INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

TECNOLÓGICO
DE MONTERREY®

A clustering genetic algorithm for inferring protein-protein functional interaction sites

THESIS PRESENTED TO OPT FOR THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE
BY

JOSÉ JUAN TAPIA VALENZUELA

Advisor:	DR. EDGAR EMMANUEL VALLEJO CLEMENTE	
Co-Advisor:	DR. JUAN ENRIQUE MORETT SÁNCHEZ	
Thesis comitee:	DR. CARLOS ARTEMIO COELLO COELLO	
	DR. MIGUEL GONZÁLEZ MENDOZA	
Referees:	DR. JUAN ENRIQUE MORETT SÁNCHEZ	President
	DR. MIGUEL GONZÁLEZ MENDOZA	Secretary
	DR. EDGAR EMMANUEL VALLEJO CLEMENTE	Vocal
	DR. CARLOS ARTEMIO COELLO COELLO	Vocal

Atizapán de Zaragoza, Edo. Mex., Mexico, July 2009

ABSTRACT

In the thesis we present a technique for clustering different proteins into groups that would help biologist to infer significant protein-protein functional interactions. The approach that will be used is the Genetic Algorithms, incorporating techniques such as clustering and parallelization.

We first approach this interaction prediction challenge as an optimization problem, in specific using phylogenetic profiles and treating the formation of clusters based around these profiles as an optimization problem. First results demonstrate that this approach produces results that are competitive with respect to classical clustering approaches.

In the expanded version, we propose the formulation of the protein-protein functional interactions prediction as a multi-objective optimization problem. In this view, clustering is conducted by considering multiple objectives associated to different genomic attributes. Although previous authors have proposed the use of Multi-objective clustering algorithms to mine biological data, it has only been used as a means to increase the clustering capabilities of the algorithm, and never to be able to introduce new biological data into the process. As such, we propose the use of the Multiple Objective Clustering Evolutionary Algorithm as a means to combine the competent clustering capabilities over irregular landscapes that Genetic Algorithms possess, and consider various sources of genomic data as a multi-objective problem.

Particularly, in this work we conducted a series of experiments with three different attributes: phylogenetic profiles, gene directionality and intergenetic distance. We used genomic data provided by the COG and GeCont databases, among others.

Experimental results demonstrated that our method is capable of producing competitive results as validated by experimentally confirmed databases on functional interactions between proteins such as DIP and ECOCYC. The proposed method shows that different genomic attributes can be used simultaneously to increase biological significance in the reconstruction of protein-protein functional interactions.

CONTENTS

1 Introduction **8**
- 1.1 Motivation . . . 10
- 1.2 Problem statement . . . 11
- 1.3 Hypothesis . . . 11
- 1.4 Thesis objectives . . . 11
- 1.5 Thesis organization . . . 12

2 Background **13**
- 2.1 Biological context . . . 13
- 2.2 The importance of proteins . . . 13
  - 2.2.1 Protein-protein interactions . . . 14
  - 2.2.2 Biochemistry methods to determine protein-protein interactions . . . 15
  - 2.2.3 Prediction of functional protein-protein interactions . . . 16
  - 2.2.4 Genomic context . . . 16
  - 2.2.5 Phylogenetic profiles . . . 17
  - 2.2.6 Other *in silico* methods for detecting protein-protein interactions . . . 18
- 2.3 Computational context . . . 18
  - 2.3.1 Simple Genetic Algorithms . . . 18
  - 2.3.2 The grouping genetic algorithm . . . 19
  - 2.3.3 Multi-objective genetic algorithms . . . 19
  - 2.3.4 Using genetic algorithms as a clustering method in Bioinformatics . . . 23
- 2.4 K-means . . . 24

3 Proposed model **25**
- 3.1 The input data . . . 25
  - 3.1.1 The COG Database . . . 25
  - 3.1.2 The GeCont Database . . . 27
  - 3.1.3 Other databases . . . 27
- 3.2 Preparing the databases . . . 28
  - 3.2.1 The COG database . . . 28
  - 3.2.2 The GeCont database . . . 29
  - 3.2.3 The DIP and Ecocyc databases . . . 30
  - 3.2.4 The E. Coli operon database . . . 30
- 3.3 The grouping genetic algorithm . . . 32
  - 3.3.1 Basic Structures . . . 33
  - 3.3.2 Genetic Operators . . . 33
  - 3.3.3 Using a single evaluation function . . . 36
  - 3.3.4 Selection operators . . . 37
  - 3.3.5 Improving the algorithm performance . . . 38
  - 3.3.6 Integrating the algorithm . . . 38
- 3.4 Results Using a single objective evaluation function . . . 39
  - 3.4.1 Number of groups . . . 39

3.4.2 Running time . . . 40
3.4.3 Validation of results . . . 40
3.5 Early Results Discussion . . . 43
3.6 Introducing multi objective fitness evaluation functions . . . 44
3.6.1 Phylogenetic profiles . . . 44
3.6.2 Transcriptional directionality . . . 45
3.6.3 Intergenic distance . . . 45
3.6.4 Cluster size . . . 45
3.7 Implementation of the evaluation function . . . 46

4 Final results **48**
4.1 Formation of the Pareto front . . . 48
4.1.1 Statistical analysis on the pareto front . . . 48
4.2 Execution time . . . 51
4.3 Arranging our algorithm . . . 51
4.4 Validation of results . . . 51

5 Conclusions **53**
5.1 Future work . . . 55
5.1.1 Other computational improvements . . . 56
5.2 Biological future work . . . 56

LIST OF FIGURES

2.1 Central dogma of Molecular Biology 14
2.2 Co-immune precipitation 15
2.3 Interacting proteins are near each other, and transcribe in the same direction 17
2.4 Example of a Pareto front. The yellow region represents the feasible area. The red line represents the pareto front. The blue points represent the current population . . 21
2.5 Example of an unevenly distributed Pareto front 22

3.1 The *lac* operon 28
3.2 Structure of a population 33
3.3 Behavior of the fitness functions using crossover and mutation 34
3.4 Behavior of the fitness functions using mutation only 35
3.5 Example of how the `MergeStrong` operator works 35
3.6 Example of how the `rehashWeak` operator works 35
3.7 Example of how the `weeding` operator works 36
3.8 Average run of the GGA engine 39

4.1 Initial random distribution of the population 49
4.2 Pareto Front formation. The curve is bended towards the viewer 49
4.3 Raw superposition of a subset of our runs of the algorithm 49
4.4 Phylogenetic profiles vs transcription directionality 50
4.5 Intergenic distance vs phylogenetic profiles 50
4.6 Intergenic distance vs transcription directionality 50

LIST OF TABLES

2.1 Example of phylogenetic profiles 17

3.1 Genomes represented in COG database (part I) 26
3.2 Genomes represented in COG database(part II) 27
3.3 Example of an entry of the COG database. 29
3.4 Example of an entry of the GeCont database. Some fields are ommited 29
3.5 The context table entity relationship model 30
3.6 The Gecont-COG table ER model 30
3.7 The context table entity relationship model 31
3.8 An example of the DIP table 31
3.9 Example of early clustering genetic algorithms 31
3.10 Example of early clustering genetic algorithms 32
3.11 Example of a phylogenetic profile centroid 37
3.12 Percentage of correct EcoCYC/DIP group assignations 40
3.13 Excerpt from the GGA group where the *purEK* pair is located 41
3.14 Excerpts from the *k*-means group were the *purEK* disjointed pairs were located . . 42
3.15 Excerpt from the GGA group where the *Leu* and *SpeED* operon pair are located . . 42
3.16 Exerpt from the GGA group where the *hflA* operon pair is located 43
3.17 Example of a context array field from an hypothetical COG 45
3.18 Example field of the context 3 dimensional array 46

4.1 Results chart 52
4.2 Excerpt from the group where the DNA biosynthesis operon is located 52

1 Input Texture 62
2 Output Texture 62
3 Time comparison 63

LIST OF ALGORITHMS

1 k-means 24
2 Evaluation function for the single objective genetic algorithm 37
3 Overview of the complete algorithm 38
4 Algorithm for the creation of a new population 47
5 Kernel for the computation of a distance matrix 61
6 Kernel for obtaining the distance values 63

1. INTRODUCTION

At its very core, every living being on this planet can be represented as an encoded piece of data known as a genome. It is through the expression of this genome that every single biological process operates. When a genome is expressed, it is eventually transformed into the proteome, or the collection of proteins and protein machines that make up as species. While the genome is the footprint that identifies a species, the proteins are the workers through which all the biological process take place, such as DNA replication, transcription and translation itself, splicing, secretion, cell cycle control, etc. It is through the stacking of all the protein processes within a system, or through the multitude of protein-protein interactions that higher level biological systems can be created.

As such, the analysis of the building blocks of Biology is fundamental in our endeavor to increase our biological knowledge. Recent years have seen a tremendous increase in the amount of genomic and proteomic data we have at our disposal. Since the discovery of the genome of the *bacteriophage MS2* in 1976 by Fiers [1], passing through the discovery of the genome of the first free-living organism, the *Haemophilus influenzae* in 1995[2], and that of the Human Genome in 2003, we now have millions of bytes of data to use in the various endeavors biology is engaged in.

However, merely possessing this large amount of data, while highly convenient, is not actually useful by itself. We must be able to extract valuable information from this raw data, information that will allow us to discover new knowledge, be it the epidemiology of a certain disease, the root of certain hereditary condition and others.

Biologists around the world have made a lot of economic and scientific investment into discovering and extracting this kind of characteristics from raw data. Experimental methods such as co-immune precipitation, or microarrays have allowed us to extract much information that tells us about the different properties that protein, and protein-protein complexes have.

However, despite our best efforts there is still a big number of poorly characterized proteins,

that is, proteins whose function has not been clearly recognized. However, there is a strong economic cost involved in this type of *in vivo* experiments, that makes exhaustive experiments unfeasible.

Because of this, the development of computational methods for determining functional relationships between proteins from sequence and genomic data has becoming an increasingly important area of research in bioinformatics and computational biology. In effect, revealing unknown protein interactions in functional pathways and perhaps, their association with disease relies crucially on sound computational algorithms capable of producing meaningful predictions.

A collection of robust computational approaches for reconstructing functional modules of proteins have been developed in recent years [3]. These methods rely crucially on genomic attributes such as gene co-expression, gene co-occurrence, genes proximity, gene directionality, gene fusion, among others [4]. All of these data attributes have been extracted from complete genomes of different species, so they have been available only since recent years. This very large data corpus possesses a vast amount of informative implicit relationships waiting to be discovered by computational techniques.

Among these modern post-genomic computational approaches, phylogenetic profiles –the correlated presence and absence of genes among a collection of organisms, have proven to be particularly effective [5, 6]. Theoretically, with the increasing availability of complete genomes from more organisms, this method holds the promise of increasing efficiency. Particularly, phylogenetic profiles have been used for assigning protein function, for localizing proteins in cells, and for reconstructing metabolic pathways, among other applications.

The efficacy of predictions obtained from phylogenetic profiles depends critically on its cluster analysis stage –the grouping of proteins with similar phylogenetic patterns. Most clustering algorithms used in phylogenetic profiles to date often require *a priori* information on clustering parameters. Namely, the number of clusters and the initial positions of centers, among others, although there are a few exceptions. Further, the absence of negative examples in testing data sets allows for the presence of false positives that need to be discarded experimentally, which is extremely expensive in practice.

Different clustering algorithms have been previously used with phylogenetic profiles. Several approaches can be identified in the literature. Previous work in our laboratory obtained promising results by identifying transitive functional relationships, then analyzing these features in order to detect functional modules of proteins using the Bond Energy Algorithm[7]. Fuzzy C-means has also been used for extending the coverage of predictions to include pairs of interacting proteins with relatively dissimilar phylogenetic profiles [8].

We have found as more promising to develop clustering methods based on genetic algorithms. In previous experiments, clustering genetic algorithms have yielded competitive results on detecting protein-protein functional interactions[9]. Furthermore, they have proved capable of evolving appropriate clustering parameters which eliminates the need for *a priori* empirical specification and tuning. We believe the latter represents an important advance in the practice of cluster analysis in general.

In addition, even though the predictions provided by most of the post-genomic approaches

have proven to be useful, the integration of different genomic attributes within a general purpose, comprehensive algorithms have remained elusive. The exception has been the *a posteriori* additive integration of the results provided by different methods [10].

This combination and integration approach of all the information we have at our disposal is of extreme importance, because of two main factors. First, the information we receive is often incomplete, and second, many times there is a lot of noise in our experimental data, so the information ends up being contradictory. As such, we need methods that are able to take in this fuzzy information that is the product of vastly different methods and combine them into a single data mining algorithm.

All in all, a multi-objective approach to protein-protein functional prediction would be highly convenient in our endeavors to encompass all the biological information at our disposal from a computational point of view. Not only that, but an effective method that allows us to combine all those methods without losing information in the process, but on the contrary, to sinergically build upon combined to discoveries in order to discover new data is the only way to go. As such, we propose MOCEA (Multi Objective Clustering Evolutionary Algorithm for inferring functional protein-protein interactions), a framework we have developed, whose structure and functionality we will explain in the next chapters.

1.1 MOTIVATION

It is known that a living being is formed primarily by five components, those being nucleic acids, lipids, water, carbohydrates and proteins.[11] All of these are highly important and have their own function, however it is the proteins that most define the character of a cell since they are involved in every process within it. They may be enzymes that catalyze biochemical reactions, or have structural or mechanical functions. Cell signaling, cell adhesion, immune responses, their multiple functions are vast. Their full understanding would be of importance not only in the theoretical understanding of our ecosystems, but also in discovering new ways of fighting diseases, genetic disorders and other problems that affect our human species.

However, up to this day the full understanding of the whole range of functions all of the proteome still remains as a long term goal. Proteomics, which is the branch of the biology that dedicates to the research of the structure and expression of proteins, is still an area in its early stages. We know that depending on a proteins structure, patterns of co appearance between proteins among different species, directionality of the RNA transcription and other factors we can draw important conclusions that could be crucial in furthering our understanding of the proteome. However, to this day discoveries have been scarce.

There is a large range of methodologies and algorithms, both in its *in vivo* and *in vitro* incarnations that have been developed to infer these interactions. However, there has been little work on approaches that compile all this information in a single algorithm in order to add the advantages and cover for the disadvantages for each of these methods

1.2 PROBLEM STATEMENT

There exist a necessity to predict protein-protein interactions from Biological data using in silico methods. Furthermore it should be ideal to create a way to predict this interaction networks using information from varied sources, a method that is resistant to incomplete, and often contradictory information, and that yet allows all information to be combined into a single searching algorithm, and correctly use the data to predict the information we are interested in.

