e.g$ pseudo-domains labelled as 8, 75).

Accordingly, with the information derived by this preliminary analysis it is reasonable that single pseudo-domains cannot be sufficient to account for a higher response value of the system. Only when these pseudo-domains interact with other pseudo-domains in different positions it is possible to achieve much higher response values.

Figure 6.3 shows the interaction between pseudo-domains in the first and second position of the enzymes. The green bar is associated to interactions between pseudo-domains, which lead to higher values of the response (over the 3^{rd} quartile of the distribution of the response).

We can observe that some pseudo-domains in position 1 ($e.g$ 8 and 80) interacting with different pseudo-domains in position 2 generate opposite results; this implies that some pseudo-domains are not important by them-

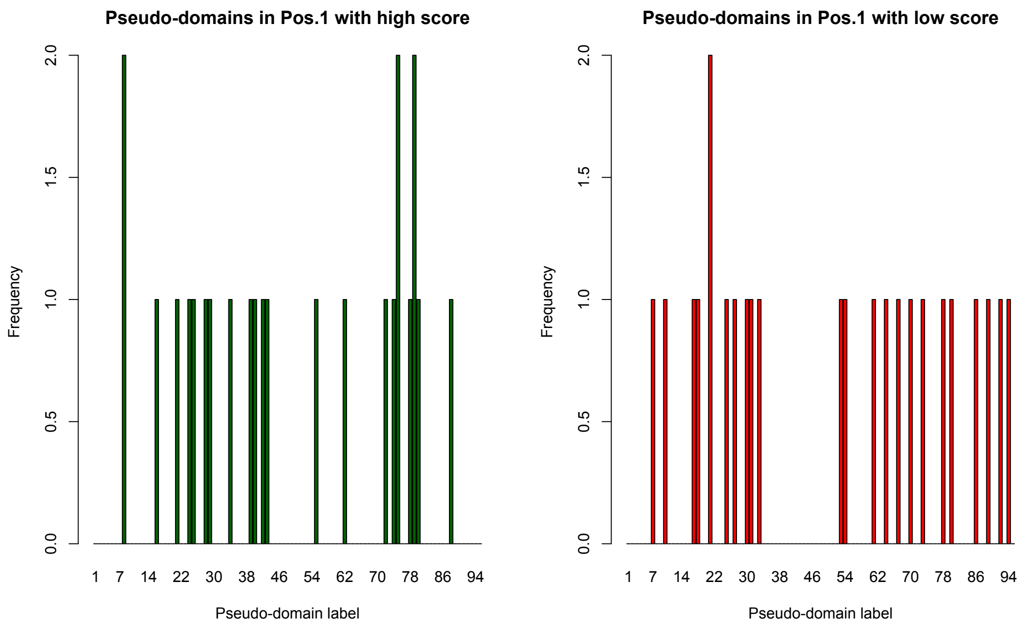


Figure 6.2: Presence of domains in position 1 in the lowest and highest experiments of the first generation of points.

selves, but they need to interact with other pseudo-domains to form an enzyme building block with higher probability to be good.

These results are a good starting point for our approaches and allow us to extract some information from the data and to confirm considerations done by the biologists at the begining of the experiments.

