IDENTIFYING PEPTIDE MOTIFS USING GENETIC ALGORITHMS

by

Deniz Tanrıseven (Sağlık)

B.S. in Computer Engineering, Eastern Mediterranean University, 1998

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Boğaziçi University
2000

ACKNOWLEDGEMENTS

I would like to thank to my dear professor Uğur Sezerman for his great efforts and understanding in this work. Without him, I would have been lost.

Thanks a lot to Assoc. Prof. Cem Ersoy and Assoc. Prof. H. Levent Akın for their valuable ideas and help.

Finally, many thanks to my family and dear friends for their support and courage.

ABSTRACT

IDENTIFYING PEPTIDE MOTIFS USING GENETIC ALGORITHMS

Finding the ligand motifs binding to the receptor molecules is crucial in vaccine and drug design, especially for the MHC-peptide problem. In this work, for determining the peptide motifs binding to specific MHC molecules, we have used regression analysis. In order to find the the optimum regression line, genetic algorithm (GA) techniques are used because in traditional regression analysis methods, you may not be able to reach the optimum solution. The optimum regression line generated by the GA also determines the factors on the MHC molecules that makes the peptide bind to these MHC molecules.

The efficiency of the GA is tested by doing several tests on its different parameters, and the optimum set of parameters are determined for this problem. Results have shown that we are able to predict second position of a peptide motif with 95 per cent exact match or 100 per cent close match within one standard deviation of the predicted equation. We have divided last position's data into two parts in order to explain it with two regression lines. Predictions for the last position of the peptide motif with the first regression line resulted in 80 per cent exact match. Second regression line resulted in 75 per cent exact match.

ÖZET

GENETİK ALGORİTMALAR KULLANARAK PEPTİDLERDE MOTİF BULMA

Alıcı moleküllere bağlanan peptid motiflerini bulmak, aşı ve ilaç dizaynında çok önemlidir. Bunun en önemli uygulaması ise MHC-peptid problemidir. Bu çalışmada belirli MHC moleküllerine bağlanan peptid motiflerine karar vermek için regresyon analizi kullanıldı. Geleneksel regresyon analizi metodları ile her zaman optimum sonuç yakalanamadığı için optimum regresyon doğrusunu bulmak için genetik algoritma (GA) teknikleri kullanıldı. GA ile bulunan optimum regresyon doğrusu peptid motifini belirlemekle beraber MHC moleküllerinde, peptidlerin bu moleküllere bağlanması için gerekli olan etkenleri de bulmaktadır.

GA'nın yeterliliği değişik uygulama teknikleri ile test edilmiş ve bu problem için optimum parametre seti belirlenmiştir. Sonuçlar, peptid motifinin ikinci pozisyonunun bulunmasında yüzde 95 birebir uyumluluk ve yüzde 100, bir standart sapma ile uyumluluk göstermiştir. Son pozisyonu iki regresyon doğrusuyla açıklayabilmek için veri ikiye bölünmüştür. İlk regresyon doğrusu ile peptid motifinin son pozisyonunun bulunmasında yüzde 80, ikinci regresyon doğrusu ile ise yüzde 75 doğru tahmin yapılabilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABREVIATIONS

A	First pocket of an MHC Class I molecule
B	Second pocket of an MHC Class I molecule
C	Third pocket of an MHC Class I molecule
D	Fourth pocket of an MHC Class I molecule
E	Fifth pocket of an MHC Class I molecule
F	Sixth pocket of an MHC Class I molecule
P_2	Second peptide position
P_3	Third peptide position
$P_{5/6}$	Fifth or sixth peptide position
$P\Omega$	Last peptide position
Å	Angstrom distance
$Å^3$	Volume in Angstrom cube for sizes of amino acids
Ω	Last position of a peptide
α	Alpha-helix protein structure
β	Beta-sheet protein structure
σ	Standard deviation

APC	Antigen presenting cell
DNA	Deoxyribonucleic acid
GA	Genetic algorithm
GFA	Genetic function approximation
HLA	Human lymphocyte antigen
LMP	Low molecular-mass polypeptide
MHC	Major histocompatibility complex
MRI	Magnetic resonance imaging
QSAR	Quantitative structure-activity relationship
RER	Rough endoplasmic reticulum
RNA	Ribonucleic acid
SGA	Simple genetic algorithm
SSE	Sum of squares error

SSR	Sum of squares regression
SST	sum of squares total
TAP	Transporter of antigenic peptide
TC	Cytotoxic T cell
TCR	T-cell receptor
TH	Helper T cell