1.3 HYPOTHESIS

We first propose that the problem of the discovery of functional protein-protein interactions can be treated as an optimization problem, and as such it can be solved through the use of genetic algorithms. Secondly, we hypothesize that there must be a structured way to not only combine into a single scheme different protein-protein functional prediction methods, but to integrate them into a single searching method that combines each different algorithm individual prowess into a single optimization function. For this, we propose the use of multi-objective genetic algorithms in order to combine the different fitness function that we developed for the first part of the thesis. The result must be an algorithm whose discovery capability is more than that resulting from simply the optimization of overlapping each different method results.

1.4 THESIS OBJECTIVES

The main objective of this thesis is to provide a robust algorithm capable of correctly identifying clusters of proteins which have a functional interrelationship. We understand as functional interelationship a process which requires the participation of a group of many proteins to be completed, and that is extended among a set of species.

In order to extract these characteristics we need to use a computational model that will allow us to do so with a high level of flexibility in being able to easily incorporate very different types of genomic data into a search algorithm in order to increase the reliability and significance of our results. Our proposed solution is a genetic algorithm, in specific a variant based on the Grouping Genetic Algorithms (GGA) introduced by Emmanuel Falkenaeur [12]. Falkenaeur proposed a variant of classic genetic algorithms where the gene structure was optimized for clustering purposes, however Falkenaeur's main application was the optimization of production lines.

We applied a first version using phylogenetic profiles and clustering genetic algorithms that rivaled the results of simple clustering methods, however it failed the encompass information relevant to protein-protein interactions other than phylogenetic profiles, and overall this version of the algorithm, had the tendency to create groups that were too big to be realistic.

Our proposed revision expands on the original by introducing and redefining operators specific for our problem, expanding the landscape so that it is transformed into a multi-objective optimization environment (using phylogenetic profiles, transcription directionality, etc.).

Our results with this expanded version improve significantly over the original, single objective algorithm. Experimental results show that progressively including more and more biological data has a definite impact on the quantity of biological information discovered, not to mention that because of how multi objective genetic algorithms are structured, it allows a higher degree of freedom in terms of selecting what parameter has a certain degree of priority over others.

1.5 THESIS ORGANIZATION

This thesis is organized as follows. In Chapter 2 we will provide a brief theoretical context of the biological and computational terms and topics we will be working on. Then, in chapter 3, our proposed solutions based on parallel grouping genetic algorithms are described. On chapter 4 we will present the results we obtained through the application of our proposed model. Finally, on chapter 5, the future work of this thesis is described and discussed.

2. BACKGROUND

2.1 BIOLOGICAL CONTEXT

Throughtout this section we will shed light on some of the most important Biology principles that will be used in this thesis. The full understanding of these terms will be crucial in understanding not only the concepts we used and applied during our experiments, but also on the importance they have on the grander scheme of Genomics, and thus the importance of this thesis for the area, thus careful reading of this chapter is of utmost importance.

2.2 THE IMPORTANCE OF PROTEINS

The central dogma of biology states (in simplified terms) that within a cell, its DNA will be transcripted to mRNA, that this RNA will find its way to the ribosome, and be translated to a protein to perform a certain function. Once proteins are created they will start performing various functions, like being enzymes and catalyzing biochemical reactions, or providing cell signaling, or even being part of the structure of a cell. Many proteins create more complex structures among themselves that allow them to perform higher level structures, which eventually leads to the formation of living cells and higher level organisms. We can see a graphical representation of this dogma in Fig. 2.1. This thesis work will entirely focus on the protein machine state, more specifically in the interactions resulting from the combination of various proteins.

[1]Image created by Daniel Horspool

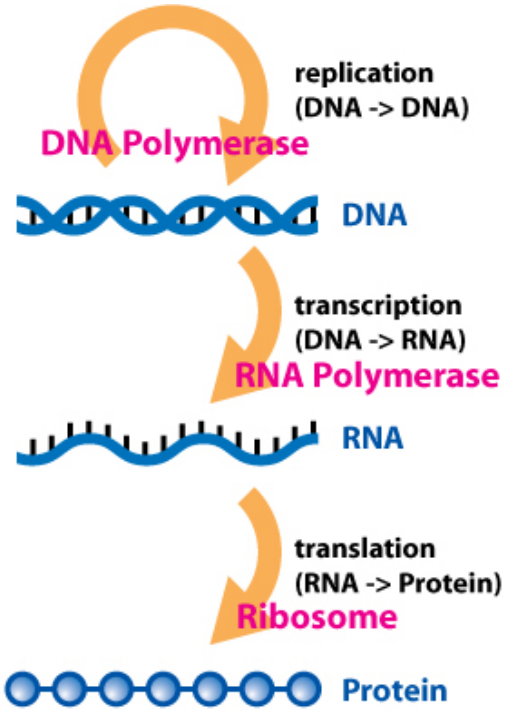

Figure 2.1: Central dogma of Molecular Biology
[1]

2.2.1 PROTEIN-PROTEIN INTERACTIONS

Protein-protein interactions refer to the association of protein molecules and the study of these associations from the perspective of biochemistry, signal transduction and networks.

The interactions between proteins are important for many biological functions. For example, signals from the exterior of a cell are mediated to the inside of that cell by protein-protein interactions of the signaling molecules. This process, called signal transduction, plays a fundamental role in many biological processes and in many diseases (e.g. cancer). Proteins might interact for a long time to form part of a protein complex, a protein may be carrying another protein (for example, from cytoplasm to nucleus or vice-versa in the case of the nuclear pore importins), or a protein may interact briefly with another protein just to modify it (for example, a protein kinase will add a phosphate to a target protein). This modification of proteins can itself change protein-protein interactions. For example, some proteins with SH2 domains only bind to other proteins when they are phosphorylated on the amino acid tyrosine. All in all, protein-protein interactions are of central importance for virtually every process in a living cell. Information about these interactions improves our understanding of diseases and can provide the basis for new therapeutic approaches.

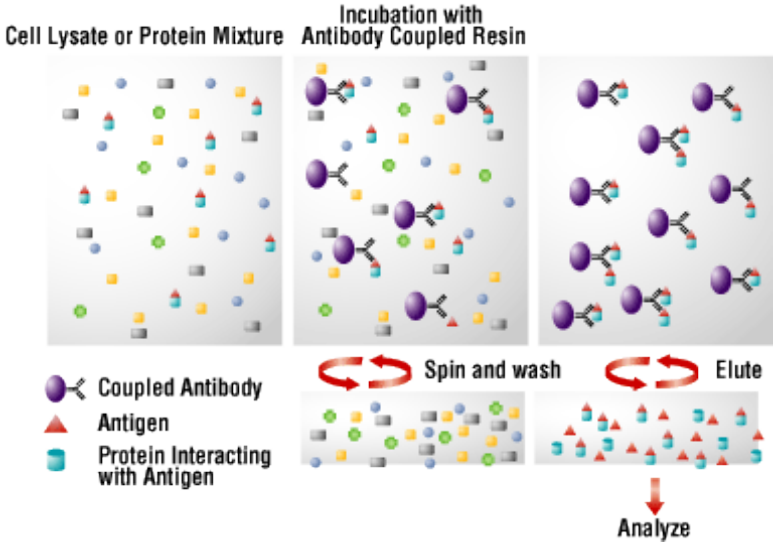

Figure 2.2: Co-immune precipitation
[2]

2.2.2 BIOCHEMISTRY METHODS TO DETERMINE PROTEIN-PROTEIN INTERACTIONS

There is a great variety of methods in the field of biochemistry that have been used to discover these types of interactions, each with their own asset of strengths and drawbacks.

One of the most known is that of Co-immune precipitation[13]. Normal immune precipitation works, like its name describes by using an antibody that specifically binds to a protein that interests us, and helps us to precipitate the antigen of that protein, and thus isolates that protein from a larger group. Similarly, co-immune precipitation follows the same principle, but in this case we are targeting a protein that is believed to be a part of a complex of proteins. In other words, that is part of a larger interaction process, such that when it is precipitated it will carry the whole molecular machine with it. A graphical representation of how this works can be seen in Figure 2.2

All of this are some of the many Biological methods that exist to determine our topic of interest, that is protein-protein interactions. However, as we can see, most of them are built up over the idea that we have a hypothesis of what a possible protein complex would be, and the Biological methods help us to determine the validity of that hypothesis. Obviously, going by mere trial and error is not only extremely time and resource consuming, but extremely prohibitive in terms of the financial resources that need to be invested in each experimental iteration. That is why there has been recent interest from people in different research areas to come up with a set of methods that serve as a guideline, or as a more formal compass of interesting research areas where the methods that we have explained in this section can be applied.

[2]Image created by Daniel Horspool

2.2.3 PREDICTION OF FUNCTIONAL PROTEIN-PROTEIN INTERACTIONS

The prediction of these interactions has been aided with several areas from the knowledge discovery area of computer science, such as data mining, clustering, graph theory and the such. Functional interaction prediction combines the knowledge from fields such as bioinformatics and structural biology in order to determine, identify and catalogue interactions between protein complexes, or even the existence of these complexes themselves. More than a replacement, *in silico* experimentation is a field that serves as a compass and a complement to *in vitro* experimentation.

2.2.4 GENOMIC CONTEXT

Introduction

Understanding the context into which a certain gene that encodes a protein of interest is located is of crucial importance when inferring protein-protein interactions through computational methods [14]

Distance between species

When we compare genetic or proteomic data between any two species, it is important to consider the genetic distance and evolutionary history between them. By this, we mean that similar sections in dissimilar species are more meaningful from an evolution point of view than similar sections in closely related species. This is because the genome of these species hasn't suffered as much evochanges as those of other species, and similar sections between the genomes could merely be some kind of coincidence. In contrast, species that belong to different *genus*, or even different families tend to vary in large portion of their genome. However different analysis show that there are sections that are conserved almost *verbatim* between these species. Moreover, many of these sections have been found to encode protein that perform vital functions in the organism, genes that are so crucial that not even millions of years of divergent evolutionary history have been able to purge. From this stems that if a set of proteins remain present in a vast array of species, the possibilities that there is an underlying reason behind that are high. Most likely a functional interaction.

Direction and adjacency of transcription

According to Biology's central dogma, when the DNA is transcripted to mRNA it searches for the specific region that will be converted, and does it in a single direction by searching for a START and TERMINATE signal molecules. As such, it is the case that when a protein pathway function is to be performed normally we have that the protein belonging to this pathway will be transcripted in the same direction, and they most probably will be located close to each other. Many evolutionary proven protein-protein interactions, like the *purEK* operon shown in Fig. 2.3

Figure 2.3: Interacting proteins are near each other, and transcribe in the same direction

Table 2.1: Example of phylogenetic profiles

Protein	g_1	g_2	g_3	g_4	g_5
p_1	1	1	0	1	1
p_2	1	1	1	0	1
p_3	1	0	1	1	1
p_4	1	1	0	0	0
p_5	1	1	1	1	1
p_6	1	0	1	1	1
p_7	1	1	1	0	1
p_8	1	0	0	1	1

show that this assumption is consistent with reality, and as such is a sufficiently strong method to predict protein-protein interactions

2.2.5 PHYLOGENETIC PROFILES

Phylogenetic profiles describe patterns of presence-absence of proteins in a collection of organisms. The construction of phylogenetic profiles[15] begins with a collection of k completely sequenced genomes G from different organisms and a collection of l proteins P of interest. For each protein p_i, a phylogenetic profile is represented as a k-length binary string $s = s_1 s_2 \cdots s_k$ where $s_j = 1$ if protein p_i is present in genome g_j and $s_j = 0$ if protein p_i is absent in genome g_j.

Table I shows an example of the construction of phylogenetic profiles. In this example, phylogenetic profiles are constructed for 8 proteins $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$ indicating their presence (or absence) in 5 genomes $(g_1, g_2, g_3, g_4, g_5)$ from different organisms. Note that all proteins $p_1, \ldots, p_8$ are present in genome g_1.

Functional coupling of proteins is then inferred by clustering proteins according to the intrinsic similarities of the underlying phylogenetic profile patterns. It is often concluded that proteins associated to the same cluster are functionally related. For the example shown in Table 2.1, a potential functional association between proteins p_2 and p_7 would be identified by this method as they posses identical phylogenetic profiles.

The logic underlying this reasoning is that proteins with similar phylogenetic profiles are likely to interact in performing some biological process. In effect, there should be an evolutionary pressure acting on a group of proteins in order to preserve a function that confers an advantage to the organisms.

2.2.6 OTHER *IN SILICO* METHODS FOR DETECTING PROTEIN-PROTEIN INTERACTIONS

There are many other methods apart from the ones we have mentioned in this section that have been used with good results in the Computational Biology community. One of such are the so called voting methods, like the STRING repository[10]. This type of methods attempt to create a framework in which several different approaches are integrated into a single scheme. For this, all of the 'sub methods' output is converted into a binary entry, a vote of sorts is given to each method. After that, it becomes a weighted sum of each proposed relationship, and the resulting $interactome$ is built from that. As we can see, there are some problems associated with this method. First, by restricting the votes of each procedure to be binary we are obviously losing information. For example, if we have a method that returns values between the [0,1] range, we have to determine a certain threshold that tell us from which point it becomes 1 or 0. Another obvious problem that arises from this is that either we will have to set the threshold values in a more or less arbitrary way, or we will have to resort to yet another algorithmic layer in order to determine the best possible value for this cutoff.

2.3 COMPUTATIONAL CONTEXT

For the analysis of the biological input data we received, we used two distinct techniques from the Evolutionary Computation corpus. One is a variant of classic genetic algorithms called Grouping Genetic Algorithms, which we used to discover the functional clusters proteins are divided into. The other one was multiobjective genetic algorithms, through which we combined the different input data we received into a single schema, in order to optimize the search process. We will start this section by explaining how a classic genetic algorithms works, and then proceed to explain the variants we used.

2.3.1 SIMPLE GENETIC ALGORITHMS

Genetic algorithms can be considered as a search technique whose algorithm is based on the mechanics of natural selection and genetics. It has been successfully used in realms as diverse as search, optimization, and machine learning problems, since they are not restricted by problem specific assumptions such as continuity, existence of derivatives or unimodality.[16]. In rough terms, a genetic algorithm creates a collection of possible solutions to a specific problem. Initially those solutions are tipically randomly generated solutions, so their initial performance is normally poor. However, no matter how bad, there will be small segments of our collection of solutions that will be nearby our desired solution, that is, partially correct answers. Genetic Algorithms exploit this characteristic by recombinating and progressively creating better solutions, so that by the end of the run we have achieved one solution that is at least nearly optimal.