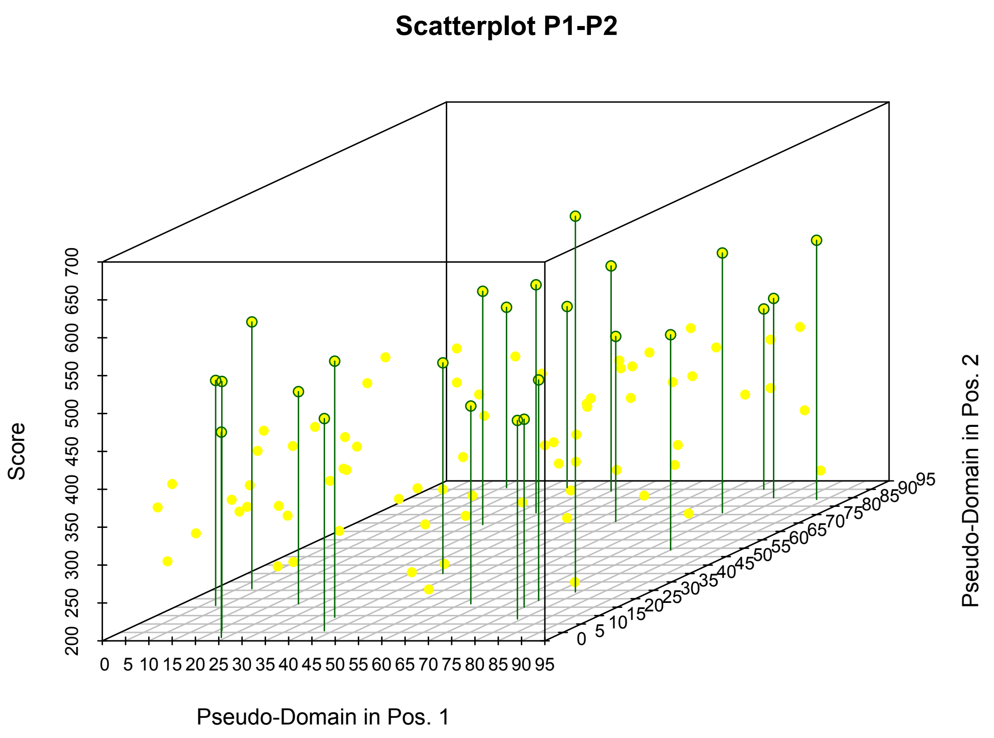


Figure 6.3: Interaction between pseudo-domains in position 1 and in position 2 at the first generation: the green bar represents high values of the response.

6.3 Model Based Ant Colony Design: Implementation and Results

The Model Based Ant Colony Design (MACD) is characterized by two different phases, as shown in Figure 5.2 in Chapter 5:

– Simulative Model Based Phase (S-MB phase): this phase is responsible for the search procedure of the approach. The predictive model simulates the response allowing the algorithm to move in the complex space for selecting the next set of proteins;

– Learning and Experimental Phase (L-E phase): during this part of the approach, the candidate enzymes, selected by the S-MB phase, are tested and added to a dataset containing all the evaluated enzymes. Generation by generation new candidate enzymes extend the dataset permitting a refinement of the predictive model. The evaluation of the points is done as explained in Chapter 1.

These two phases have different settings for the parameters of the algorithm. In S-MB phase the parameters are chosen following the considerations obtained during the simulative case study. In the L-E phase, the parameters depend on the real experimentation so we have to consider the limits imposed by biologists. See Table 6.2.

S-MB Phase						
	N	Gen.	ρ	Heuristic	Local Seach	Pred. Model
SA-$\mathcal{MM}$AS	96	100	0.80	No	SA	Yes

	N	Gen.
L-E Phase	96	5

Table 6.2: Paramenter Setting of the Model Based Ant Colony Design. L-E Phase is to the real experimentation.

	Steps	Iteration	p_0	α
SA	2	190	0.80	0.95

Table 6.3: Paramenter Setting of Simulating Annealing (SA).

Considering the small number of generations that can be evaluated during the experimentation and the need to explore the search space as much as possible, we decide to implement the following rule in the algorithm:

– at step i the best solution is compared with the best solution at step $i-1$. If the best solution at step $i-1$ is not incremented by at least 4% then all the probabilities, p_{ij}, with which an ant chooses to go from one pseudo-domain to another, are set to be equal (uniform distribution). This rule is applied starting from $i = 3$.

This rule enables the approach not to stop too early in a specific area of the experimental space.

In Table 6.4 some descriptive statistics of the 5 generations of enzymes are shown. In generation 1, the best reached value is equal to 696.0 and the average is 381.4. Generation by generation, MACD was able to increase the value of the best enzyme by 8% and the average value by more than 50%. The worst generation is number 4 due to the choice of making it independent of previously acquired information. In fact, the best value of generation 2 is not incremented by at least 4% in generation 3.

Gen.	*Min.*	1*st Qu.*	*Median*	*Mean*	3*rd Qu.*	*Max*	*St. Dev.*
1	242.0	322.2	375.0	381.4	433.5	696.0	26.9
2	410.0	505.5	554.5	546.7	594.0	724.0	68.3
3	444.0	546.0	575.5	575.9	603.8	745.0	49.2
4	256.0	386.8	447.5	446.1	511.2	647.0	85.4
5	539.0	565.5	589.5	599.1	625.0	756.0	44.4

Table 6.4: Main descriptive statistics for all the 5 generations.

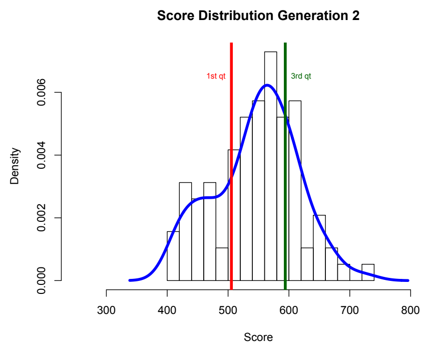
In Figure 6.4, the trend in the 5 generations is more evident. Step by step the distribution of the response is moving to higher values. In fact the minimum value of generation 5 is 539.0, which is higher than all the minimum values of the generations. It means that MACD is moving to a part of the search space where the responses are higher.

To understand if the approach is in a plateau or if it is moving to a good region of the search space, we decided to do a new generation of enzymes. The descriptive statistics are summarized in Table 6.5. Fig. 6.5 shows the distribution of the generation.