1. INTRODUCTION

Nature has a robust way of evolving successful organisms. The organisms that are not suitable for an environment die, whereas the ones that are fit live to reproduce. This Darwinian theory of evolution depicts biological systems as the product of the ongoing process of natural selection. A genetic algorithm can be considered as a form of evolution that occurs on a computer [1,2]. In its simplest terms, it means the survival of the fittest. The choice of the fittest is done based on a set of operations such as selection, crossover, and mutation applied many times. The genetic algorithm begins with a random population - a set of individuals that are the candidate solutions, and the selection is done according to a fitness function, which depends on the problem and is very crucial in the algorithm's success [3].

Genetic algorithms (GAs) can be used for both solving problems and modeling evolutionary systems. They are widely used in function optimization, ordering problems and automatic programming. The idea of GA is also very promising in the field of computational molecular biology. There are several application areas such as Fragment Assembly – the sequence determination of DNA from DNA fragments [3,4], Structure Determination such as structure determination of proteins [5,6], and Motif Discovery that is finding motifs in biological sequences.

The aim of this master thesis was to apply GA for the identification of motifs in small protein sequences of peptides that are binding to the major histocompatibility complex (MHC) molecules. Here a motif can be defined as a pattern common to a set of nucleic or amino acid subsequences, which share some biological property of interest such as being DNA binding sites for a regulatory protein [7].

MHC molecules play a very important role in the intracellular immune response that is destroying the virus that has managed to enter into the cell. The physiological function of a MHC molecule is to bind to degraded fragments of antigen generated inside infected cells and display them for recognition by T cells. T-cell receptors exist only on the surface

of T cells, and the antigenic determinants they recognize are peptides derived from foreign proteins and complexed with MHC molecules on the surface of cells.

The viral peptides that bind to MHC molecules are usually eight to 10 residues long, whereas the protein sequence of a virus is much longer than this. There are many possibilities of eight to 10 residue long peptides for a given sequence and not all of those can bind to the MHC. Which of the peptides among different possibilities can bind to a MHC molecule is a major problem, known as the MHC-peptide problem [8,9]. Note that, knowing the specific peptide sequence, which can bind to a MHC molecule is very important for the design of new vaccines.

Antibody and T cell mediated immune responses are initiated through genes contained within MHC, so in this sense the MHC represents the front end of the adaptive immune response to invading pathogens. A large amount of effort has gone into understanding of this gene complex, the genes encoded within the region, and the role the complex plays in immunobiology and human pathology, including diseases of both infectious and genetic origin [10]. Many researchers are carrying out different studies to identify peptide sequences that bind to specific MHC molecules, and there are many MHC-peptide sequence databases constructed as a result of these studies [11-16], which were used in our study. By using the known MHC-peptide sequences as input data, and genetic algorithm technique for problem solving, our aim was to predict the possible peptide motifs that will bind to MHC molecules so that for a given individual with a known MHC, which viral peptide sequences will be able to bind to that specific MHC can be determined.

The first step in this study was to determine which of the MHC sequence positions were the defining positions for the peptide to bind to it. These positions are called peptide-binding motifs of the MHC molecule. After predicting these positions of a group of different MHC molecules, the next step was to predict the peptide anchor residues that will bind to those positions of the unknown MHC molecules. The genetic algorithm technique together with the multiple regression analysis has been used as the solution method to our problem. The multiple regression analysis was used to find the fitness values for the individuals that will be fed into the genetic algorithm phase. Another important result of the regression analysis is the set of MHC sequence positions that will be used to make

future predictions of peptide sequences. Initially a simple genetic algorithm (SGA) was implemented and later modifications to it have been done in order to increase the performance of the SGA.

The second section is devoted to the MHC-peptide problem, its biological and historical background, which states the problem in detail, and previous works done in order to overcome this problem.

In the third section, you will find a brief introduction to GAs, their historical background, and some problems tackled with GAs, which are related to the problem we were aiming to solve.

In the fourth section, the problem definition and the methodology that we have followed in our implementation is explained in detail.

The fifth section presents the results of our implementation and the sixth section gives the conclusion.

2. MHC – PEPTIDE PROBLEM

2.1. Biological Background

The immune system in vertebrates provides a defense mechanism against foreign bodies such as viruses and bacteria. Three main properties are essential to its operation: specific recognition of foreign molecules, which also involves discrimination between self and non-self; the ability to destroy the foreign bodies; and a memory mechanism that results in a more rapid response to a second infection by the same microorganism.