Although Genetic algorithms have greatly diverged since their original inception into a big number of different variants like Genetic Programing, Evolutionary Computation, Evolutionary Strategies and the such, the basic principles that all share can be described as follows:

- An encoding: Genetic algorithms, in most cases, don't directly work with the parameters of the underlying problem, instead those are encoded into a way particular to our problem. Such an encoding is called a *Gene*. Genes themselves are arranged into chromosomes, which represent a possible solution to a problem. Chromosomes are grouped into a populations, which is a set of solutions the program presents to a problem. Through different operations, the algorithm keeps improving those solutions until an appropriated one is found.

- A Fitness function: Rather than using derivatives or other auxiliary knowledge, the fitness function is used as a measure of the "survival strength" of the individuals of our population[17]. In more concrete terms, a fitness function will evaluate an specific condition we want our solution to meet. For example, in a maximization problem, the best solutions would correspond to the parameters that gave provide highest objective function values.

- A reproduction scheme: It refers to a mechanism that tells us how to select the members of a population that will make it to the next generation. It is not simply a matter of selecting the best individuals, but also of maintaining variability in order to keep a broad search scope and avoid local optima, while at the same time identifying interesting search spots and exploiting them.

- A recombination scheme: One of the cores of a genetic algorithm, which basically tell us the process that we will use to combine the different solutions into a new solution. This can be achieved by recombining two solutions (crossover), tweaking a single solution (mutation) or by some other scheme.

2.3.2 THE GROUPING GENETIC ALGORITHM

As mentioned earlier, the Grouping Genetic Algorithms (GGA) is a technique developed by Emmanuel Falkenaeur that stemmed from the fact that traditional GA's performed poorly when applied to grouping problems. Falkenaeur proposed a design where the chromosomes were not a set of elements which conformed a solution, but that the genes themselves represented the groups, and as such the chromosomes were a collection of groups, instead of directly containing the elements. As such since the base structures are quite different, the crossover and mutation operators vary significantly compared to those of Simple Genetic Algorithms. We will go in greater detail of our implementation in Section 3.3

2.3.3 MULTI-OBJECTIVE GENETIC ALGORITHMS

Classic Genetic Algorithms, as defined by Holland[18] consider a single objective function to use. (That is, if there are multiple objectives to optimize they must be collapsed to a single objective function). While this is useful for a limited set of problems, a larger collection of the problems that can be found in nature are restricted by multiple constraints, or must meet a diverse number of requirements at the same time. At the same time, multi-objective problems have always presented

one of the greatest challenges for algorithm designers. Because of this, alternative approaches, such as those presented by Multi-Objective Genetic Algorithms have gained much interest in recent years. [19].

Once all of this is said and done, the problem of how to use multiple fitness functions into a single algorithm remains. Multiple solutions have been designed, however they can be grouped into two main classifications: *a priori* techniques and *a posteriori* techniques. The former ones need to have the importance of our objectives defined before the search is undertaken. These include linear aggregations techniques, lexicographic techniques (which give a priority to different objectives) and so on. These techniques have been shown to be weak on multiple comparison works, since they require a predetermination of the objective priority, which means we are more or less arbitrarily limiting the search space, and as such we cannot find all the optimal solutions for our problem. On the other hand a posteriori techniques implicitly searches for the solutions that optimize all of our objective functions. There are multiple examples of these kind of algorithms, however the ones who have been reported to give the best results [19] and have a more widespread used are the Pareto Genetic Algorithms family (like the Nondominated Sorting Genetic Algorithm(NSGA)[20], or the SPEA2 by Zitzler[21]). In this work, we will proceed to use one of the variants of the algorithm, known as the Niched Pareto Genetic Algorithm.

Pareto optimum

When working with multi-objective systems, the concept of optimum changes because we have to find good compromises or trade offs among our objectives. For example, not all of our objectives may be directly linearly related, or they may actually be contradictory, yet we still have to maximize(or minimize, depending on the problem) both of them as much as possible.

The most commonly adopted definition of optimality is the one proposed by Vilfredo Pareto in 1896, that tells us that the vector or chromosome $\vec{x}^*$ is Pareto optimal if there exist no feasible vector $\vec{x}$ within our population that would decrease some criterion without causing a simultaneous increase in at least other criterion.

We also need to introduce the definition of Pareto Dominance. A vector $\vec{u} = \{u_1, ..., u_k\}$ (where each member is a particular fitness value) is said to dominate another vector $\vec{v}$ when $\vec{u}$ is partially less than $\vec{v}$, that is, at least one the members in $\vec{u}$ is strictly less than its equivalent member in $\vec{v}$ (for a minimization problem) and the rest of the members in $\vec{u}$ are equal or less than their counterparts.

Conversely, Pareto non-dominance of a vector $\vec{u}$ to a vector $\vec{v}$ means that $\vec{v}$ does not dominate $\vec{u}$,or in plain words, we cannot find a value in $\vec{v}$ that is less than a value in $\vec{u}$ without also finding another fitness in $\vec{v}$ that is bigger than its $\vec{u}$ counterpart.

Over these definitions we create the Pareto Set (P^*) which is the collection of all the vectors in our population that are non-dominated by the rest of the population.

The evaluated objective vectors of P^* build up what is known as the Pareto Front (PF^*), geometrically the PF^* can be defined as the boundary of the design region, or in the locus of the tangent points of the objective functions. [19]

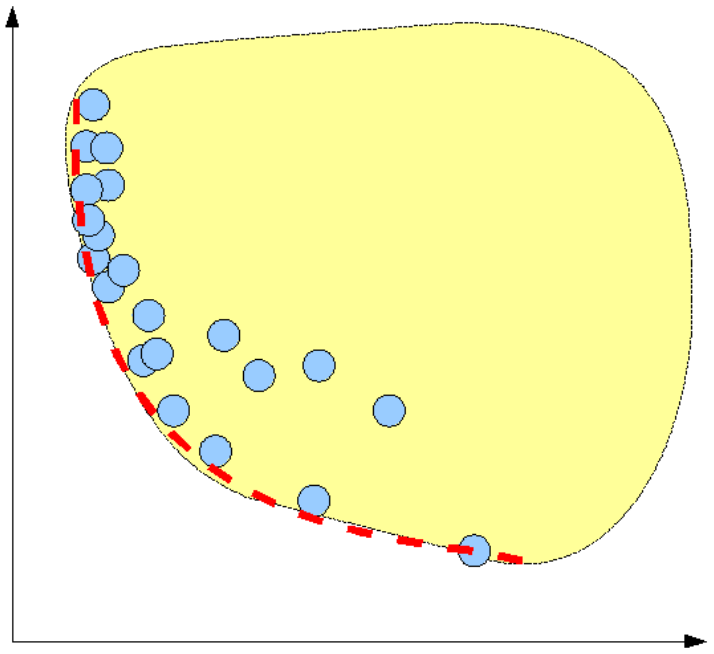

Figure 2.4: Example of a Pareto front. The yellow region represents the feasible area. The red line represents the pareto front. The blue points represent the current population

All these concepts are best exemplified in Fig. 2.4. This is an hypothetical minimization problem limited by two constraints, shown by axis X and Y. The yellow area represents those points that meet with the minimum limitations imposed by our constraints (in other words, any point that appeared in this area would be an unfeasible solution). The points represented by blue circles represent the solutions in our current population. Since this is a minimization problem, we say that the points that are furthest from the red dotted line are strictly Pareto dominated by the points that are over this read line, given that this circles represent a better minimization solution than the others. On the other hand, all of the circles that are over the red line are non dominated by the rest of the population. The set formed by our non dominated solutions is the Pareto set. The red line represent the set of solutions towards which all of our population should eventually converge, the Pareto front of our solution set.

Niching techniques

Niching methods are a series of techniques developed to maintain diversity within a population. It is often the case that when the Pareto front is being built, the population has an early convergence towards some zone of the Pareto front, leaving whole regions untouched. Obviously this goes against the spirit of generating the whole Pareto set, so the niching methods were introduced as a set of techniques that aim to maintain diversity. [22]

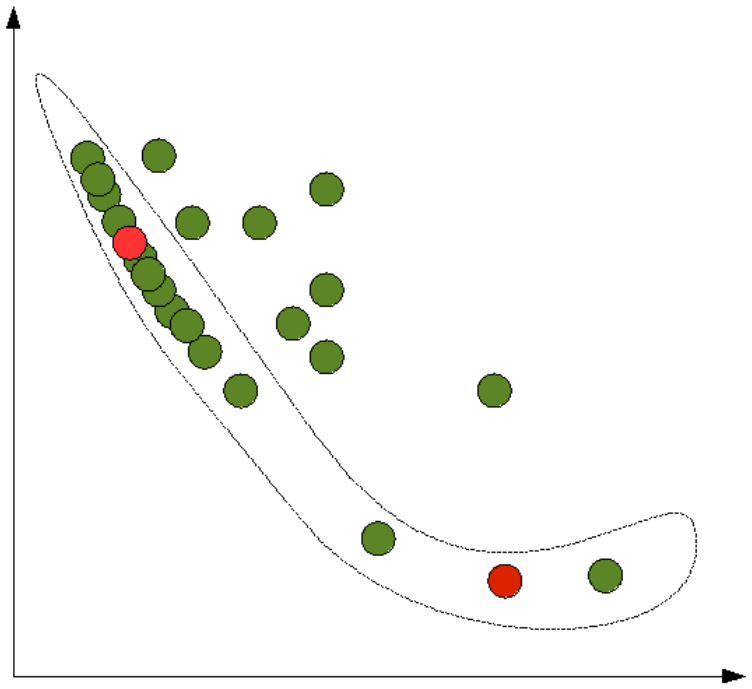

Figure 2.5: Example of an unevenly distributed Pareto front

Fitness sharing

Fitness sharing refers to a technique in which we analyze the placement of solutions (e.g. in objective function space) and penalize those located in the most crowded areas. By doing this we avoid having all the chromosomes in one area and ensure variability. We do this by applying formula 2.1 where d_{ij} refers to the distance between chromosomes *i* and *j*, and σ_{share} refers to the distance threshold, a parameter we introduce to the algorithm that defines what is the maximum distance for two chromosomes to be considered as part of the same niche. We apply this formula to our candidates, and the one which has the smallest fitness is located in the most crowded area, and as such is the one that holds the lowest exploratory value. This is best exemplified in Fig. 2.5. In this figure we have a Pareto front, represented by the zone enveloped by a dotted line, and two candidates for our next generation, represented by the red circles. As we can see, the left candidate is in a more crowded area when compared to the right candidate. By choosing the right candidate over the left one, we maintain more variability within our population, and diminish our possibilities of falling into a local minima.

$$\sum_{i=0}^{N} sh(d_{ij}) \tag{2.1}$$

$$sh(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_{share}) & d_{ij} \leq \sigma share \\ 0 & otherwise \end{cases} \tag{2.2}$$

The niched Pareto genetic algorithm

The niched Pareto algorithm combines both a Pareto optimum next generation construction with a fitness sharing scheme to maintain variability. The version we use is the one described in [23], a niched tournament Pareto genetic algorithm. We will go in greater detail about how do we implement this algorithm in a further section when we explain our work. However in a broad sense we use a tournament scheme in order to select those members of the current population which will pass to the next population. The winners are selected by choosing those who are non dominated with respect to their competitors against a sample subset of the population. In case we find that the competitors are tied, we use a fitness sharing scheme to select those that will pass to the next generation. Through this method we can progressively construct better Pareto fronts.

As a first approach, we selected this method, which is considered of first generation over other more recent alternatives because of its ease of implementation. In order to cover its lack of an elitism scheme, we introduced a separate elitism function that selected the best members of each individual fitness function, and the solution that best balanced the different fitness functions in our algorithm.

2.3.4 USING GENETIC ALGORITHMS AS A CLUSTERING METHOD IN BIOINFORMATICS

Genetic Algorithms have been previously used as an aid in different clustering processes to automatically determine many of the parameters these algorithms need. One of the most common uses is to calculate the number of centroids an algorithm will work with, or helping the algorithm to avoid local minima.[24] However the use of Genetic Algorithms as a clustering process in and of itself has only raised interest in recent years.

Grouping Genetic Algorithms were first developed by Falkenauer in 1998 when applied to the optimization of production lines, and from then it has been applied to multiple areas, because of its ability to seemingly find the optimal clustering of landscapes where classic clustering algorithms have serious problems to work with great success, like Intrusion Detection Systems[25] and Control systems[26], however its use in bioinformatics is scarce at best.

On the other hand, Multiple Objective Genetic Algorithms have seen a wider use in Bioinformatics, with many successful applications in areas such as System Optimization [27, 28], and inverse problems[29]. However its combination with Grouping Genetic Algorithms hasn't seen much development. One of the first approaches was proposed by Deb[30], when he applied applied a multi-objective optimization algorithm in order to classify cancer data. Deb reported that he had obtained satisfactory results in obtaining optimal clusters with minimal false positives and sizes. Similar approaches have been tried more recently by Facelli and others[31].

However, in those works the multiple objective functions have only been used as a measure to introduce multiple computational clustering characteristics into the classification process. Deb used fitness functions that minimized both the cluster size and the false positives. Although the introduction of such data is of great help for the classification process of the proposed data, we believe that it would also be interesting to introduce various biological data in the form of different

fitness functions in order to not only increasing the computational congruence of our clusters, but also of their biological relevance.

In other words, while there have certainly been attempts at introducing techniques from both the clustering and evolutionary computation community in order to mine biological data, it has always been used from a strictly computational perspective. The different fitness functions and clustering factors that previous authors have applied do not take into account the great diversity that biological input data brings to a problem. While this certainly is a challenge in and of itself, it also represents a vast horizon of data that would allow us to extract a lot of information if we could combine the different input parameters into a single data mining process.

2.4 K-MEANS

In this section we will proceed to briefly explain what is and how does the k-means method works, which is the reference method we used. K-means is one the most widely used clustering methods because of its easiness of implementation and wide use. The algorithm was first published by Lloyd in [32], and since then it has become one of the staple algorithms in the community, and as such it is a good reference algorithm for any new algorithm to be benchmarked against.

In broad terms, k-means is a partitioning algorithm which aims to make k clusters out of n observations, in which each point belongs to the cluster with the nearest mean. Hence, we possess k means. We can find the pseudocode of the algorithm in 1. As we can observe from the pseudo code, the algorithm is very simple, however it has been found to be one of the most flexible clustering algorithms among all the options available.