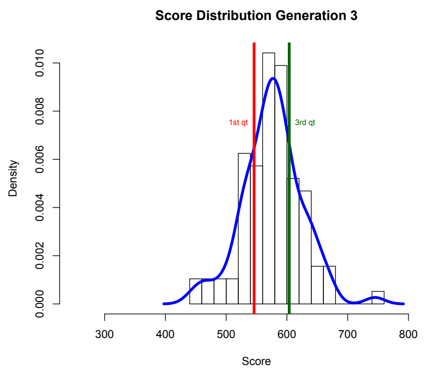
Gen.	*Min.*	1*st Qu.*	*Median*	*Mean*	3*rd Qu.*	*Max*	*St. Dev.*
6	537.0	579.5	604.0	616.1	635.5	834.0	52.8

Table 6.5: Descriptive Statistics for Generation 6.

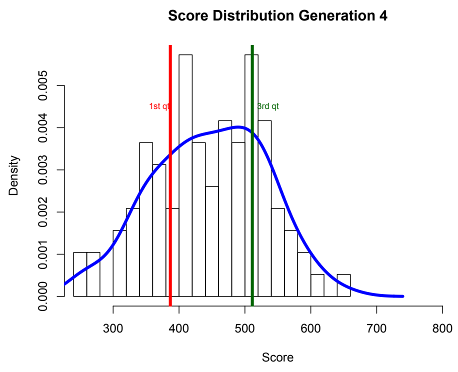
As we can see, in the last generations, MACD was able to find a higher score with respect to the other previous generations. With this application



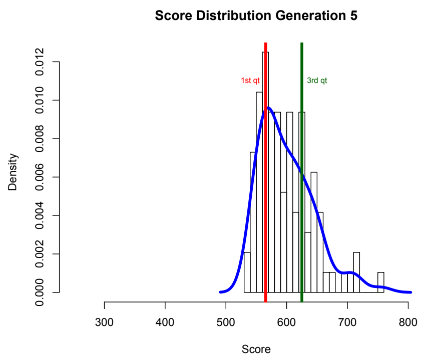
(a)



(b)



(c)



(d)

Figure 6.4: Score Distributions for Generations number 2 (a), 3 (b), 4 (c) and 5 (d).

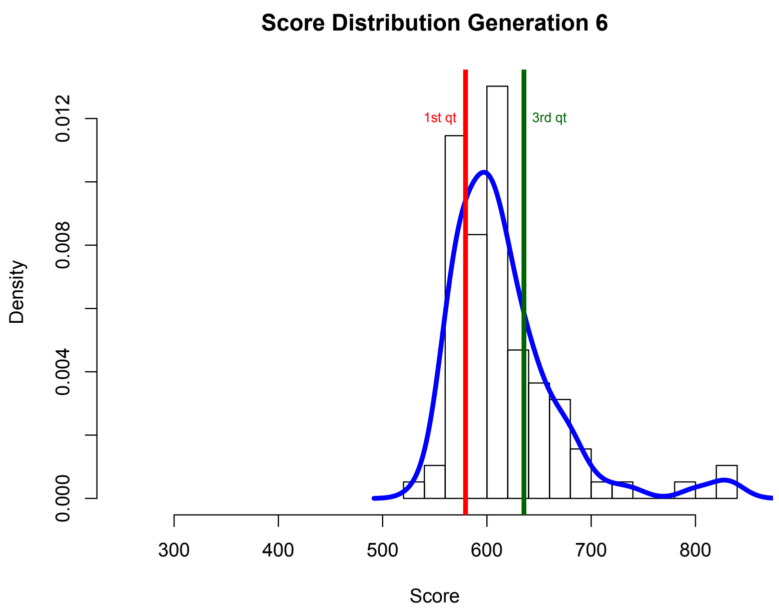


Figure 6.5: Score Distributions for Generation 6.

we want to investigate if the method is able to move towards good regions of the search space starting from a set of randomly chosen enzymes. We apply the Mann-Whitney U test [16], where the null hypothesis is that the distributions of both samples are equal and the alternative hypothesis is that there is a location shift in one sample, it means that the two samples show a significant difference in medians. We apply the non parametric test between generation 6 and generation 1 to understand if MACD is able to increase the score values from the first generation to the last one. The resulting p-value is equal to 2.2×10^{-16} then we refuse the null hypothesis and we accept the alternative one. The two generations show a significant difference in medians. In Fig. 6.6, it is evident that the median of generation 6 is higher with respect to generation 1.

In the last generation the best reached enzyme Score was equal to 834. We can observe that the average of the generation is not increased much with respect to the previous one. This is because it is not stuck in a region with high Score but is able to explore the search space.