Foreign invaders are recognized through specific and tight binding of the proteins of the immune system to molecules specific to the foreign organisms. The sites on the foreign molecules that are recognized by the immune system are called antigenic determinants. Antigenic determinants interact with two different classes of antigen receptors on the surface of the two major cell types of the immune system. Antibodies, which are also known as immunoglobulins, act as antigen receptors on the surface of B cells, which are stimulated by antigen binding to secrete antibodies into the bloodstream. The major function of a T-cell receptor, which resides on the T cell is to recognize and destroy the virus-infected cells [17].

Antibodies at the surface of B cells are involved in the extracellular immune response that is they recognize and bind intact soluble molecules or molecules on the surface of invading microorganisms outside the body cells. On the other hand, T-cell receptors are involved in the intracellular immune response that is they directly destroy the body cells, which the foreign bodies have managed to infect. One important point in the recognition of an antigenic determinant by the T-cell receptor is that, the antigen should be presented as a part of a complex with a third important class of protein molecules, the molecules encoded by the major histocompatibility complex, shortly MHC, on the surface of cells. The function of the MHC molecules is to bind to degraded fragments of antigen generated inside infected cells, also called antigen presenting cells (APC), and display them for recognition by T cells. The complex formed by the MHC molecule and the

antigen bound to it is recognized by the T-cell receptor as a foreign organism, and destroyed by the T cell after recognition and proper binding.

2.1.1. Structure and Classes of MHC Molecules

There are two main kinds of MHC molecules; class I and class II. Both kinds of molecules belong to the immunoglobulin superfamily, but they are not immunoglobulins. The immunoglobulin, the T-cell receptor and the molecules of MHC belong to a family of proteins that seems to have evolved by duplication and diversification from a single ancestral domain. The crystal structure of immunoglobulin was solved long before either of the two molecules and the presumed ancestral domain structure is therefore known as an immunoglobulin or immunoglobulin-like domain and all three belong to this superfamily [17]. Class I MHC molecules are found on all nucleated cells, while class II molecules show a more restricted expression pattern. Class II molecules are always expressed on B cells, interdigitating dendritic cells, and thymic epithelial cells.

Class I MHC molecules are composed of an α chain, which is the heavy chain of the MHC structure and a light chain, called β_2-microglobulin (β_2m). The α chain is divided into three domains, α1, α2, and α3 [18]. Class II MHC molecules are composed of an α chain and a β chain of which the α chain is divided into two domains, α1 and α2, and the β chain is also divided into two domains, β_1 and β_2 as seen in Figure 2.1 [19]. The extracellular domains of both class I and class II molecules show variability in their amino acid sequences, yielding grooves with different shapes. The grooves cradle processed antigens, holding the antigens steady for interaction with the T-cell receptor (TCR) on T cells. CD8 TC cells, a type of T cells, recognize antigen in association with class I MHC molecules, while CD4 TH cells, another type of T cells, recognize antigen in association with class II MHC molecules. The CD8 and CD4 accessory molecules have a weak affinity for MHC class I and class II molecules, respectively. They interact with the MHC-antigen peptide complex, thus stimulating intracellular events that cause T cell activation.