Algorithm 1 k-means

Require: for n observations and k target clusters

```
Randomly initialize the k centroids position.
while algorithm has not converged do
  (assign each observation to the nearest mean)
  (move each mean location to the calculated average of each
  centroid)
end while
return  population
```

3. PROPOSED MODEL

In Tapia and Vallejo[9] we presented an initial approach to obtaining protein-protein interaction complexes. In this section we will proceed to explain the framework we have developed for this endeavor.

3.1 THE INPUT DATA

3.1.1 THE COG DATABASE

One of the most well known Phylogenetic Profiles databases is provided by the Cluster of Orthologous Groups of proteins[33], which we have previously used coupled with other clustering algorithms with great success[8, 7]. This database consists of a collection of conserved protein families (COGs) that are presumed to be orthologous.

Particularly, we relied on the 43 completely sequenced genomes COG initial version, in which phylogenetic profiles are represented as 26-length binary strings. These genomes come from three different domains: 6 *archaea*, 19 *bacteria* and 1 *eukaryota*. In constructing phylogenetic profiles, 43 genomes were collapsed into 26 representative genomes. The reason was that some of these genomes are very similar to each other or they belong to a subspecies of other organism represented in the database. The organisms represented in the COG database are listed in Table 3.1.

The number of that can be represented by a 26-length binary vector is $2^{26} = 67,108,864$. This number is far greater than the number of known proteins. The COG database provides a collection of 3,307 phylogenetic profiles, which are fed into our clustering algorithm.

Table 3.1: Genomes represented in COG database (part I)

ID	Description	Domain
A	*Archaeoglobus fulgidus*	archaea
O	*Halobacterium sp. NRC-1*	archaea
M	*Methanococcus jannaschii*	archaea
	Methanobacterium thermoautotrophicum	archaea
P	*Thermoplasma acidophilum*	archaea
	Thermoplasma volcanium	archaea
K	*Pyrococcus horikoshii*	archaea
	Pyrococcus abyssi	archaea
Z	*Aeropyrum pernix*	archaea
Y	*Saccharomyces cerevisiae*	eukaryote
Q	*Aquifex aeolicus*	bacteria
V	*Thermotoga maritima*	bacteria
D	*Deinococcus radiodurans*	bacteria
R	*Mycobacterium tuberculosis*	bacteria
	Mycobacterium leprae	bacteria
L	*Lactococcus lactis*	bacteria
	Streptococcus pyogenes	bacteria
B	*Bacillus subtilis*	bacteria
	Bacillus halodurans	bacteria
C	*Synechocystis*	bacteria
E	*Escherichia coli K12*	bacteria
	Escherichia coli O157	bacteria
	Buchnera sp. APS	bacteria
F	*Pseudomonas aeruginosa*	bacteria
G	*Vibrio cholerae*	bacteria
H	*Haemophilus influenzae*	bacteria
	Pasteurella multocida	bacteria
S	*Xylella fastidiosa*	bacteria
N	*Neisseria meningitidis MC58*	bacteria
	Neisseria meningitidis Z2491	bacteria

Table 3.2: Genomes represented in COG database(part II)

ID	Description	Domain
U	*Helicobacter pylori 26695*	bacteria
	Helicobacter pylori J99	bacteria
	Campylobacter jejuni	bacteria
J	*Mesorhizobium loti*	bacteria
	Caulobacter crescentus	bacteria
X	*Rickettsia prowazekii*	bacteria
I	*Chlamydia trachomatis*	bacteria
	Chlamydia pneumoniae	bacteria
T	*Treponema pallidum*	bacteria
	Borrelia burgdorferi	bacteria
W	*Ureaplasma urealyticum*	bacteria
	Mycoplasma pneumoniae	bacteria
	Mycoplasma genitalium	bacteria

3.1.2 THE GECONT DATABASE

For our experiments we used a gene database provided by the Biotechnology Institute from the National Autonomous University of Mexico (UNAM). This database was originally used as a backend for GeCont[34], a tool that graphically displays the gene context of a certain protein the user is interested in. As such, the database is quite rich in terms of the information we need, such as including the proteome of different species, starting and ending nucleotide, scientific name, corresponding number ID from the COG database etc. We used this database as our source of information for the Genomic Directionality and the Intergenic Distance fitness functions.

3.1.3 OTHER DATABASES

Although the databases we mentioned above where the ones we fed the algorithm, we used other databases as reference values for our benchmarking process. They were mainly the DIP and Ecocyc databases[35, 36], and a series of operons found in the *E. Coli.*

The DIP and Ecocyc databases consist of a collection of about 350 protein pairs whose functional relationship has been experimentally proven. Although they provide only for a small subset of true positives of what represents a much bigger interaction network, it is a necessary first step stone in our algorithm.

Similarly, an operon database was also used as in reference [37]. Operons can be defined as a functioning unit of adjacent nucleotide sequences including an operator, a common promoter, and one or more structural genes, which is controlled as a unit to produce messenger RNA (mRNA), in the process of transcription. They are normally found on prokaryotes, which is quite convenient when we consider that 80% of our reference database is formed up by such type of beings.

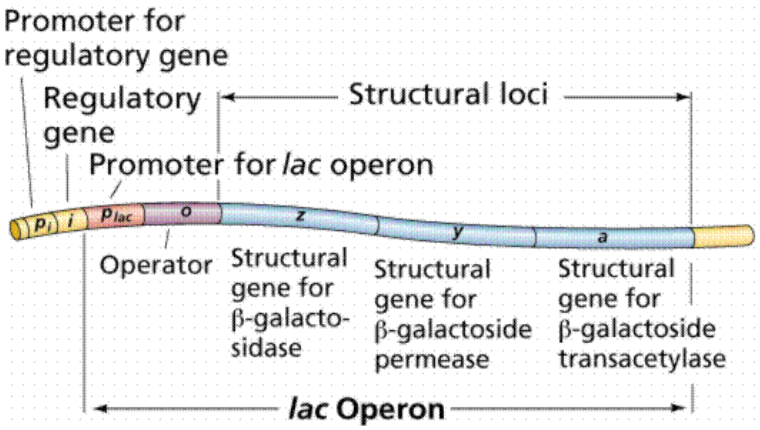

Figure 3.1: The *lac* operon
[1]

An example of one of the most common operons, the *lac* operon is illustrated in Fig. [38]. In consequence, an apt grouping mechanism would have to group together this patterns, and hence it becomes a good way to our validate our data.

3.2 PREPARING THE DATABASES

In the following section we proceed to explain the different pre processing steps we applied to the raw databases as we received them. In general, the databases that we had can be divided into two groups. Those that required little preprocessing and which were simple enough in nature, like the COG, DIP and Ecocyc databases, and those whose structure was too complicated to be directly treated (either because of their structure, their size, or both), and that as such were parsed and stored into a proper database. Examples of these are the Gecont database, and the *E. Coli* database.

3.2.1 THE COG DATABASE

The COG database possessed a relatively simple structure when compared to the other databases, as exemplified in table 3.3. It basically consists of two fields, the first being a binary string containing the phylogenetic profile, and the second being an identifier for the COG. Given the simplicity and small size of these database, we decided to keep it as a text file. While certainly we are not keeping everything to a single database, this scheme allows to perform an asynchronous loading stage, in the sense that we can create multiple threads all loading the data at once from our different data sources, instead of overloading a single database server, which effectively diminishes our loading stage computation time.

[1] Image taken from Mike Farabee's webpage

Table 3.3: Example of an entry of the COG database.

00000011111110111101000000	COG1490
10111101110010110000010000	COG1514
01011010111110111101111001	COG1670

Table 3.4: Example of an entry of the GeCont database. Some fields are ommited

190..255	+	21	16127995	thrL	-	thr operon leader peptide
337..2799	+	820	16127996	thrA	COG0460E,COG0527E	fused aspartokinase
2801..3733	-	310	16127997	thrB	COG0083E	homoserine kinase
3734..5020	+	428	16127998	thrC	COG0498E	threonine synthase

3.2.2 THE GECONT DATABASE

The GeCont database, as we obtained it was composed of a series of plain text files, where each file contained the codified genome of a different species. The format each file came with was similar to the one show in Table 3.4, which is, in order

- Nucleotide where the transcription starts and ends
- Direction of the transcription, where - stands for left and + for right.
- Number of nucleotides without introns.
- Catalog number
- Protein short name
- List of COGs associated to this protein.
- Complete name of the transcripted protein.

As we can see from that table the information does not follow the 4-normal form of relational databases, furthermore it is not directly suited as input data for our algorithm, and as such it needs some preprocessing.

What we did was parsing the data using a Java program, and outputting the data on a SQL script file, to later be inserted into a MySQL database as shown in the entity relationship diagrams in Tables 3.5 and 3.6. (We needed to separate the COG information so that the data was normalized.)

This is how we initially loaded our data. Our main program would then connect to the database and download to the program local memory space this information on demand, specially the relationship between COG's and their relative context. However, it is obvious that although we will only perform this data exchange once, when we are downloading the data the database server will have to perform one major join operation between the two tables (considering they have more than

Table 3.5: The context table entity relationship model

Field	Type	Null	Key	Default
species	varchar(40)	YES		NULL
direction	varchar(1)	YES		NULL
initialNucleotide	int(11)	YES		NULL
finalNucleotide	int(11)	YES		NULL
lenght	int(11)	YES		NULL
serialNumber	int(11)	NO	PRI	0
name	varchar(10)	YES		NULL
alternativeName	varchar(15)	YES		NULL
completeName	varchar(30)	YES		NULL

Table 3.6: The Gecont-COG table ER model

Field	Type	Null	Key
ID	int(10) unsigned	NO	PRI
serialNumber	int(11)	NO	MUL
COG	varchar(10)	NO	

3.5 million registers between the two, it is a major, and to a certain level unnecessary operation). That is why we defined a third virtual table that, while it is not normalized, it is optimized in other respects as to only contain the information that we will use in this impĺementation in a readily and easily available way. The model is shown in Table 3.7. In this table we only included identified proteins whose COG is included in the database, and we trimmed down the species to only those that also appeared in our phylogenetic profiles database. This resulted in about 100000 entries.

3.2.3 THE DIP AND ECOCYC DATABASES

The DIP and Ecocyc databases were two databases possessing a simple structure. They have two fields, one which is the group number, and a second which is the COG name, and it is organized in pairs, representing experimentally proven protein-protein relationships. An example of how it is organized can be seen in Table 3.8

As mentioned earlier, these databases were just inputed as a text file, which was parsed by the preprocessor of the genetic algorithm, and loaded as an internal array for use by the profiler engine.

3.2.4 THE E. COLI OPERON DATABASE

The E. Coli operon database as we used is a repository available publicy online in

Table 3.7: The context table entity relationship model

Field	Type	Null	Key
ID	nt(11)	NO	PRI
species	varchar(30)	YES	
direction	varchar(5)	YES	
alternativeName	int(11)	YES	
COG	varchar(10)	YES	
initialNucleotide	int(11)	YES	
finalNucleotide	int(11)	YES	
completeName	varchar(30)	YES	

Table 3.8: An example of the DIP table

Group Number	COG
4	COG0055
4	COG0056
5	COG0055
5	COG0224

`http://www.cib.nig.ac.jp/dda/taitoh/ecoli.operon.html`

In order to use it, we had a parser download the information from the page, and parse the HTML table format into an internal array of memory. Given that the number of operon groups is relatively large, we also parsed it as a MySQL database for use in the algorithm proper (as we mentioned earlier, the E. Coli operon database was used as a reference database in order to test the effectivity of the algorithm). The E-R relationship model of this database can be found in Table 3.9. The list of which cluster corresponded to what operon was kept in a separate index table, although for the statistical usage that was given to this database it was not necessary to use.

Table 3.9: Example of early clustering genetic algorithms

Element	**Cluster**	Key
id	int(10) unsigned	PRI
cluster	int(10) unsigned	
cog	varchar(10)	

Table 3.10: Example of early clustering genetic algorithms

Element	Cluster
e_1	g_1
e_2	g_3
e_3	g_4
e_4	g_2
e_5	g_1
e_6	g_3
e_7	g_2
e_8	g_1

3.3 THE GROUPING GENETIC ALGORITHM

Genetic algorithms can be defined as a search technique whose algorithm is based on the mechanics of natural selection and genetics. It has been successfully used in realms as diverse as search, optimization, and machine learning problems, since they are not restricted by problem specific assumptions like continuity, existence of derivatives, unimodality and similar.[16]. In rough terms, a genetic algorithm searches a landscape of possible solutions to a specific problem. Initially those solutions are typically generated at random, so they will not perform well. However, no matter how bad, there will be small segments of our solution collection that will be near our desired solution, that is partially correct answers. Genetic Algorithms exploit this characteristic by recombinating and progressively creating better solutions, so that by the end of the algorithm we have achieved one solution that is near or is what we seek for, the optimal solution

However, when they were first applied to clustering, Genetic Algorithms did not perform as well as they had done for other type of optimization problems. In the first proposed approach, the design was similar to that shown in Table 3.10. For a group of elements e_k — $k = 1..8$ and clusters g_i — $i = 1..4$, each gene was represented by an element-group pair, expressing which groups each element belongs to.

However, this type of representation was found to have a very poor capability of converging into useful solutions. Structures and clusters where often eliminated by the way the search space is explored by Genetic Algorithms. As a response to this, Emmanuel Falkenauer developed the Grouping Genetic Algorithms (GGA)[12]. Falkenauer proposed a design where the chromosomes were not a set of elements which conformed a solution, but that the genes themselves represented the groups, and as such the chromosomes were a collection of groups, instead of directly containing the elements.

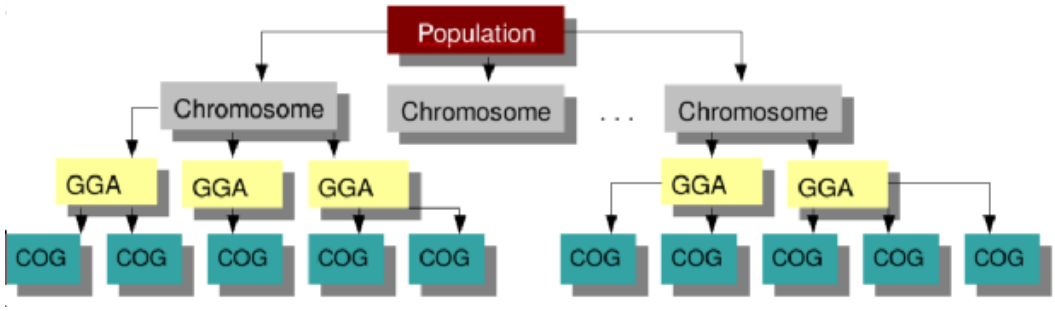

Figure 3.2: Structure of a population

3.3.1 BASIC STRUCTURES

Figure 3.2 depicts the hierarchical structure of the genetic algorithm we used. Although it is remarkably similar to that of Simple Genetic Algorithms (GA), there are some differences that are worth noticing.