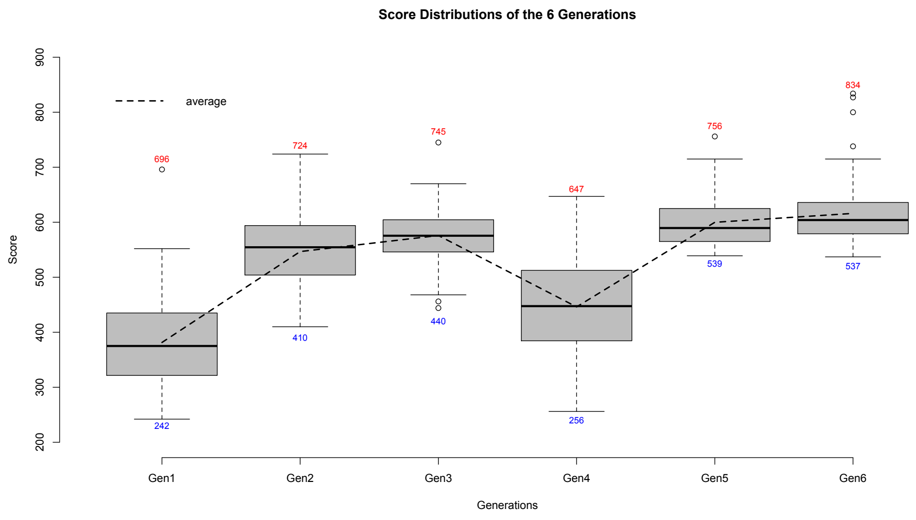


Figure 6.6: Score Distributions for 6 Generations.

In this problem, we do not know the maximum of the unknown function so we cannot say if the solution found by MACD is the optimum or a good approximation, but the approach is able to move in the search space from a "bad" region to a new one with higher values.

We highlight that MACD takes 40 minutes to produce a new set of enzymes to be tested. In this context it is a reasonable time.

6.4 Naïve Bayes Ant Colony Optimization: Implementation and Results

As described in Chapter 5, Naïve Bayes Ant Colony Optimization (NACO) is an optimization algorithm based on the combination between Ant Colony Optimization and Naïve Bayes Classifier. The main aim of this approach is to extract the relevant information from the available set of evaluated enzymes using the Naïve Bayes Approach and to explore the search space by the ACO algorithm. In this case, NACO is using the similarity matrix as described in Chapter 5, more precisely in Fig. 5.3. The parameter settings of NACO are shown in Table 6.7.

	N	Gen.	ρ	α	β	δ	threshold $\gamma \in \mathbb{R}$
NACO	96	5	0.80	1	1	2	80%

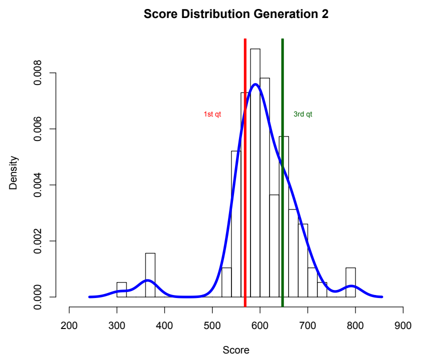
Table 6.6: Paramenter Setting of the Naïve Bayes Ant Colony Optimization. N is the population size, Gen. the number of generations, ρ the evaporation factor. α is the weight for the heuristic information. β is the weight for the pheromone. δ controls the weight of *Naïve Information* and γ is considered by the Naïve Bayes Classifier.

In Table 6.4, some descriptive statistics of the first 4 generations of enzymes are shown. Also with this approach we start from the 96 randomly chosen enzymes presented in Section 6.2. In the second generation, NACO was able to move to a better region of the search space increasing the average value by more the 58%. Moreover the increment of the best value from generation 1 to generation 2 is around 15%. Generation 3 confirms the trend of the best value; in fact, NACO was able to reach the Score 830.0 with less than 288 experimental points. Despite the fact that the best value has increased, the average of the generation is decreasing from 568.8 to 534.2. This behaviour is more evident in Figure 6.7, the distribution of the third generation has moved to the left hand side of the plot. In generation 4, the variability of the distribution is wider but the maximum value is still increasing. In the last generation, NACO is not able to increase the value of the best enzyme .The best enzyme reaches Score 845.0.

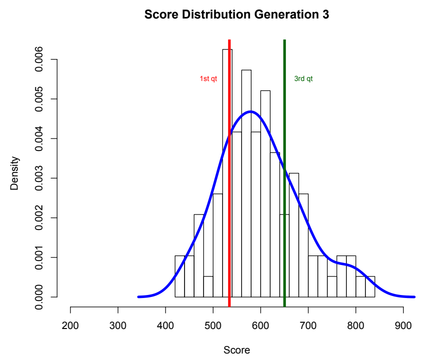
Gen.	*Min.*	1*st Qu.*	*Median*	*Mean*	3*rd Qu.*	*Max*	*St. Dev.*
1	242.0	322.2	375.0	381.4	433.5	696.0	26.9
2	306.0	568.8	601.5	602.7	647.2	792.0	73.8
3	436.0	534.2	589.0	598.3	650.2	830.0	85.8
4	241.0	476.5	579.4	579.4	712.8	845.0	140.3
5	402.0	509.5	582.0	602.4	751.2	841.0	127.6

Table 6.7: Main descriptive statistics for the 5 generations.