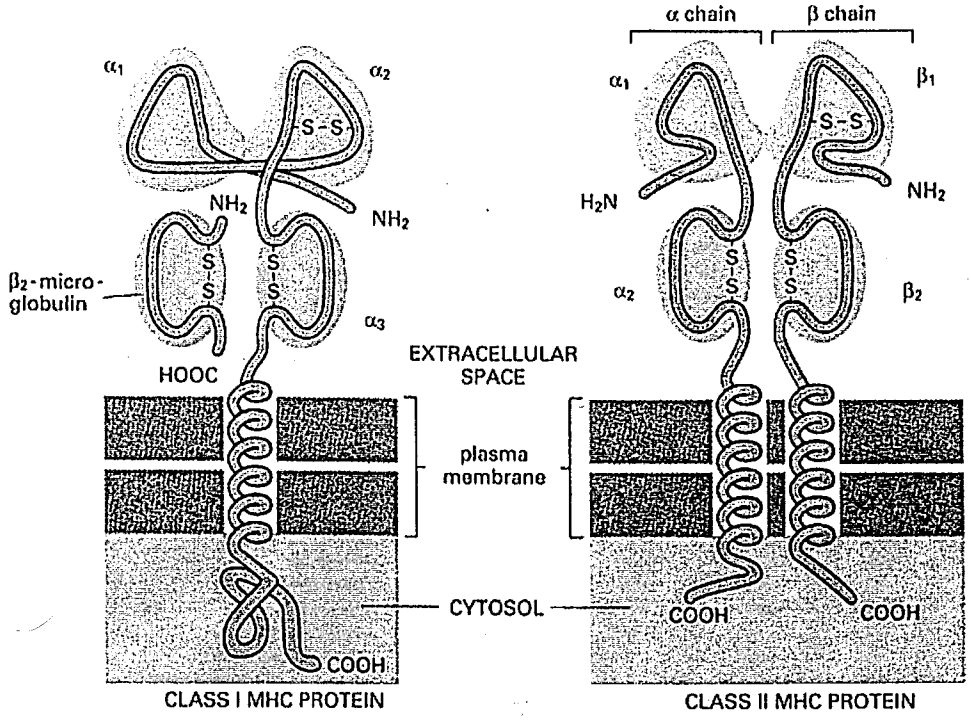

Figure 2.1. The structure of MHC class I and class II molecules

Figure 2.2 [19] shows a scheme for the structure of the ternary complex between MHC class I and class II molecules, foreign antigenic peptide, and the T-cell receptor. The peptide is located in the groove formed by the α1, α2 domains of the MHC class I molecule and α1, β1 domains of the MHC class II molecule.

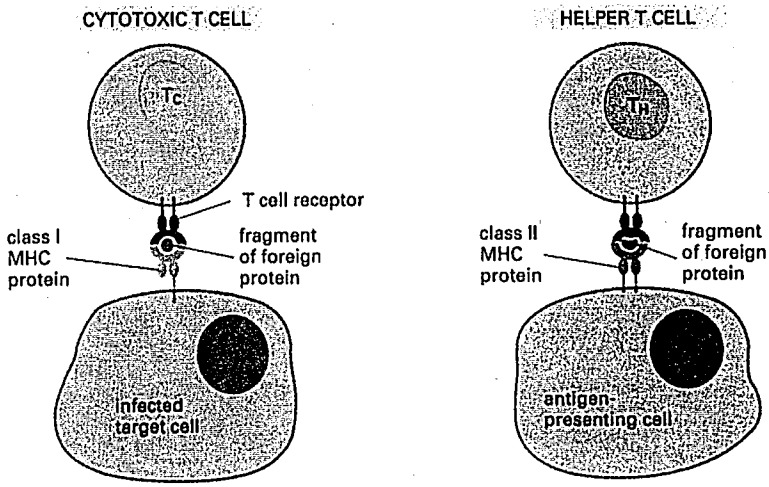

Figure 2.2. MHC Class I and Class II peptide complexes

The crystallized structure of a class I MHC molecule, human lymphocyte antigen A2 (HLA-A2) and many years of detailed molecular genetic analysis of T-cell/MHC interactions have answered very important questions in mind about the MHC molecules and their bindings to an unlimited number of antigens.

The structure of the HLA-A2 molecule is divided into two globular regions as shown in Figure 2.3 [17]. The α1 and α2 domains are viewed from the top of the molecule, showing the empty antigen binding site as well as the surface of that is presumably contacted by a T-cell receptor. One region is built up from the two immunoglobulin-like domains α3 and β_2m. Since α3 is attached to the membrane in the intact HLA molecule, this region is closest to the cell surface.