Population: The population structure that was used, as with GGA, is a collection of solutions our current generations is comprised of. It takes a series of COG's to be sorted and the fitness functions data to be used as an input. It is important to note that the algorithm does not need to be fed the number of groups that the COG's will be sorted into as an input, the algorithm will determine that by itself, this fact gives it an important edge over other classic grouping algorithms. The initial population is constructed from a random distribution, and further populations are the product of applying evolution operators to the previous ones.

Chromosome: As we mentioned earlier, a chromosome within the GGA context refers to a proposed arrangement of the elements into different clusters. Other than sharing the elements they arrange, each clusters has its own number and size of clusters.

Group Genes: A group gene in our algorithm is defined as a limited collection of COG's (a cluster), and a set of special structures called the Consensus Structures (CoSt), which are the centroids of the members of a group according to the different fitness structures that we will be evaluating as described earlier.

3.3.2 GENETIC OPERATORS

Traditionally Genetic Algorithms exploration and exploitation mechanisms can be summed up into two genetic operators: *Crossover* and *Mutation*. Crossover in clustering genetic algorithms is normally done by exchanging clusters between different chromosomes, and rearranging the possible inconsistencies afterwards (Members that are in more than one cluster, or missing members). We

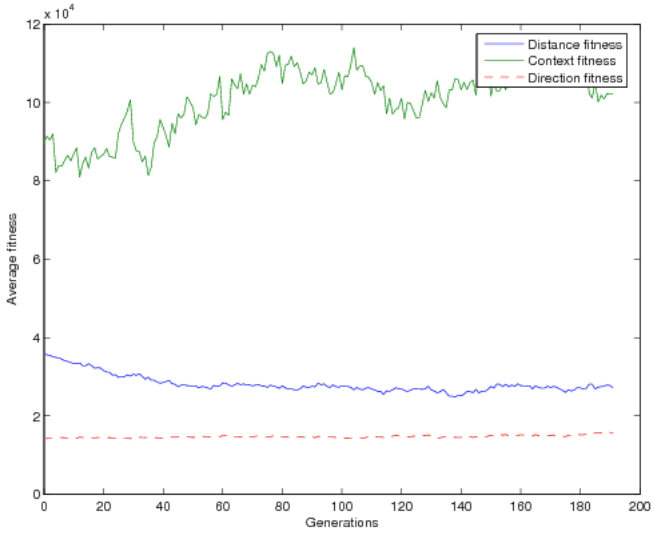

Figure 3.3: Behavior of the fitness functions using crossover and mutation

performed a comparison study with and without crossover across a series of 10 executions of the algorithm over 192 generations and calculated the average evaluation for each generation under our three biological fitness functions, the results are shown in the Figures 3.3 and 3.4. As it can be seen in our particular problem crossover does not effect noticeable enough exploratory capabilities; on the contrary, our validation process (that we will later explain) showed that more than often it was disruptive of the biological significance of the data, and as such it only slowed down the convergence process, since crossover had a computational complexity greater even than the evaluation process. Moreover, the computational complexity of the crossover operation was greater than any of its mutating counterparts. As such we decided to use only the following mutation operators.

- $RandomNewGroup$: Grabs random members from different groups and creates a new group with them.
- $RandomDeleteGroup$: Selects a random group and disperses its members through different groups in order to remove it.
- $MergeStrong$: Searches two groups whose centroids (according to a random fitness function) are very close in terms of the distance between them, and them proceeds to merge them into a single group, similar to what is done in Fig. 3.5
- $RehashWeak$: Searches for one group whose variability between its members is considerably greater (surpasses some threshold) than the average of the whole chromosome, and scatters its members across the rest of the clusters within one chromosome. An example can be seen in Fig. 3.6
- $Weeding$: Searches for outliers (according to a random fitness function) within a group and reallocates them to a group whose centroid is reasonably near from the outliers, like what is

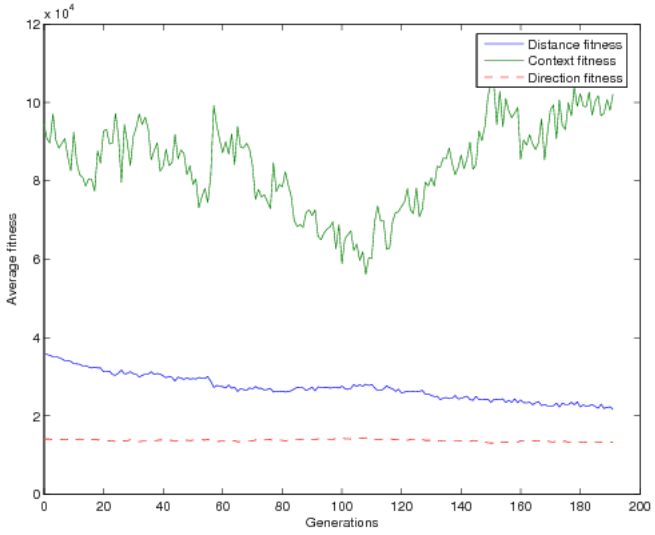

Figure 3.4: Behavior of the fitness functions using mutation only

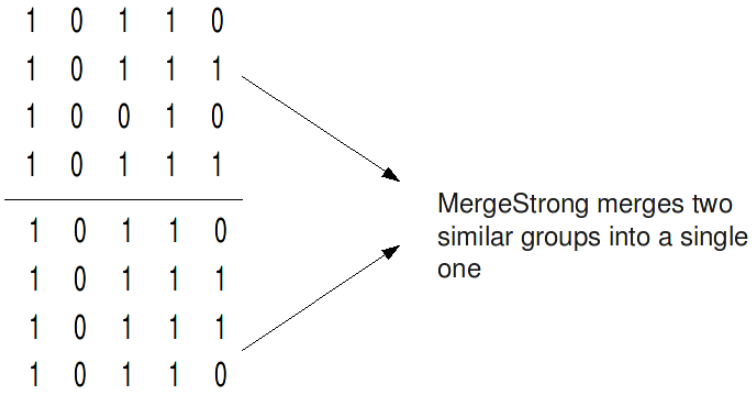

Figure 3.5: Example of how the `MergeStrong` operator works

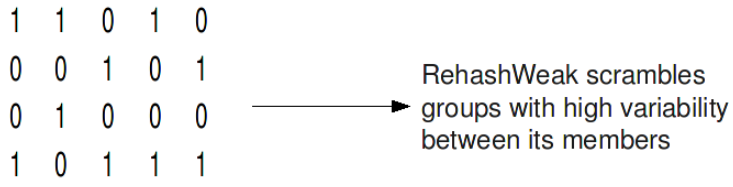

Figure 3.6: Example of how the `rehashWeak` operator works

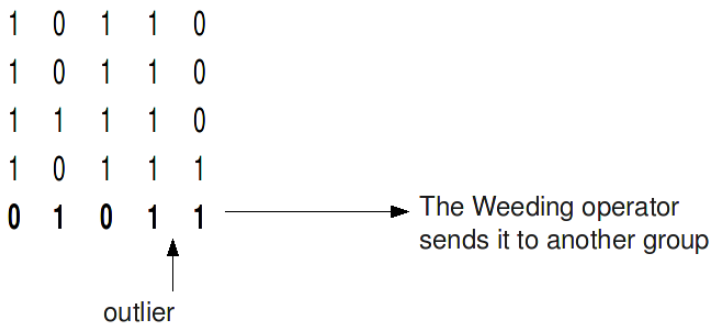

Figure 3.7: Example of how the `weeding` operator works

shown in Fig. 3.7

Normally the mutators are preferred to be entirely random in its application since theoretically this maximizes the search space exploratory capabilities of Evolutionary Algorithms. However, here we decided to introduce directed mutators that worked for specific cases, since there are specific regions from the search space that we are interested in mining. That is, we are further increasing the search capabilities of the algorithm into making it specifically explore the regions we are interested in.

3.3.3 USING A SINGLE EVALUATION FUNCTION

As it was explained before, the initial population will be constructed from an uniform distribution. Once the first generation was generated , a series of evaluation functions were applied. Which and how many evaluation functions should be used was determined after a number of empirical tests in which various techniques were tried, included but not limited to, doing a *n* to *n* correlationship comparison of the different species all the COG's within a group gene appeared in, or creating an artificial type of COG comprised of the average of all the COG's included in a group, and compare the COG's against that average.

The best metric that was found was creating a Cost akin to that used in motif analysis. The fitness operation would then only be applied n times (one for each member of the group against the Cost) reducing the evaluation time by one polynomical order. This also proved having a high performance and efficiency in doing an exploratory work across the search space, not showing any significant inferiority to doing a COG per COG comparison. An increasingly bigger bonus and penalization was given for every COG that was sufficiently near (or far) from the Cost. This was done in order to award those groups whose members were generally close to each other, against those groups which merely had a large number of members.

The process that computes the fitness for each Genetic Group is described in Algorithm 2. In the algorithm, the $lowerThreshold$ and an $upperThreshold$ is used to give a pendulum effect to the algorithm. If two pair of COG's within a group have less than a predetermined number of differences, the algorithm will reward it by further decreasing the fitness value, which, because it

Table 3.11: Example of a phylogenetic profile centroid

Protein	g_1	g_2	g_3	g_4	g_5
p_1	1	1	0	1	1
p_2	1	1	1	0	1
p_3	1	0	1	1	1
p_4	1	1	0	0	0
p_5	1	1	1	0	1
$centroid$	1	0.8	0.6	0.4	0.8

is a minimization problem, it will make it a better solution. On the other hand, if a cluster has a big number of differences, the algorithm will retaliate by punishing the solution more and more each time a new difference is found. This is done in order to favor highly cohesive clusters, and to avoid clusters with a high variance within its members.

Algorithm 2 Evaluation function for the single objective genetic algorithm

```
pFactor ← 1
nFactor ← 1
for all COG's within the group do
  COGEvaluation = differences with the centroid
  if COGevaluation < lowerThreshold then
    COGevaluation* = pFactor
    pFactor-= positivePendulum
  else if COGevaluation > upperThreshold then
    COGevaluation* = nFactor
    nfactor+= negativePendulum
  end if
  groupEvaluation+ = COGevaluation
end for
return groupEvaluation
```

3.3.4 SELECTION OPERATORS

There are several operators in the Genetic Algorithms literature that we could have used, including but not limited to, Roulette, Boltzmann probability distribution, Tournament, Elitism.[17] At the end,preliminary experiments suggested that a combination of both Tournament and Elitism had the best results for our particular application.

3.3.5 IMPROVING THE ALGORITHM PERFORMANCE

We also applied a parallel global genetic algorithm architecture that enabled our algorithm to be run on parallel computing systems. Although it was only implemented on a Quad Core CPU, and a part of the algorithm was tested on a GPGPU system, the algorithm is capable of running on any HPC enabled system.

What we did was for the fitness function evaluation, and next population formation functions to dynamically divide the population among the different processors in our system in that timestep. (This layer is abstracted from the rest so that any parallel system can be implemented later). As such, when running the `generateCentroids`, evaluate and evolve operations we randomly divided the population between each parallel process, to later merge it again for input for the next algorithm step. It was done this way in order to recover the original stochastic method inherent to genetic functions, instead of having 4 different populations evolving independently. Internal experiments show that for the 4 core system we can expect a 300% improvement on the performance of the algorithm.

3.3.6 INTEGRATING THE ALGORITHM

Now that we have described each of the parts of the algorithm, we will proceed to explain how it all connects.

Algorithm 3 Overview of the complete algorithm

```
population ← randomGenerator(100 chromosomes)
evaluator ← profileDistanceFitness
operator ← {RandomNewGroup,RandomDeleteGroup,MergeStrong}
operator ← {RehashWeak,SplitStrong,Weeding}
evolution ← {Elitism, Tournament}
load the input databases
(in parallel) population.generateCentroids
(in parallel) population.evaluate(evaluator)
while population.averageEvaluation > threshold do
  (in parallel) population.evolve(evolution,operator)
  (in parallel) population.generateCentroids
  (in parallel) population.evaluate(evaluator)
end while
return  population
```

The so called threshold that ends the cycle is normally considered a cumulative change over a number of generations. When over the generations the evaluation average does not show a significant change, the evaluation will finish.

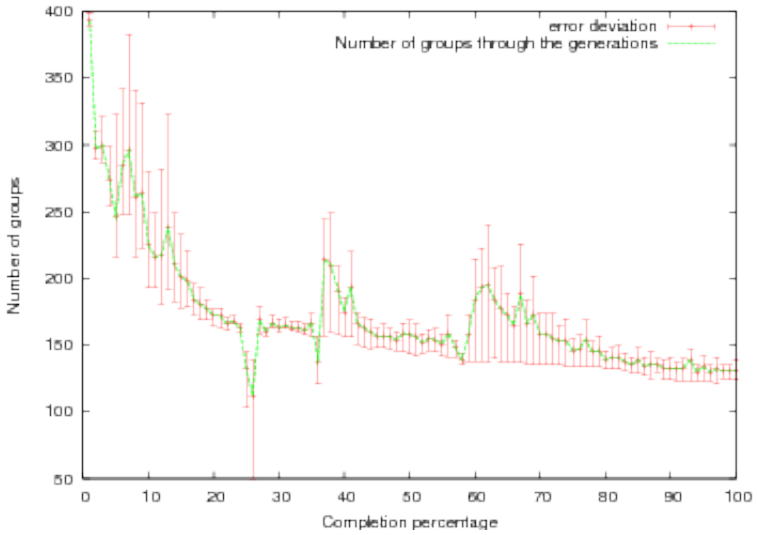

Figure 3.8: Average run of the GGA engine

3.4 RESULTS USING A SINGLE OBJECTIVE EVALUATION FUNCTION

We conducted a large series of computational experiments on obtaining the clusters on using GGA. Through this experiments, we obtained two sets of results. The Grouping Genetic algorithm gave us both the number of groups, and created the respective clusters using only lists indicating the co-occurrence of the phylogenetic data input. The data was validated using the ECOCyC and DIP databases, against the clusters proposed by the k-means method. It is important to note that for the algorithm determining the number of groups and determining the composition of the clusters is something it does at the same time, however for the sake of reporting we will be separating them.

3.4.1 NUMBER OF GROUPS

Throughtout the numerous number of computational experiments we made, the final number of clusters was fairly consistent. In Figure 3.8 we can see a graph that represents how the groups evolved throughout a series of experiments, with the brackets representing the highest variations from the average. As we can see, towards the end the algorithm always stabilized around the 120 groups area. The ideal number of groups determined by the algorithm is a number between 110 and 120 groups, however, as we can see below, the speed of the algorithm was highly dependent on how the data was initialized in the first random construction. Despite this, the algorithm always arrived at the same range in terms of the number of groups, which demonstrated the robustness of the genetic algorithm.