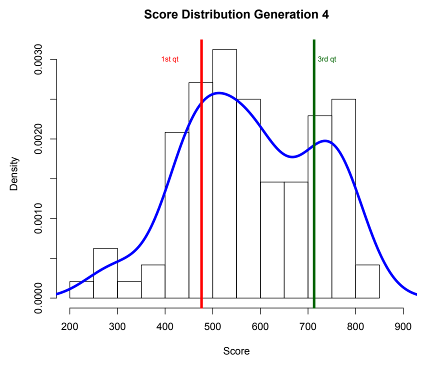
From these generations we notice that NACO is reaching, step by step, better values of the maximum and is exploring the search space in a satisfactory way. Despite that the value of 1^{st} quartile and the average values are



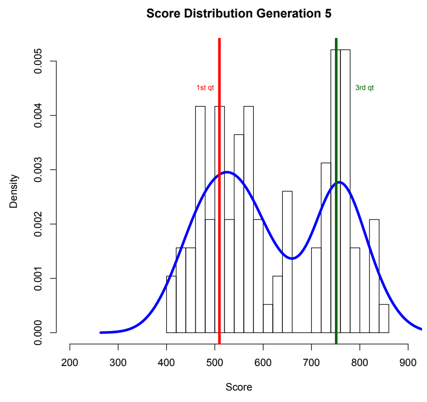
(a)



(b)



(c)



(d)

Figure 6.7: Score Distributions for Generations number 2 (a), 3 (b), 4 (c) and 5 (d).

decreasing generation by generation, the value of the best enzyme and the 3^{rd} quartile are increasing. This behaviour is more evident in generation 4, where the minimum is dwindling. In Fig. 6.8, it is clear that the variability is increasing generation by generation. The information extracted from the Naïve Bayes Classifier and the similarity matrix allows the algorithm to explore more the search space also in regions where the score is low.

Also in this case, we apply the Mann-Whitney U test to understand if there is a real location shift between the last generation and the first one. The p-value is equal to 2.2×10^{-16}. We conclude that NACO is able to move towards good regions of the search space.

Table 6.8 shows the evolution of the best enzyme.

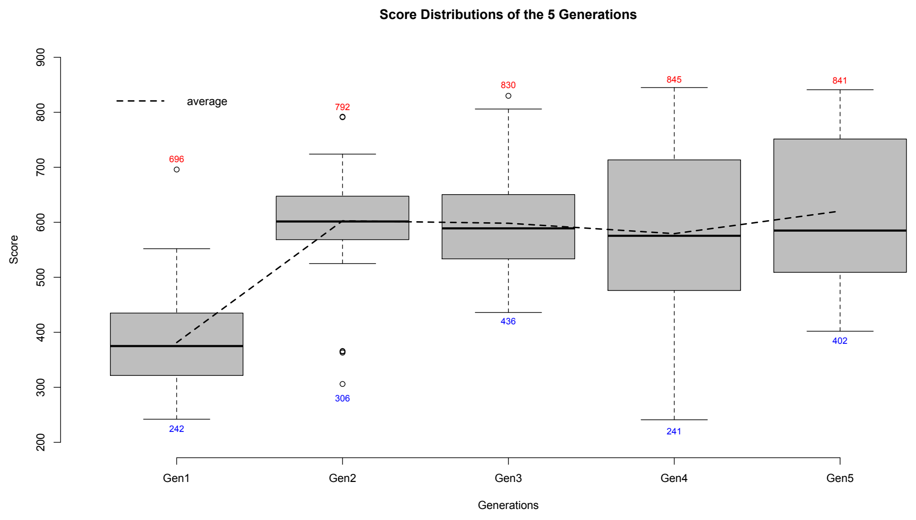


Figure 6.8: Score Distributions for 5 Generations.

	$Pos1$	$Pos2$	$Pos3$	$Pos4$	$Score$
Enzyme Gen1	79	29	2	22	696.0
Enzyme Gen2	79	29	13	84	792.0
Enzyme Gen3	24	29	75	84	830.0
Enzyme Gen4	24	22	75	84	845.0
Enzyme Gen5	24	22	3	84	841.0

Table 6.8: Evolution of the best enzyme generation by generation.

It is important to stress that NACO takes less than 10 minutes to produce a new set of enzymes to be tested.

Chapter 7

Conclusions

The motivating problem of this thesis concerns Enzyme Engineering Design and the principal aim is to develop a procedure for the exploration of a sequence space to identify the best enzyme with a biological functionality. The main problem in this field is the very large experimental space to be searched for the optimization. In our ongoing case study, the experimental space is discrete, contains more the 8.1×10^7 possible enzymes to be tested and we do not have any "a priori" information on the problem.

In this thesis, we have explored the possibility of tackling this problem combining bio-inspired algorithms with advanced statistical techniques. To this effect, we have developed two approaches which merged some of the stronger features of traditional approaches:

– Model Based Ant Colony Design (MACD);

– Naïve Bayes Ant Colony Optimization (NACO).