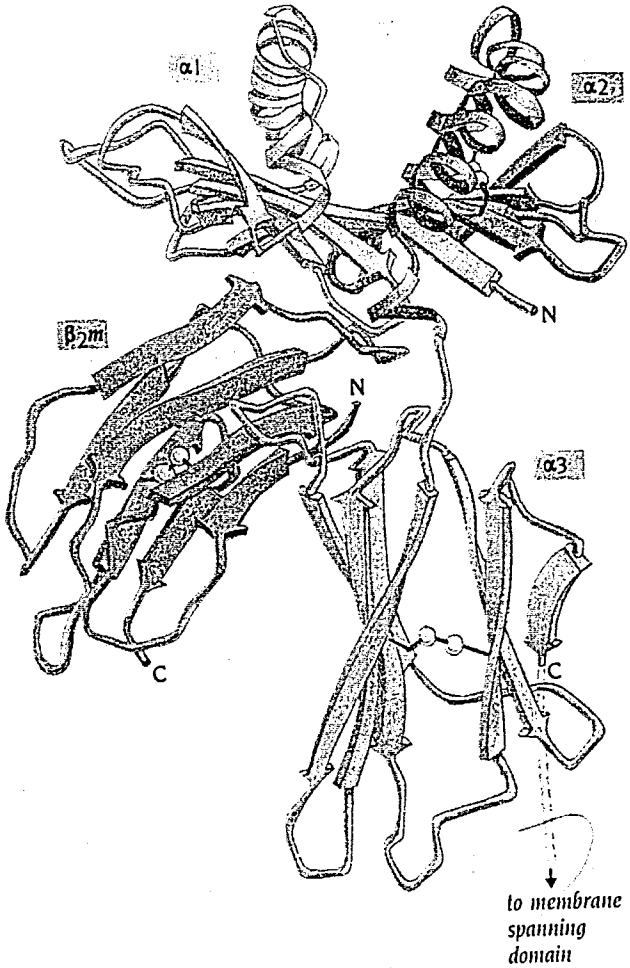

Figure 2.3. Schematic representation of the path of the polypeptide chain in the structure of HLA-A2

The second region, which presumably is facing the surface and which contains the antigen binding site, is composed of domains α1 and α2. Each domain has a very similar structure, which is quite simple as seen in Figure 2.4 [17]. Starting from the N terminus the chain forms four up-and-down antiparallel β strands called "W" followed by a helical region across the β sheet. Two domains associate by their "W" regions such a way that they are hydrogen bonded to each other in an antiparallel fashion. By this association the structure of the complete region has a "floor" of a continuous eight-strand antiparallel β sheet. This floor sits on top of the immunoglobulin-like domains α3 and β_2m. The two helical regions are above the floor almost parallel to each other and separated by a large distance, about 18 Å from center to center. A large crevice that faces the solution is thus formed with the floor forming its bottom and the helices it sides. This crevice is the antigen-binding site. In the actual structure that was determined, this crevice was occupied by an unknown antigen. Figure 2.5 [19] shows the crevice filled by a peptide of nine residues. The dimensions of the crevice, 25 Å long, 10 Å wide, and 11 Å deep, allow the binding of peptides of eight residues if they are in extended form and of about 20 residues if they are folded into an α helix [17].