Table 3.12: Percentage of correct EcoCYC/DIP group assignations

Run	DB matches
Total DB entries	346
GGA Average Matches	256.7
k-means Average Matches	203

3.4.2 RUNNING TIME

In our initial version each generation took approximately 10 seconds to be completed, from the very start of the evaluation to the generation of the next one, our algorithm runtime would ranged between 15 and 20 minutes. The algorithm was later optimized for the fitness, mutation, crossover, and special functions to be run in parallel so that we could use the server's multi processors capabilities as much as possible, and we could see an increase in speed in an order of 5 reducing the clustering time to as little as 4 minutes. The results in this section, including the average shown in 3.8, were extracted and calculated among an average of 20 runs of the algorithm which varied between 100 and 300 generations each.

Still this can be seen as a time much bigger than that shown by classic grouping algorithms (specially when compared to algorithms like k-means) however we have to take into account that our GGA algorithm has a very flexible fitness functions that permitted us not only to include co-occurrences, but several other factors that helps us in refining our groups, something that would be extremely complicated to implement in an algorithm such as k-means.

3.4.3 VALIDATION OF RESULTS

The significance of the data mined by automated methods, specially when those are intended to discover functionality which is unknown as of yet, is often difficult to assess. In the validation phase we compared our results against known Biological databases, such as ECOCyC and DIP. Similarly, the algorithm was found to identify several biological patterns whose correlation has already been experimentally proven previously.

DIP and EcoCyC comparison:

As we mentioned, we used the EcoCyC and DIP database as a means to assert the expressiveness of the results thrown by our algorithm and *k*-means. The entries found in DIP and EcoCYC entries are a good way to measure the usefulness of our non-co-occurrence related metrics, since many pairs in that list does not show that type of relationship, as proven by the direct approach k-means takes. As it is shown by the validation process, GGA demonstrated to have a higher proficiency at finding these kinds of relations.

Table 3.13: Excerpt from the GGA group where the *purEK* pair is located

COG	Phylogenetic profile
COG0581	11101101111111111110110011
COG0413	01001111111011111011110000
COG0458	11110011111111111111110000
COG0065	11101011111111111111110000
COG0685	00011111101111111111110000
COG0035	01101111111111111101110011
COG0104	11111011111111111111110000
COG0287	11110111111111111111110000
COG0352	10011011011111111111110000
COG0157	11101011101111111111110000
COG0079	11111111111111111111110000
COG0045	11110111011011111111111100
COG0382	11111111011011111111111100
COG0714	11111111111011111010010010
COG0807	10010011111111111111110100
COG0026	01010011111111111111010000
COG0251	01001111111111111111110000
COG0041	11111011111111111111110000
COG1011	01111011111110111111010010
COG0226	11101101111111111110110011
COG0778	11111011111111111111110000
COG0127	11111111111111111111110110
COG0512	11111111111111111111110000
COG0059	10101011111111111111110000
COG1985	11100111111111111111110100

Operon comparison:

In order to assess the biological expressiveness of the results we obtained, they were compared against a number of operons, as mentioned earlier. Our first comparison was with respect to the *purEK* operon (which is a combination of COG's COG0026 and COG0041); the results obtained for our GGA algorithm are shown in Table 3.13.

An excerpt of the *k*-means output is shown in Table 3.14. This method failed to find *purEK*, this method lacks the means to correlate beyond the co-appearance factor.

Similarly, we compared the results of the *Leu E. coli* operon, which includes entries COG0066, COG0473, COG0119 and COG0065. The results of this comparison are shown in Table 3.15

A comparison against the *hflA* operon was made as well. This operon consists of a putative *GTPase* (COG2262) and a putative integral membrane protease (COG0330). Our algorithm correctly

Table 3.14: Excerpts from the *k*-means group were the *purEK* disjointed pairs were located

COG	COG
COG0364	COG0034
COG0176	COG0461
COG0337	COG0284
COG0140	COG0151
COG0703	COG0150
COG0590	COG0152
COG0299	**COG0041**
COG0026	COG0047
COG0801	COG0046
COG0161	COG0108
COG0502	COG0294
COG0132	COG0476
COG1539	COG0001

Table 3.15: Excerpt from the GGA group where the *Leu* and *SpeED* operon pair are located

COG	Phylogenetic profile
COG0473	101010111111111111111111000
COG0119	101110111110111111111110000
COG0065	111010111111111111111110000
COG0066	101010111111111111111110000
COG0104	111110111111111111111110000
COG0157	111010111011111111111110000
COG0152	111110111111111111111111000
COG1989	111011011101111111111110000
COG0581	111011011111111111110110011
COG0778	111111011111111111111110000
COG0147	111111111111111111111110000
COG0512	111111111111111111111110000
COG0714	111111111101111101001001O
COG0547	111111111111111111111110000
COG0159	111111111111111111111110100
COG0133	111010111111111111111110100
COG0421	101111111110101100111110000
COG1586	101111011000101100100000000

Table 3.16: Exerpt from the GGA group where the *hflA* operon pair is located

COG	Phylogenetic profile
COG0519	11111111111111111111110110
COG0330	11111111111111111111111010
COG0611	11111101001011111111110000
COG1841	11111111111110111111011010
COG0475	10111111011111111111111010
COG0825	11001011111111111111111100
COG1120	11111000111111111101110001
COG2262	01101101111111111111010100
COG0493	00011111111111111010111010
COG0473	10101011111111111111111000
COG0026	01010011111111111111010000

assigned these two proteins to the same group, as shown in Table 3.16. On the other hand, *k*-means was not able to recognize the relationship between these two proteins, and separated them into two groups.

The *SpeED* operon also showed to have better results in our engine than *k*-means (COG1586 and COG421), as shown in Table 3.15

3.5 EARLY RESULTS DISCUSSION

The results mined by our algorithm were of more expressiveness and biological significance than those obtained by more traditional grouping algorithms, as shown by the comparison data. This was a byproduct of the added biological data that we were able to include in the fitness function that further refined our group formation process. If we were to do the same thing in *k*-means, it would require us to add this biological knowledge as an annex to the co-occurrence binary vector. However, this is not as simple as it appears. For one, the data representation structure would have to radically change, since there is no direct relationship between the data expressed by a co-occurrence database such as the COG one, and the one data expressed by a database like DIP or ECOCyC. As we start adding more biological data to the grouping function, like some text mining to determine relationship based on the number of papers a certain number of COG's appear together, or a co-appearance database, the representation problem further escalates. This problem is practically inexistent when defining this evaluators as just different steps within a fitness function in a GGA system. Each factor just alters the fitness in one way or another, without its respective data representation interfering at all with each other, making the resulting program much more scalable, which results in a much easier way to further include new biological data in the way we form groups.

Moreover, the fact that the algorithm requires minimal input is another tremendous advantage.

Not even the number of groups is of great importance, since this is evolved by the algorithm itself. When running this type of protein function discovery algorithms, we do not possess much more information than the bare minimum, and as such, the number of groups, the distance between the centroids of each group and other factors classical grouping algorithms tend to ask for, is something we traditionally determine either by using some other heuristic that will try to approximate these values, or by mere trial and error.

As such, an algorithm that makes the determination of these initial parameters as part of the search heuristic is of tremendous help. GGA's have the advantage that they will not only determine the number of groups and their composition as it has been discussed several times before, but also has the flexibility as well to determine a myriad of other values.

However, there are still some disadvantages that need to be addressed in this algorithm. Although the results are computationally consistent in terms of the correctness of the clustering of the phylogenetic profiles, they are still biologically insufficient in the information they possess. There are several ways to solve this problem. One of the approaches that has been tried is that of including more biological information into the search process, like STRING[39] does. STRING uses various information sources independently in order to determine relationship levels between a network of proteins, and then present the results to the user in a table. That is, an *a posteriori* analysis.

Not to mention that as we can see with some of the large group sizes, specially the one shown in Table 3.13, there is little control over the size of the groups, and overall a tendency to overgeneralize can be seen in the algorithm. In general, it would be desirable to be able to have a degree of control over the abstraction level the algorithm imposes over its clustering rules.

What we propose in the second part of our work is to integrate all this analysis in an *a priori* approach through Multi Objective Genetic Algorithms[40, 41]. Our hypothesis is that integrating multiple biological information into the genetic algorithm would direct us to a strong clustering technique, and ultimately give us better results.

3.6 INTRODUCING MULTI OBJECTIVE FITNESS EVALUATION FUNCTIONS

For our multi-objective computational experiments we used three distinct fitness functions. A fitness function that clustered data using phylogenetic profiles, another that clustered data using gene transcriptional directionality, a third one that considered intergenic distance among the members of a cluster and a final one that rewarded cluster connectivity.

3.6.1 PHYLOGENETIC PROFILES

Taking advantage of the modular design of our implementation, we directly adopted the Phylogenetic Profiles Distance fitness function we had used for the previous version of the algorithm and just added the other two to the engine.

Table 3.17: Example of a context array field from an hypothetical COG

	s_1	s_2	s_3	s_4	s_5
Starting Point	32	16	17	34	12
Finishing Point	45	28	21	45	34

3.6.2 TRANSCRIPTIONAL DIRECTIONALITY

In a very similar approach as the one used for phylogenetic profiles, we constructed a centroid of the group considering the number of groups that had a left or right directionality across the set of species indicated in the phylogenetic profiles as containing the proteins in the cluster, and added it as an additional field to each Group Gene. From here it becomes a task of minimizing the distance to the centroid, or minimizing the differences in directionality of the proteins found in a particular species, among all the species where our set of proteins are present. As we can see, both the phylogenetic profiles and transcription directionality fitness functions tend to create smaller groups.

3.6.3 INTERGENIC DISTANCE

In order to implement a Intergenic distance fitness function we first added a nx2 array (where n is the number of phylogenetic profiles) that would contain the information of where the transcribed protein original information started and finished, for each of the n species the array contains information from. (Table 3.17 contains a reduced example of this field). This information would be later used by the fitness function. Which would create a list of these fields for each of the members of the cluster it is analyzing, and arrange them on a species-starting/ending point- protein hierarchy in a nx2xk 3-dimensional array, where k is the number of members in the cluster. We can see an example for a single species s_n on Table 3.18. The third dimension is formed by forming a list of these for a set of species. From here, it becomes a matter of adding up all the distances between transcription points.

So, the intergenic distance fitness function is calculated as described in equation 3.1

$$\sum_{s=0}^{N} sp(s,p) - ep \tag{3.1}$$

$$\tag{3.2}$$

3.6.4 CLUSTER SIZE

This fitness function was introduced in order to balance the divisive nature of the other fitness functions. Since they tend to minimize distance, one of the approaches the genetic engine could

Table 3.18: Example field of the context 3 dimensional array

	p_1	p_2	p_3	p_4	p_5
Starting Point	32	49	60	90	160
Finishing Point	45	54	73	99	189

take would be that of making just unitary groups. In order to counter that tendency the cluster size fitness function was introduced, which is a fitness function that rewards those groups that despite having a low variability have a greater size than the average of the population. This fitness function has the disadvantage of being artificially created, in that it has no biological basis and is only used because of its clustering merits, however on the other hand it allows for a numerical graduation of the population cluster size that would allow someone analyzing the Pareto front to have a more educated selection of the optimal solution according to the particular application were the algorithm is being used.

3.7 IMPLEMENTATION OF THE EVALUATION FUNCTION

We used a niched Pareto algorithm as described by Horn[23], which combines both a Pareto optimum next generation construction with a fitness sharing scheme to maintain variability. In a broad sense we use a tournament scheme in order to select those members of the current population which will pass to the next population. The winners are selected by comparing two members of the population against a sample subset of the same population, and selecting the one which is non dominated in respect to their competitors . In case we find that the competitors are tied, we use a fitness sharing scheme to select those that will pass to the next generation. Through this method we can progressively construct better Pareto Fronts.

Our implementation is done as shown in Algorithm 4, for a parent population *parent*, and a target population *offspring*.

Additional to this, as we mentioned earlier we introduced a separate elitism function that covered for the NGPA inherent lack of a best solution conservation scheme. This elitism function worked by selecting the best member of each individual fitness function, and the so-called 'knee', that is, the solution that best balances all the fitness functions in our algorithm.

Algorithm 4 Algorithm for the creation of a new population

```
   while offspring.size < threshold do
2:1: c1 ← parent.randomMember
3:   c2 ← parent.randomMember
4:   list[] ← parent.randomMembers
5:   tournament(c1,c2,list)
6:   if c1 is dominated and c2 is nonDominated then
7:     offspring.add(c2)
8:   else if c2 is dominated and c1 is nonDominated then
9:     offspring.add(c1)
10:   else
11:     c3 ← fitSh(c1,c2)
12:     offspring.add(c3)
13:   end if
14: end while
15: return  offspring
```

4. FINAL RESULTS

4.1 FORMATION OF THE PARETO FRONT

The algorithm succeeded in conforming a viable Pareto Front that combines in different degrees the different fitness functions we used as shown in Fig. 4.2 (the intergenic distance fitness function of each chromosome is not included in the plot, however we have to remember that the result vector is a 4-dimensional value). Although further selection from this point is a task where a proper Biologist should take the decision of which criteria combination is more viable, we selected the element from the Pareto front that maximizes the profile distance, genome directionality and intergenic distance fitness functions for comparison purposes.

4.1.1 STATISTICAL ANALYSIS ON THE PARETO FRONT

For statistical purposes, we ran our algorithm 30 times for 500 generations each instance, using the four fitness functions described earlier. At the 500th population, every run of the algorithm had 200 members. We feeded all this data to a statistical engine and produced comparison plots using as a basis the mean and the standard error of the mean as measuring basis. For example,in 4.3 we are plotting a subset of the runs we performed merely superposed one over the other. Taking this data, we performed the statistical operations over every record of the algorithm, and obtained Figure 4.4. As we can see in this and from the statistical information in Fig. 4.5 4.6, there is a clear formation of a Pareto front that is consistently formed over several instances of our algorithm, further proving the stability and correctness of our stochastic approach.