Both the proposed approaches represent an interactive process where the dialogue between design and laboratory experimentation at each generation creates a path in the combinatorial search space that may lead toward a region of optimality. Generation after generation, the evolving design requires a small number of experimental points to test, and consequently a small investment in terms of resources. The small number of tests make each experimental phase really fast and it is always possible to monitor how much improvement there is from one generation to the next.

Moreover, an advantage of Naïve Bayes Ant Colony Optimization is that it is not computational intensive, in fact it takes less than 10 minutes to generate a new population of candidate enzymes. This advantage allows the researcher to be fast in creation and analysis of possible candidate enzymes.

Furthermore MACD and NACO treat the relevant information contained in each generation in a different way with respect to classical optimization algorithms. The special role played by some factors and some particular levels, or the effect of different order factor interactions are not ignored, but identified and used to construct the next generation design. The predictive model and the Naïve Bayes Classifier have demonstrated to be good at extracting this information from data. These two techniques can in fact uncover the main relevant factors, detect and weight the main interactions. Therefore the predictive model and the Naïve Bayes Classifier can supplement the Ant Colony Optimization approach discovering and communicating information between successive populations of experiments.

MACD and NACO have shown the ability to treat with complex problems characterized by a large experimental space defined by a large number of parameters and their possible levels. Anyway the two approaches can be used in different situations. In fact, in presence of *a priori* information on the problem, NACO can be chosen as the solution. Moreover NACO takes few minutes to produce a new set of candidate solutions to be tested therefore it is a powerful method when the time is an important constraint. MACD can be chosen in absence of information on the problem. The predictive model has demonstrated to be a good way to simulate the response surface and it can make predictions about unexplored regions of the search space. In this way, the experimenter can do hypotheses about regions that he is not going to test in the experimentation.

In the physical experimentation MACD and NACO have demonstrated their ability to identify new enzymes in a large search space of competitive candidates and have shown a remarkable shift of the initial population towards higher response values areas of the search space.

In fact, the best sequence from each generation has been aligned against NCBI non-redundant protein database using BLASTp local alignment software to assess any possible sequence similarity to natural sequences at the level of primary structure. The BLAST analysis did not reveal any significant sequence similarity even when using permissive parameters. Most of the hits retrieved showed limited similarity on short amino-acid stretches and none of the hits belong to the enzymatic family of the target enzyme (*i.e.* C.fusarum serine esterase). These results suggest that the algorithms have explored a region of the sequence space not sampled by Natural evolution, thus identifying artificial sequences deprived of any a priori information that fold into tertiary structure closely related to the target one. Within this framework the algorithms have successfully met the biological requirement of this project as outlined in Chapter 1.

Despite the overall structural similarity, top scoring sequences lack the beta-sheets core of the target enzyme which prevent selected sequences to be used as template without further refinements. The impossibility to achieve the desired target structure may be due to different factors. First, the algorithms might need more iterations to achieve the global optima. Second, the global optima might not be reachable due the finite number of domain used. Indeed, the sequence space related to a 200 amino-acids long protein counts 20^{200} different enzyme sequences. In this work we employed only 95 domains of 50 amino acids long yielding a sequence space of 95^4, which represent only a minor fraction of the original one and might not contain the optimal solution.

Finally, a number of open problems must be solved to allow the development of new methods that combine advanced statistical techniques and optimization algorithms. These problems suggest a variety of research directions. One such direction would be to investigate the possibility to allow an automatic learning of the structure of the predictive model. In MACD, the current framework requires an initial model and it allows to adapt the model to the data at each step. It would be preferable an automatic selection of statistical models. In fact, since the number of variables and their interactions is large, typically there are multiple candidate models yielding similar predictive accuracy. Another direction would be to investigate the possibility to find alternative approaches to extract hidden information from the data. In NACO, we use a Naïve Bayes Classifier to understand which elements mostly influence the score. It would be interesting to study different class of information measures able to detect the relations between elements and to guide the search phase of the optimization algorithms towards regions of optimality.

In terms of applications of MACD and NACO, the attention should be paid in the evaluation of generations needed to reach a satisfactory solution. Furthermore, it would be interesting to widen the performance studies to more complicated benchmark functions, and subsequently to study the methods generalization properties in different application fields.

Acknowledgements

First of all I would like to express my gratitude to Prof. Alessandra Giovagnoli for her continuous support during the period of my doctoral studies.

I wish to thank the PhD School Coordinator, Prof. Daniela Cocchi, for her advice during the PhD period.

Furthermore I would like to acknowledge the help of Thomas Stützle and Mauro Birattari, who assisted me during the time I spent at IRIDIA, research centre of the Université Libre de Bruxelles, and provided very useful comments on this work.

I am grateful to the three directors of the European Centre for Living Technology: Prof. Irene Poli, Prof. Norman Packard and Prof. Kristian Lindgren.

Then special thanks to my colleagues at the ECLT: Davide De March, Michele Forlin, Alessandro Filisetti, Debora Slanzi, Giovanni Minervini, Giovanni Zemella, Davide Ferrari, Alex Graudenzi, Davide De Lucrezia and Elena Lynch.