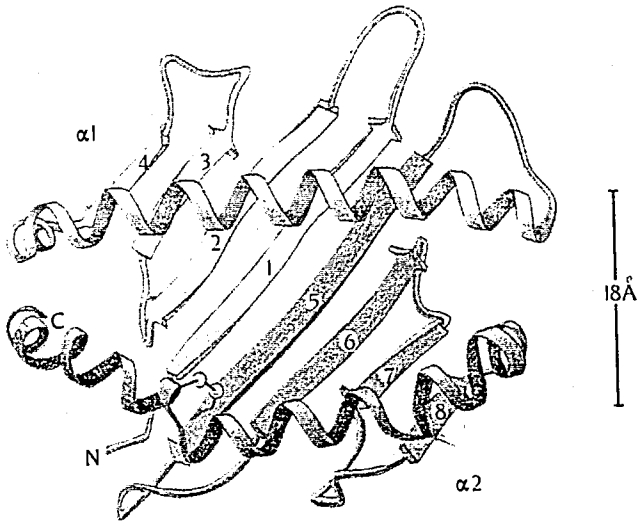

Figure 2.4. Schematic representation of the top surface of HLA-A2

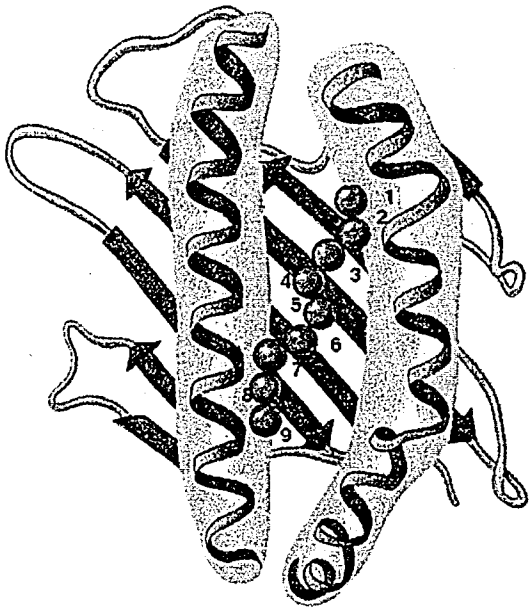

Figure 2.5. Peptide at the groove of HLA-A2 molecule

Class I MHC molecule's heavy chain is among the most genetically polymorphic protein known. The polymorphisms are due to a large number of point mutations at specific positions in the amino acid sequences of α1 and α2 domains. The studies on the structure of HLA-Aw68 molecule showed that the principal differences between the molecules of HLA-A2 and HLA-Aw68 are in the size and precise location of 13 amino acids that have been mutated. The studies on MHC molecules showed that the MHC molecules are mainly similar to each other with only size changes and amino acid differences at few positions. These few amino acid changes caused by mutation, cause major changes in specificity for side chains of peptide antigen [17].

2.1.2. Antigen Processing and Presentation

The APC degrades antigen into small peptides (processing), links the peptides to MHC molecules, and then expresses the MHC- peptide complex on its cell surface (presentation). Whether an antigen peptide binds preferentially to class I or class II MHC molecules depends upon how that antigen entered the cell. Antigens that are produced within the cytoplasm by endogenous processing of the APC tend to associate with class I MHC molecules. Endogenous antigens are proteins being produced within the human cell (endogenous processing), such as viral proteins produced during viral replication, proteins

produced by intracellular bacteria during their replication, and tumor antigens produced by cancer cells. Exogenous antigens are antigens that enter from outside the body (exogenous processing), such as bacteria, fungi, protozoa, and free viruses tend to associate with class II MHC molecules [20].

2.1.2.1. MHC Class I – Peptide Association (Endogenous Processing). Class I MHC molecules are thought to interact with peptides that have been degraded by proteasomes, part of a large cytoplasmic proteolytic complex called the low-molecular-mass polypeptide (LMP). The degraded peptides are carried into the rough endoplasmic reticulum (RER) by the transporter of antigenic peptides (TAP). Upon peptide binding, the interaction between a class I MHC α chain and β_2-microglobulin is stabilized and the complex is routed through the Golgi to the plasma membrane. The complex is now available for interaction with CD8 TC cells.

2.1.2.2. MHC Class II – Peptide Association (Exogenous Processing). Like class I molecules, class II molecules are synthesized in the RER. The class II α and β chains reside there as a complex with an additional polypeptide called the invariant chain (Ii). The presence of the invariant chain prevents peptide binding to class II MHC molecules within the endoplasmic reticulum and facilitates their routing to an endosomal compartment. Here in the endosomal or lysosomal compartment, there is a degraded antigen peptide. The invariant chain comes off the complex, exposes the groove of the class II molecule, and allows the antigen peptide to slip into the groove. The class II-antigen peptide complex is then transported to the surface of the APC where it is available for interaction with CD4 TH cells.

2.2. Historical Background

The identification of MHC ligands is a crucial step toward establishing T-cell-based immunotherapies for infectious diseases, auto-immune diseases and cancers. A vaccine for a disease is developed by using the viral peptides of that disease that bind to a specific MHC molecule within the infected cells, so that it will be presented to the cell surface and the infected cell is therefore be recognized and destroyed by the T or the B cells. The viral peptide that is known to bind to the MHC molecule, is used to produce attenuated live viral

vaccines capable of limited growth within the cytoplasm of infected cells, so that the body gets immunized to this disease. Therefore, knowing the specific MHC/peptide complexes, is a major issue in developing new vaccines.

There are many studies being carried by researchers on both class I and II MHC molecules. The following observations stated are obtained from studies on MHC class I molecules. Researchers have managed to divide the binding groove of MHC class I molecules into pockets [21,22], and further studies on these pockets resulted in the classification of pockets of different class I molecules into pocket families and superfamilies [23].

Studies have shown that, each individual has a limited number of different MHC proteins (e.g. six MHC molecules at most for a human), but hundreds of various peptides are displayed on the cell surface. Therefore, each protein must be able to display many different peptides [24]. On the other hand, the MHC molecule binds to peptides tightly, allowing one to assume that a very specific fit is required. Consequently, how can the MHC protein bind with high affinity for peptides but also be promiscuous in its binding specificity? The answer to this question lies in the groove of the molecule. Only a single antigen-binding site exists. If this groove acted as the lock in the very common lock and key model of many proteins, each MHC molecule could only bind to a single selected peptide. Instead, the studies on the MHC class I [22] showed that the groove consists of various pockets. A pocket is defined as the unit having an affinity for a corresponding peptide side chain. Some pockets have a well-shaped structure with an affinity for only one side chain. Other pockets have an affinity for a group of side chains, and sometimes the boundaries between the pockets are not clear. In order to locate the pockets of a molecule's binding groove, Matsumura and co-workers [22] calculated the solvent-accessible surfaces with probes of different radii. They were not only able to locate these pockets, but also find their varying depths. Six pockets, designated as A through F, were found. A, B, C, and F are considered deep pockets. Pockets D and E are considered shallow pockets. Some residues are involved in only one well-defined pocket, whereas others are actually shared by two or more pockets.