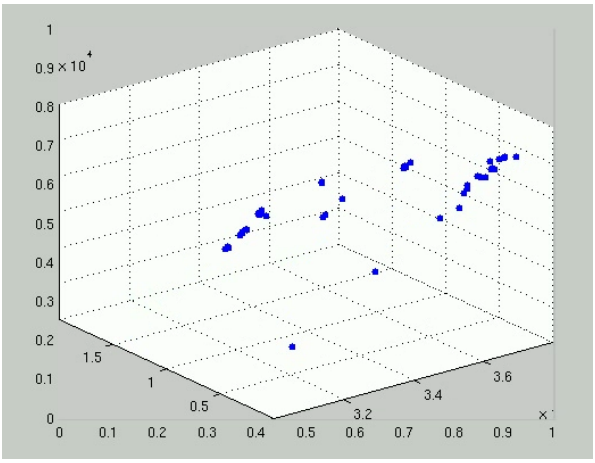

Figure 4.1: Initial random distribution of the population

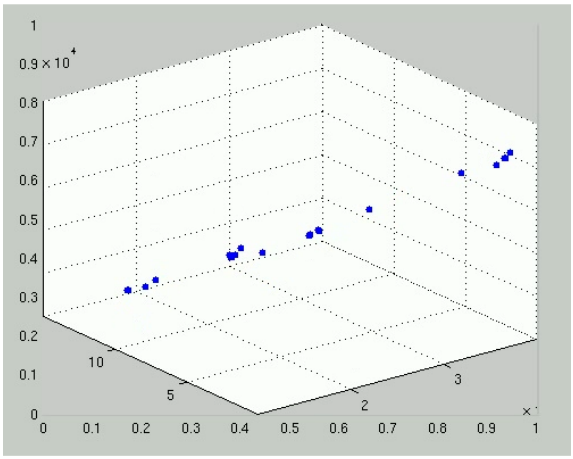

Figure 4.2: Pareto Front formation. The curve is bended towards the viewer

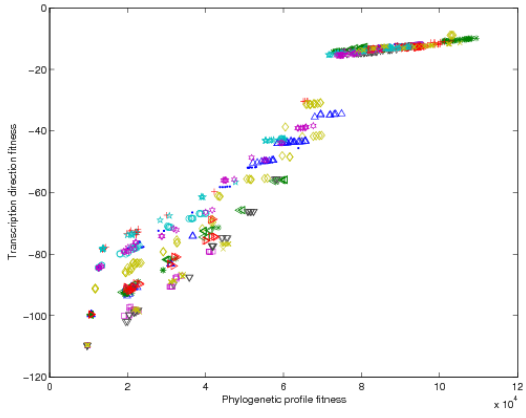

Figure 4.3: Raw superposition of a subset of our runs of the algorithm

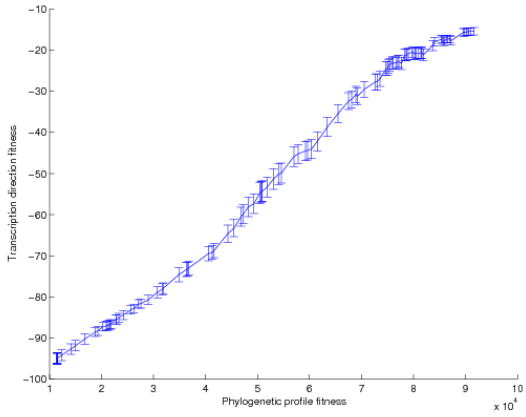

Figure 4.4: Phylogenetic profiles vs transcription directionality

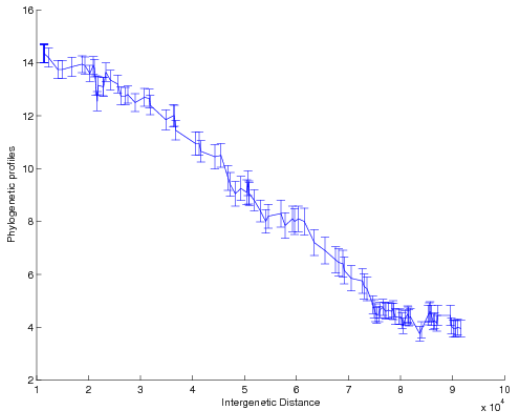

Figure 4.5: Intergenic distance vs phylogenetic profiles

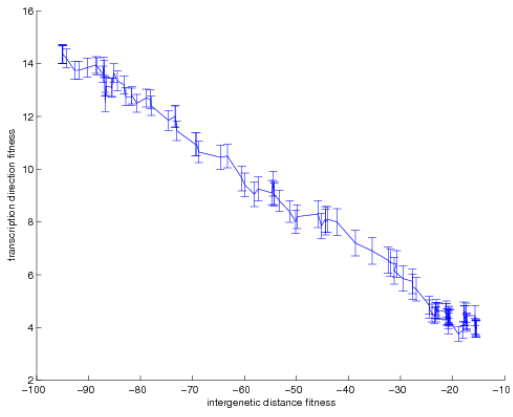

Figure 4.6: Intergenic distance vs transcription directionality

4.2 EXECUTION TIME

Despite having to deal with four evaluation functions instead of one as in our previous version, the computational complexity of the Genomic directionality and Connectivity fitness functions is equally linear - $O(n)$ -, while the Intergenic Distance function is O(n^2), being the overall complexity of the algorithm of a low polynomical order. As such, the only real bottleneck of the overall algorithm is the number of generations it takes to reach convergence.

4.3 ARRANGING OUR ALGORITHM

We ran our algorithm under several combination modes, each one was characterized by the number and the type of fitness functions we used. Two multi-objective optimization algorithms with only a phylogenetic profile distance or transcription directionality as the fitness functions (combined with the size fitness function), three modes resulting from the combination of our three biological fitness functions in pairs, and one mode resulting from the combination of all three fitness functions.

4.4 VALIDATION OF RESULTS

We compared against the same databases as we did in our first approach. In our first run, the algorithm ran only with phylogenetic profile distances, which is the same algorithm that was shown earlier with some minor optimizations. Although the clustering itself was not bad, when compared to real biological data it was not able to detect many operons (In particular it was able to consistently detect 6 operons. *clpPX* (clp Protease), *glmUS* (used for amino sugar biosynthesis), *SpeED*, *hFLA*, *leu E. coli*, and *purEK*. In our second run we only considered genomic directionality (GD). It is interesting to note that although the performance of this fitness function was slightly better than that shown by phylogenetic profiles, the overall convergence time for the algorithm in this run was much larger, being one of the slowest modes. Moreover, the engine showed severe difficulties in creating consistent groups. (In general this seems to be a characteristics of all the modes that lacked the phylogenetic profiles objective function). This mode was able to consistently detect 8 operons. In our third run we used both objective functions. This time around the algorithm was able to correctly identify over 15 operons, including the *dnaK* operon (made up from dnaK (COG0443) and dnaJ (COG0484), the group is shown in Table 4.2. It was also able to identify the *carAB* operon and the *fixABC* operon, in addition to the ones discovered by the previous algorithms and 10 others, which shows the robustness of the algorithm since some of these operons have considerable distance in terms of their phylogenetic profile but they are near each other in terms of their directionality and viceversa, completing each other in its space search. The overall table of results can be checked in Table 4.1

As we can see, while k-means has an arguably better performance than MOCEA when we use low levels of information, it is as we increase the biological data available to our algorithms that the strength of multi-objective optimization becomes more dominant, for the performance gain in each step is clear.

Table 4.1: Results chart

Mode	Operons Found	DIP-EcoCyc
PP	22	61
ID	14	37
PP & GD	90	88
PP & ID	110	87
ID & GD	20	58
PP & ID & GD	**125**	**100**
All with cross-over	30	50
k-means with PP	45	103
k-means with ID	47	78
k-means with both	46	85
Total	442	346

Table 4.2: Excerpt from the group where the DNA biosynthesis operon is located

p_1	COG0576	01110011111111111111111111	-010–10000010101101111001	
p_2	COG0210	01110011111111111111111111	-110–00011011010001101111	
p_3	COG0484	01110011111111111111111111	-010–01000010001100010001	dnaJ
p_4	COG0443	01110011111111111111111111	-010–01100011111101100000	dnaK
p_5	COG0190	01010011111111111111111111	-1-0–10101000101100111000	
p_6	COG0166	01110111111111111111110111	-101-10011010101111001-001	

5. CONCLUSIONS

Throughout this thesis work we have presented and demonstrated the viability of using multi-objective clustering genetic algorithms as a tool to infer protein-protein interactions. In the first approach we successfully proved that single objective clustering genetic algorithms, as a standalone technique is a good but not sufficient method to infer these types of interactions. Expressing phylogenetic profiles as a minimization problem was simple enough to use as a fitness function. In terms of the operators we used we inherited some of the operators that Falkenaeur first defined when he introduced clustering genetic algorithms, like randomly creating and destroying groups, and we defined some of our own, like merging groups whose centroids were particularly near, destroy those whose variance within its members was too big and searching and rearranging outliers within the clusters. We remained fairly classical in terms of the general structure of the rest of the algorithm, like using a random initialization, and using tournament as a selection function, since preliminary results showed that the algorithm gave better results through such procedures.

However, as we could appreciate there were still some glaring deficiencies in the method. One of the most notorious was that the first version of our model tends to make cluster that are too general (like in Table 3.13). As such, we need other methods that allow the algorithm to both be able to better discern the characteristics that make a group of proteins functionally related, and that allows for a level of control over the level of abstraction of the algorithm.

We then hypothesized that including more biological data in the form of expanding the fitness function as a multi-objective genetic algorithm would increase the quality of the results. As such during the second part of this thesis we demonstrated our initial hypothesis that considering additional sources of genomic data related to our biological problem has a definite positive effect in discovering patterns that are both computationally coherent in terms of the principles of clustering, and that have a real Biological significance. We introduced three additional functions, intergenic distance, genomic context and cluster size. The first two addressed the first problem we mentioned

in the previous paragraph, in that we provided our framework with the information it needed to improve the significance of the clusters it created. The other one, cluster size, while it was an artificial fitness function, at the same time it was a way that allowed us to have a degree of control over the clustering abstraction we desired from our method.

In terms of the rest of the genetic algorithm components, we inherited most of the ones that we had used in the previous iteration of our program, except for the selection function. Instead of the tournament selector we had previously employed, we constructed a niched pareto tournament scheme[23]. The methods related to the Pareto front are the most used by the community[19], and as such its effectiveness has been repeatedly tested by the community. The niche technique developed by Horn et. al. allows for a more comprehensive construction of the Pareto front, and as such we are guaranteed a better array of solutions.

Results demonstrated that the algorithm progressively produced better results as we introduced more and more biological information. It has often been asserted by many Biologists that the best computational algorithms should be able to obtain the best possible results with a minimum amount of information. However it is our belief that in realms as vast as Molecular Biology we should be able to incorporate into our algorithms as much available data as possible in a way that our methods are able to organize and make sense of all this data, and as such obtain a more rich, and meaningful result. The algorithm also retains the other benefits that its previous proposals possessed[9], like that of requiring minimal input data, and being able to evolve its own parameters over time (like the group size, which is a major problem with many clustering algorithms).

We believe that the proposed method can be generalized to problems outside of Biology. It is important to consider as much contextual information when designing a data mining algorithm as it is possible, in any kind of problem. It is often the case that as computer scientists we lose sight of the context information of our problem, and concentrate on making a computationally coherent model that is often too minimalistic, or lack the capability to be further expanded so that it becomes increasingly equivalent to the reality that we want to model.

The model we presented throughtout this thesis approximates this problem by allowing us to build a framework where it is extremely easy to introduce new optimization data into an existing search method. Further study on the applications of these type of algorithms in relationship to the data mining of biological data should be of great interest to the community.

Moreover, the fact that the algorithm requires minimal input is another tremendous advantage. Not even the number of groups is of great importance, since this is evolved by the algorithm itself. As mentioned before, when running this type of protein function discovery algorithms, we do not possess much more information other than the bare minimum provided by the problem, much less specific information like the number of groups, the distance between the centroids of each group and other factors classical grouping algorithms tend to ask for. As such this is something we traditionally determine either by using some other heuristic that will try to approximate these values, or by mere trial and error in the worst case scenario.

Returning to our problem, it is also important to note that despite the multi-objective system working better than its single objective and non genetic algorithm counterparts, there is still a big number of operons that our algorithm fails to classify, including the widely known *lac* operon.

It is our belief that introducing further biological data would help in the algorithm being able to discover all these relationships, all the while its ability to detect false positives would also increase, leading to more biological coherent clusters.

In summary from the computational side this algorithm can be considered as a comprehensive approach within the evolutionary algorithms community. We are successfully combining techniques from the parallel, multi objective and clustering divisions of genetic algorithms Since our problem was inherently a clustering problem, the usage of the clustering division was a given. Our first approach using phylogenetic profiles demonstrated that genetic algorithms was a viable method for protein protein interaction prediction, however the results weren't as comprehensive and informative as required. As such we introduced a multi objective scheme that allowed us to increase the quantity of data the fitness functions could fit in, leading to a better performance against our benchmark database. Parallel genetic algorithms were introduced as a means to speed up the algorithm.

On the biological side, we successfully constructed a framework that is able to predict protein-protein functional interactions using multiple and varied biological data in an *a priori* fashion. To the best of our knowledge this is the first time that genetic algorithms have been used to predict protein-protein information in a way that allowed us to introduce as much biological data as we desired, as long as it could be expressed as a minimization/maximization problem. We believe that this method could provide a foundation stone for a comprehensive framework that combines all the protein protein prediction methods known to the community.

5.1 FUTURE WORK

There are several lines on investigation that can be taken from where our algorithm is at this moment.

On the computational front we are interested on further speeding up the algorithm by using different methods from the parallel computing community. While we applied global parallel genetic algorithms in this thesis work as a means to accelerate the computation time, we are interested in implementing the algorithm in a parallel framework. This can be approached through two means. One is to build a distributed cluster, and to create an island genetic algorithmic framework. In a distributed cluster system we have a series of interconnected computer nodes, which have the characteristic of each node having a relatively strong computer power, however the communication bus is relatively slow. In this kind of architectures, normally the communication step is the bottleneck of the algorithm. As such an architecture such as island genetic algorithm, which evolve each island, or each population independently, and just "migrates" a small number of individuals between islands is ideal for this kind of setup.

A second approach would be applying our algorithm to a GPGPU architecture. This is explained in further detail in Appendix A.

We could also use modern alternatives to evolutionary algorithms that have demonstrated good results in other areas, like particle swarm optimization[42], or differential evolution[43].

5.1.1 OTHER COMPUTATIONAL IMPROVEMENTS

Another interesting research path would be incorporating ideas from the messy genetic algorithms framework [16]. The main idea in messy genetic algorithms is to take a bottom up approach towards the formation of the final solution. Roughly speaking, it divides the population into several populations which concentrate on the formation of a particular "building block", or partial solution. In terms of the Pareto optimum, it could be understood as several building blocks specialized in the construction of a particular section of the Pareto optimum. This could be achieved by giving preference, in different degrees, to our several fitness functions for each of the subpopulations. We believe that this approach would lead both to a more comprehensive formation of the Pareto front, and to a faster genetic algorithm when combined to the parallel approach described earlier.