Thanks to my colleagues for bearing with me in "via Zamboni 18": Marta Francesca Lilja Di Lascio, Enrico Foscolo and Maroussa Zagoraiou.

Thanks to my friends and my family.

Bibliography

[1] T. Aita, M. Iwakura, and Y. Husimi. A cross section of the fitness landscape of dihydrofolate reductase. *Protein Eng*, 14(9):633–638, 2001.

[2] D. Ashlock. *Evolutionary Computation for Modeling and Optimization.* Springer, New York, 2006.

[3] A. C. Atckinson, A. N. Donev, and R. D. Tobias. *Optimum Experimental Designs, with SAS.* Oxford University Press, 2007.

[4] A. Baldi Antognini, A. Giovagnoli, D. Romano, and M. Zagoraiou. Computer simulations for the optimization of technological process. In P. Erto, editor, *Statistics for Innovation*, pages 65–88. Springer, 2009.

[5] R. Baragona, F. Battaglia, and I. Poli. *Evolutionary Statistical Procedures.* Springer-Verlag, 2011.

[6] Y. Bar Yam. *Dynamics of Complex Systems.* Adison-wesley, 1997.

[7] S. Bershtein, M. Segal, N. Bekerman, N. Tokuriki, and D. S. Tawfik. Robustness epistasis link shapes the fitness landscape of a randomly drifting protein. *Nature*, 444(7121):929–32, 2006.

[8] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Acm Computing Surveys*, 35(3):268–308, 2003.

[9] I. Borg and P. J. Groenen. *Modern Multidimensional Scaling: Theory and Applications.* Springer, 2005.

[10] M. Borrotti. *A model based algorithm for evolutionary design of experiments.* Technical Report 24, IRIDIA, 2009.

[11] G. E. P. Box. Evolutionary operation: a method for increasing industrial productivity. *Applied Statistics*, 6:3–23, 1957.

[12] L. Breiman. *Better subset selection using non-negative garotte.* Technical report, University of Berkeley, California, 1993.

[13] V. Cerny. Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *J. Optim. Theory Appl.*, 45:41–51, 1985.

[14] C. Chiarabelli and D. De Lucrezia. The worlds of the prebiotic and never born proteins. *Origins of Life and Evolution of Biospheres*, 27:357–361, 2007.

[15] J. Y. Choi, S. H. Bae, X. Qiu, and G. Fox. High performance dimension reduction and visualization for large high-dimensional data analysis. In *International Conference on Cluster, Cloud and Grid Computing*, 2010.

[16] W. J. Conover and R. L. Iman. Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician*, 35(3):124–129, 1981.

[17] J. De Leeuw. Application of convex analysis to multidimensional scaling. In J. Barra, F. Brodeau, G. Romier, and B. V. Cutsem, editors, *Recent Developments in Statistics*, pages 133–145, 1977.

[18] D. De Lucrezia, G. Minervini, and I. Poli. *A novel scoring function to evaluate secondary structure similarity.* Technical report, ECLT European Centre for Living Technology, April 2009.

[19] D. De Lucrezia, G. Minervini, and I. Poli. *A novel similarity metric to evaluate secondary structure.* Technical report, ECLT European Centre for Living Technology, February 2009.

[20] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organinzing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.

[21] M. Dorigo. *Optimization, Learning and Natural Algorithms (in Italian).* PhD thesis, DEI, Politecnico di Milano, Italy, 1992.

[22] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.

[23] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant colony for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[24] M. Dorigo, V. Maniezzo, and A. Colorni. *Positive feedback as a search strategy.* Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.

[25] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Ieee Transactions On Systems Man and Cybernetics Part B-Cybernetics*, 26(1):29–41, 1996.

[26] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, Mass., 2004.

[27] A. Fernandez-Gacio, M. Uguen, and J. Fastrez. Phage display as a tool for the directed evolution of enzymes. *Trends Biotechnol*, 21(9):408–414, 2003.

[28] R. Fernandez-Lafuente, C. M. Rosell, and J. M. Guisán. Enzyme reaction engineering: synthesis of antibiotics catalysed by stabilized penicillin g acylase in the presence of organic cosolvents. *Enzyme Microb Technol.*, 13(11):898–905, 1991.

[29] M. Forlin, I. Poli, D. De March, N. Packard, G. Gazzola, and R. Serra. Evolutionary experiments for self assembling amphiphilic systems. *Chemometrics and Intelligent Laboratory Systems*, 90(2):153–160, 2008.

[30] M. R. Garey. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1979.

[31] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[32] F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[33] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Pub. Co., Reading, Mass., 1989.

[34] T. L. Griffiths and A. Yuille. A primer on probabilistic inference. *Trends in Cognitive Science*, 10(7):1–11, 2006.

[35] T. Hastie and R. Tibshirani. *Generalized Additive Models.* Chapman and Hall, 1999.