Pockets A and F play key roles in this binding. They are located at the ends of the groove and accommodate the NH_2 and COOH termini, respectively. The positioning of the NH_2 terminus is achieved by three, conserved tyrosine residues at positions 7, 159, and 171. The hydroxyl groups of these residues form hydrogen bonds with the amino group and the carbonyl oxygen of the peptide's first residue. The side chain of the first residue points upwards towards the solvent. Therefore, there is little restriction on the type of amino acids that can be accommodated by Pocket A [22].

Pocket F consists of a hydrophobic floor and a hydrophilic entrance. The carboxyl group of the last residue and the carbonyl oxygen of the penultimate residue form hydrogen bonds with the side chains of Tyr84, Thr143, Lys146, Try147, and Asp70. Unlike the residues accommodated by Pocket A, these amino acid side chains point toward the floor of the groove. Therefore, bulky aromatic residues are restricted from Pocket F [22].

Peptides recognized by the MHC class I molecules tend to be eight to 10 amino acids long. The ends of the peptide are bound to Pockets A and F, and some anchor residues may bind to the middle of the groove. Depending on the length of the peptide, a prominent bulging from the groove will occur. Longer peptides will have a more pronounced bulging from the middle of the groove [25]. The bulging may allow for recognition by and direct interaction with the T-cell receptors. The MHC class I molecule can bind to a variety of peptides, because it binds to the region that is common among peptides - the backbone (anchor positions), and it ignores the varying side chains of the peptides.

Now we will see how researchers have divided the pockets stated above into families and superfamilies. As mentioned before, peptides that bind class I MHC molecules are restricted in length and often contain key amino acids, anchor residues, at particular positions. The side-chains of peptide anchor residues interact with the polymorphic complementary pockets in MHC peptide-binding grooves and provide the molecular basis for allele-specific recognition of antigenic peptides. In their study [23] the researchers established correlation between class I MHC sequence markers that occur at the polymorphic positions lining structural pockets. They have analyzed the structures of nine crystallized class I MHC molecules and modeled structures of another 39 class I MHC

molecules, and showed that class I pockets can be classified into families that are distinguishable by their common physico-chemical properties and peptide side-chain selectivities. They have tested the correctness of their results on 20 other MHC class I molecules. Their study can be used to expand the repertoire of known, peptide-binding motifs of class I MHC molecules. Below, we will take a closer look to their study.

For peptides binding to a particular class I allele, some peptide positions (anchor positions) are primarily occupied by a particular residue or by a few closely related residues. These positions include P_2 frequently, which is contained in pocket B, P_3 infrequently, which is contained in pocket D, $P_{5/6}$ infrequently, which is contained in pockets C/D and P_Ω always, which is contained in pocket F. The occurrence of sequence patterns defined by anchor positions and anchor residues i.e. the peptide-binding motifs explains why each allelic form of class I molecule binds a broad, yet defined range of peptides.

The procedure they have followed in their study can be summarized as follows. Initially, sequence alignment and profile analysis of the experimentally identified peptides associated with each of the 68 class I MHCs were used to gather the amino acid occurrence frequencies at P_2, P_3, $P_{5/6}$ and P_Ω. For 48 class I MHC molecules that had six or more known peptide ligands, peptide anchor positions and associated anchor residues were obtained from the frequency data. The working assumption was that anchor residues identified at peptide positions P_2, P_3, $P_{5/6}$ and P_Ω will bind, respectively, pockets B, D, C/D and F. MHC residues forming these pockets were identified by an analysis of the solved crystal structures; conserved pocket residues delineate the framework, whereas polymorphic residues of a particular pocket determine its unique environment. The structural modeling of the polymorphic pocket side-chains and corresponding anchor side-chains demonstrated that the observed pocket specificities are derivative of the pocket structures. The observation that groups of MHC pockets exhibit similar physical properties and peptide side-chain selectivities led to a natural classification of the 48 class I MHC pockets into distinct pocket families. MHC sequence markers and consensus peptide anchors were identified for each pocket family. The sequence markers were used to allocate the pockets of the other 20 class I molecules, which had fewer than six known peptide ligands, and the accuracy of using the family consensus anchors as a prediction of

the anchors for the unknown class I molecules was assessed. The test results of this study were promising, and widely being used to search a protein sequence for peptides that fit the class I-binding motifs.