5.2 BIOLOGICAL FUTURE WORK

We have made clear that this framework is specifically designed for it to be easy to construct an increasingly complex and comprehensive fitness function. MOCEA allows for any type of biological parameter to be introduced into the algorithm as long as it can be expressed as a minimization or a maximization problem. For example, phylogenetic profiles, and transcription directionality can be expressed as a minimization of the variance between the profiles of a group. Intergenetic distance can be expressed as a minimization of the distances in the genome among the proteins in a cluster. In the same way, we could consider other type of optimization functions related to the genetic context of our proteins. For example, we could consider the evolutionary distance between the different species we are analyzing. If a set of proteins is conserved among a group of species which are genetically close, it may not tell us much information about the functional relationship between these proteins, given that the conservation of certain genetic patterns could be mere coincidence. On the other hand, if we observe the same conservation pattern on species whose evolutionary history diverged a great time ago, and who are only loosely related, then we can have a greater confidence on the kind of patterns we find between these two species. As such, we can express our species through a distance graph, and we can insert a new optimization function which will seek to maximize the distance between the members of a clusters, such that clusters that only find patterns between similar species have less value than clusters that manage to find patterns across a wide arrange of species.

This is just an example for the kind of biological functions that could be inserted in the framework. We could also use protein microarrays. These are measurements devices normally used in biomedicine and biostatistics to determine the presence and amount of certain type of proteins in a given sample. Similarly to how we can extract statistical data from phylogenetic profiles, the same principle could be applied to microarrays.

When the algorithm is more mature in terms of the biological information it considers, it would also be extremely indicative to perform more thorought benchmarks with respect to similar comprehensive methods, such as voting methods or Bayesian networks.

BIBLIOGRAPHY

[1] FIERS, W., CONTRERAS, R., ET AL. Complete nucleotide sequence of bacteriophage ms2 rna: primary and secondary structure of the replicase gene. *Nature 260*, 5551 (1976), 500–507.

[2] ADAMS, M., FLEISCHMANN, R., ET AL. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science 269*, 3 (1995), 469–512.

[3] BORK, P., DANDEKAR, T., DIAZ-LAZCOZ, Y., EISENHABER, F., HUYNEN, M., AND YUAN, Y. Predicting function: from genes to genomes and back. *Journal of Molecular Biology 283*, 4 (1998), 707–725.

[4] EISENBERG, D., MARCOTTE, E., XENARIOS, I., AND YEATES, T. Protein function in the post-genomic era. *NATURE-LONDON-* (2000), 823–826.

[5] PELLEGRINI, M., MARCOTTE, E., THOMPSON, M., EISENBERG, D., AND YEATES, T. Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proceedings of the National Academy of Sciences 96*, 8 (1999), 4285.

[6] WU, J., KASIF, S., AND DELISI, C. Identification of functional links between genes using phylogenetic profiles. *Bioinformatics 19*, 12 (2003), 1524–1530.

[7] WATANABE, R., MORETT, E., AND VALLEJO, E. Inferring modules of functionally interacting proteins using the Bond Energy Algorithm. *BMC Bioinformatics 9*.

[8] FERNANDEZ, J., VALLEJO, E., AND MORETT, E. Fuzzy C-means for inferring functional coupling of proteins from their phylogenetic profiles. In *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB'06. 2006 IEEE Symposium on* (2006), pp. 1–8.

[9] TAPIA, J., AND VALLEJO, E. A clustering genetic algorithm for inferring protein-protein functional interactions from phylogenetic profiles. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on* (2008), pp. 2757–2763.

[10] VON MERING, C., JENSEN, L., KUHN, M., CHAFFRON, S., DOERKS, T., KRUGER, B., SNEL, B., AND BORK, P. STRING 7–recent developments in the integration and prediction of protein interactions. *Nucleic Acids Research 35*, Database issue (2007), D358.

[11] GIBSON, G., AND MUSE, S. V. *A Primer of Genome Science*. Sinauer, 2005.

[12] FALKENAUER, E. *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc. New York, NY, USA, 1998.

[13] SELBACH, M., AND MANN, M. Protein interaction screening by quantitative immunoprecipitation combined with knockdown (QUICK). *Nature Methods 3* (2006), 981–983.

[14] LIN, C., AND WANG, M. Predicting protein function by genomic context: quantitative evaluation and qualitative inferences. *Genomic Research 10*, 8 (2000), 1204–1210.

[15] TATUSOV, R., FEDOROVA, N., ET AL. The cog database: an updated version includes eukaryotes. *BMC Bioinformatics 4* (2003), 41–54.

[16] GOLDBERG, D. E. *Genetic algorithms, in search, optimization, and machine learning*. Addison-Wesley, 1989.

[17] ENGELBRECHT, A. *Computational Intelligence: An introduction*. Wiley, 2007.

[18] HOLLAND, J. *Adaptation in Natural and Artificial Systems. An introduction*. University of Michigan Press., 1975.

[19] COELLO COELLO, C., ET AL. *Evolutionary Algorithms for Solving Multi Objective Problems*. Kluwer Academic Publishers, 2002.

[20] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation 6*, 2 (2002), 182–197.

[21] ZITZLER, E., LAUMANNS, M., THIELE, L., ET AL. SPEA2: Improving the strength Pareto evolutionary algorithm. In *EUROGEN* (2001), pp. 95–100.

[22] SARENI, B., AND KRAHENBUHL, L. Fitness sharing and niching methods revisited. *Evolutionary Computation, IEEE Transactions on 2*, 3 (1998), 97–106.

[23] HORN, J., NAFPLIOTIS, N., AND GOLDBERG, D. Multiobjective optimization using the niched pareto genetic algorithm. *Urbana 51* (1993), 61801–2996.

[24] WU., F. Genetic weighted k-means algorithm for clustering large-scale gene expression data. *BMC Bioinformatics 9*, 6 (2008).

[25] LIN, C., AND WANG, M. Genetic-clustering algorithm for intrusion detection system. *International Journal of Information 2*, 2 (2008), 218–234.

[26] MILANO, M., AND KOUMOUTSAKOS, P. A clustering genetic algorithm for actuator optimization in flow control. In *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, pp. 263–269.

[27] CURTEANU, S., LEON, F., AND GALEA, D. Alternatives for Multiobjective Optimization of a Polymerization Process. *Journal of Applied Polymer Science 100*, 5 (2006), 3680–3695.

[28] MANDAL, C., GUDI, R., AND SURAISHKUMAR, G. Multi-objective optimization in Aspergillus niger fermentation for selective product enhancement. *Bioprocess and Biosystems Engineering 28*, 3 (2005), 149–164.

[29] SOMEREN, E., ET AL. Multi-criterion optimization for genetic network modeling. *Signal Processing 83* (2003), 763–775.

[30] DEB, K., AND RAJI REDDY, A. Reliable classification of two-class cancer data using evolutionary algorithms. *BioSystems 72*, 1-2 (2003), 111–129.

[31] DE SOUTO, M., AND FACELLI, K. Multi-objective clustering ensemble. In *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems* (2006).

[32] LLOYD, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137.

[33] TATUSOV, R., NATALE, D., ET AL. The cog database: new developments in phylogenetic classification of protein from complete genomes. *Nucletic Acids Research 29*, 1 (2001), 22–28.

[34] CIRIA, R., ABREU-GOODGER, C., MORETT, E., AND MERINO, E. GeConT: gene context analysis. *Bioinformatics 20*, 14 (2004), 2307–2308.

[35] SALWINSKI, L., MILLER, C., SMITH, A., PETTIT, F., BOWIE, J., AND EISENBERG, D. The Database of Interacting Proteins: 2004 update. *Nucleic Acids Research 32*, 90001 (2004), 449–451.

[36] KARP, P., KESELER, I., SHEARER, A., LATENDRESSE, M., KRUMMENACKER, M., PALEY, S., PAULSEN, I., COLLADO-VIDES, J., GAMA-CASTRO, S., PERALTA-GIL, M., ET AL. Multidimensional annotation of the Escherichia coli K-12 genome. *Nucleic Acids Research 35*, 22 (2007), 7577.

[37] ITOH, T. Evolutionary instability of operon structures disclosed by sequence comparisons of complete microbial genomes. *Molecular biology and evolution 16*, 3 (1999), 332–346.

[38] DUESTER, G., CAMPEN, R., AND HOLMES, W. Nucleotide sequence of an Escherichia coli tRNA (Leu 1) operon and identification of the transcription promoter signal. *Nucleic Acids Research 9*, 9, 2121–2139.

[39] JENSEN, L., KUHN, M., ET AL. String 8–a global view on proteins and their functional interactions in 630 organisms. In *Pubmed* (2009).

[40] TAPIA, J., MORETT, E., AND VALLEJO, E. A Clustering Genetic Algorithm for Genomic Data Mining. *Foundations of Computational Intelligence Volume 4: Bio-Inspired Data Mining* (2009), 249.

[41] TAPIA, J., MORETT, E., AND VALLEJO, E. MOCEA: A Multi Objective Clustering Evolutionary Algorithm for Inferring Protein-Protein Functional Interactions. In *GECCO 2009* (2009).

[42] ENGELBRECHT, A. *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.

[43] PRICE, K., STORN, R., AND LAMPINEN, J. *Differential evolution: a practical approach to global optimization*. Springer, 2005.

[44] SCHATZ, M., AND TRAPNELL, C. Fast exact string matching on the gpu. *Center for Bioinformatics and Computational Biology* (2007).

IMPLEMENTING THE FITNESS FUNCTION ON A GPU

GPGPU refers to a relatively novel approach where we take advantage of the highly parallel computing capabilities of the graphics processor unit for tasks other than graphics related ones. GPGPU has saw great success in many different areas, and Bioinformatics is no exception. In [44], a string matching algorithm, one of the most common type of problems in gene analysis, was successfully ported to the GPU, meeting a 35x performance increase in terms of the computing time spent to find the desired string. We have already succeeded in implementing the phylogenetic profile fitness function in the GPU as a test case, and the construction of a distance matrix between all the phylogenetic profiles in our database.

The first task that has to be appointed is the construction of the distance matrix. The process starts by copying the list of phylogenetic profiles into a $n \times m$ texture, where n is the number of profiles and m the length of each of them, as can be seen in Table 3. The output matrix is computed according to the pseudocode 5, which will result in a $n \times n$ distance matrix, being n the number of phylogenetic profiles we possess. i and j refer to the number n of phylogenetic profiles we are going to evaluate. k refers to all the species in our phylogenetic profile. A reduction operators refers to reducing a set of values (a vector of values) to a single value through a binary operator. In this case, a sum, since we want to obtain the addition of all the xor's. (For example, a reduction of the vector 3 4 5 using the binary operator ADD would be 12, since 12 is the sum of 3,4 and 5. Reduction is one of the fundamental operations of the parallel computing paradigm)

Algorithm 5 Kernel for the computation of a distance matrix

```
   for concurrently each i in inputTexture do
2:1: for concurrently each j in inputTexture do
3:     for concurrently each k in species do
4:       temp_k =i_k xor j_k
5:     end for
6:     distanceMatrix(i,j) = Perform a reduction over all temp_k
7:   end for
8: end for
```

As it was mentioned earlier, the computation of such a table is quite difficult to do through direct means on a CPU, given the very big size of the structure and the massive quantity of computations. On the other hand, such a task is almost trivial to do on the GPU, as this size can be easily mapped and stored on a texture. Moreover, the operations required to buildup the data (a series of XOR's) are very similar and repetitive in nature, perfect to be delegated to the GPU. As such it is clear why this fitness function should be done on an architecture designed for parallel computations.

Table 1: Input Texture

n/m	0	1	2	3	4
0	1	1	0	1	1
1	1	1	1	0	1
2	1	0	1	1	1
3	1	1	0	0	0
4	1	1	1	1	1
5	1	1	1	0	1

Table 2: Output Texture

n/m	0	1	2	3	4	5
0	0	2	2	2	1	2
1	2	0	2	2	1	1
2	2	2	0	1	1	1
3	2	2	1	0	3	2
4	1	1	1	3	0	1
5	2	1	1	2	1	0

An example of the input texture structure is shown in Table 1, and the resulting distance matrix is shown in Table 2

After this table is built up is when the program is ready to receive input data from the genetic engine, evaluate each group it receives according to the distance matrix and return an array with the fitness it calculated for each group. The fitness it calculates is defined by the algorithm 6.

So it will compute the distance of all the members of a group against the rest of the members within the group, and return the sum of these values. Again, despite this being a simple set of operations, the number of times it has been done is simply too large (approximately 20000 times per generation), and the repetitive nature of the operation perfectly suits the requirements of this process.

The time comparison of the GPU and CPU implementations are shown in Table 3

Our algorithm showed to take advantage of the parallel capabilities of the graphic card. Table 3 shows a comparison of the computing time it takes to do the same operations in both the CPU and the GPU. Despite the GPU implementation having to compute a look up table, this is only a one time operation which significantly benefits the computing time of the GPU in the long run. The CPU takes nearly 250 times as much as our GPGPU algorithm to perform the same simple table filling function. Although the performance improvement is astonishing, it could be argued that since this is a one time process it is of little relevance when taking into account the total

Algorithm 6 Kernel for obtaining the distance values

```
   for concurrently each i in inputTexture do
2:1: for concurrently each j in inputTexture do
3:     distanceStructure(i,j) = tex(i,j) {build the intermediate
       structure}
4:   end for
5: end for
6: for concurrently each i in inputTexture do
7:   Apply a reduction function to the intermediate structure
8: end for
```

Table 3: Time comparison

Operation/Device	GPU(ms)	CPU (ms)
distance matrix construction(3307 elements)	321.47	95093.835938
Group comparison(500 groups)	.499(ms)	50
Group comparison(6000 groups)	.537(ms)	682.5

time. In consequence, of much more significance is the difference seen in the computation of the distance fitness function in each generation. While the GPU algorithm shows an excellent running time (half a millisecond) with good scalability (seeing close to no increase in running time when increasing the groups to evaluate tenfold), the CPU algorithm shows a much bigger running time, taking more than 10ms in evaluation time per group, making it a not so scalable algorithm.

However, these two were only implemented as a test case, and were not integrated to the whole algorithm framework. It would be a future investigation line to port the whole algorithm to run under a GPGPU framework, and to compare the performance. It has already been shown that most of the critical functions show a high degree of parallelization, so we should be seeing a big leap in the algorithm, performance wise.