[36] A. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58(3):54–59, 1962.

[37] A. Hoerl and R. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

[38] J. Huang, S. Ma, and C. H. Zhang. Adaptive lasso for sparse high-dimensional regression models. *Statistica Sinica*, 18(4):1603–1618, 2008.

[39] I. M. Johnstone and D. M. Titterington. Statistical challenges of high-dimensional data. *Philosophical Transactions of the Royal Society A-Mathematical Physical and Engineering Sciences*, 367(1906):4237–4253, 2009.

[40] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Network*, volume 4, pages 1942–1948, 1995.

[41] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[42] J. Knowles. Closed loop evolutionary multiobjective optimization. *IEEE Computational Intelligence Magazine*, 4:77–91, 2009.

[43] J. Kohonen, S. Talikota, J. Corander, P. Auvinen, and E. Arjas. A naïve bayes classifier for protein function prediction. *In Silico Biology*, 9:23–34, 2009.

[44] K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence.* Chapmann & Hall, 2004.

[45] C. Koukouvinos, K. Mylona, and D. E. Simos. A hybrid saga algorithm for the contruction of e(s^2) optimal cyclic supersatured designs. *Journal of Statistical Planning and Inference*, 139:478–485, 2008.

[46] J. B. Kruskal and M. Wish. *Multidimensional Scaling.* Sage Pubblications Inc., 1978.

[47] J. Kuelbs and A. N. Vidyashankar. Asymptotic inference for high-dimensional data. *Annals of Statistics*, 38(2):836–869, 2010.

[48] A. H. Land and A. G. Doing. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[49] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Travelling Salesman Problem.* Wiley, 1985.

[50] H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, pages 321–353, Norwell, MA, 2002. Kluwer Academic Publishers.

[51] P. L. Luisi. Contingency and determinism. *Philos Transact A Math Phys Eng Sci*, 361:1141–1147, 2003.

[52] L. J. McGuffin, K. Bryson, and D. T. Jones. The psipred protein structure prediction server. *BMC Bioinformatics*, 16(4):404–405, 2000.

[53] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley and Sons, 1996.

[54] L. Meier, S. Van De Geer, and P. Buehlmann. High-dimensional additive modeling. *Annals of Statistics*, 37(6B):3779–3821, 2009.

[55] G. Minervini, G. Evangelista, L. Villanova, D. Slanzi, D. De Lucrezia, I. Poli, P. L. Luisi, and F. Polticelli. Massive non-natural proteins structure prediction using grid technologies. *BMC Bioinformatics*, 10:1–9, 2009.

[56] T. M. Mitchell. *Machine Learning*. McGraw-Hill, http://www.cs.cmu.edu, on-line edition, 2005.

[57] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[58] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley, 2009.

[59] C. Nicolucci, S. Rossi, C. Menale, T. Godjevargova, Y. Ivanov, M. Bianco, L. Mita, U. Bencivenga, D. G. Mita, and N. Diano. Biodegradation of bisphenols with immobilized laccase or tyrosinase on polyacrylonitrile beads. *Biodegradation*, 2:81–101, 2010.

[60] J. Pearl. Evidential reasoning using stochastic simulation of casual models. *Artificial Intellingence*, 21(11):2657–2666, 1987.

[61] J. Pearl. *Probabilistic Reasoning in Intelligent System*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[62] R. Rahinakumar and W. C. Wimley. Biomolecular engineering by combinatorial design and high-throughput screening: small, soluble peptides that permeabilize membranes. *J. Am. Chem. Soc.*, 130(30):1141–1147, 2008.

[63] P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 71:1009–1030, 2009.

[64] C. Ribeiro. *Essays and Surveys in Metaheuristics*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[65] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3:175–184, 1960.

[66] D. Röthlisberger, O. Khersonsky, A. M. Wollacott, L. Jiang, J. DeChancie, J. Betker, J. L. Gallaher, E. A. Althoff, A. Zanghellini, O. Dym, S. Albeck, K. N. Houk, D. S. Tawfik, and D. Baker. Kemp elimination catalysts by computational enzyme design. *Nature*, 453(7192):190–195, 2008.

[67] C. A. Sarkar, I. Dodevski, M. Kenig, S. Dudli, E. Mohr, E. Hermans, and A. Plückthun. Directed evolution of a g protein-coupled receptor for expression, stability, and binding selectivity. *PNAS*, 105(39):14808–14813, 2008.

[68] T. Stützle and H. Hoos. Max-min ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[69] J. C. Talakad, P. R. Wilderman, D. R. Davydov, S. Kumar, and J. R. Halpert. Rational engineering of cytochromes p450 2b6 and 2b11 for enhanced stability: Insights into structural importance of residue 334. *Arch Biochem Biophys.*, 494(2):151–158, 2010.

[70] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B-Methodological*, 58(1):267–288, 1996.

[71] W. Torgerson. *Theory and Methods of Scaling*. Wiley, 1958.

[72] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 67:768–768, 2005.