3. GENETIC ALGORITHMS

3.1. An Overview of GAs

3.1.1. Historical Background

Evolutionary Algorithms have been independently studied with the idea that evolution could be used as an optimization tool for engineering problems by several computer scientists during 1950s and 1960s. By using operators inspired by natural genetic variation and natural selection in their systems, researchers tried to evolve a population of candidate solutions to a given problem.

Evolutionary algorithms evolved during last 30 years are a branch of Guided Random Search Techniques, which are based on enumerative techniques but use additional information to guide the search. Evolutionary algorithms are based on natural selection principles. This form of search evolves throughout generations improving the features of potential solutions by means of biologically inspired operations. They can be divided in three main categories: Evolutionary Strategies, developed by Rechenberg [26] in 1973 and further developed by Schwefer [27] in 1977, Evolutionary Programming, developed by Fogel and co-workers [28] in 1966; and GAs, developed by Holland [29] in 1975. Holland's original goal was to formally study the phenomenon of adaptation as it occurs in nature and to develop ways in which the mechanisms of natural adaptation might be imported into computer systems rather then to design algorithms to solve specific problems.

Evolutionary strategies use mutations as search mechanisms and selection to direct the search toward the prospective regions in the search space. Evolutionary programming is a technique in which candidate solutions to given tasks were represented as finite-state machines, which were evolved by randomly mutating their state-transition diagrams and selecting the fittest. GA generates a sequence of populations by using a selection mechanism, and use crossover and mutation as search mechanisms. The principal difference between GAs and the other two is that GAs rely on crossover, a mechanism of

probabilistic and useful exchange of information among solutions, to locate better solutions, while evolutionary strategies and evolutionary programming use mutation as their primary search mechanism.

Holland's GA is a method for moving from one population of chromosomes (eg. strings of bits) to a new population by using a kind of natural selection together with the genetic-inspired operators of crossover, mutation and inversion. The working of GA is based on the schema theory and the building block hypothesis of Holland [29] and Goldberg [30], which will be explained in later sections.

3.1.2. Biological Terminology

The biological terms used in GAs are borrowed from biology, but they usually refer to the entities that are much simpler than the real biological ones as stated below [8,31].

The information that is necessary to build each protein or RNA (Ribonucleic acid) found in an organism is encoded in DNA (Deoxyribonucleic acid) molecules. For this reason, DNA is sometimes referred to as the "blueprint of life". All living organisms consist of cells, and each cell of an organism has a few long DNA molecules, called chromosomes. Certain contiguous stretches along a DNA molecule encode information for building proteins and to each kind of protein in an organism, there is usually only one contiguous stretch called a gene. Each gene is located at a particular locus (position) on the chromosome. A gene can be considered as encoding a trait, such as eye color. The different possible settings for a trait (e.g. blue, brown) are called alleles.

Complete set of chromosomes inside a cell is called a genome. The number of chromosomes in a genome is characteristic of a species. The particular set of genes contained in a genome is called genotype. Under fetal and later development, the genotype gives rise to the organism's phenotype, which forms the physical and mental characteristics of the organism.

Organisms whose chromosomes are in pairs are called diploid; organisms whose chromosomes are unpaired are called haploid. During sexual reproduction, recombination

(crossover) occurs: in each parent, genes are exchanged between each pair of chromosomes to form a gamete (a single chromosome), and then gametes from the two parents pair up to create a full set of diploid chromosomes. In haploid sexual reproduction, genes are exchanged between the two parents' single-strand chromosomes. Offspring are subject to mutation, in which single nucleotides (elementary bits of DNA) are changed from parent to offspring, the changes often resulting from copying errors. The fitness of an organism is typically defined as the probability that the organism will live to reproduce, or as a function of the number of offspring the organism has.

In GAs, chromosome is a candidate solution to a problem, and it is often encoded as a bit string. The genes are either single bits or short blocks of adjacent bits that encode a particular element of the candidate solution. An allele in a bit string is either 0 or 1. Crossover is the exchange of genetic material between two single-chromosome haploid parents. Mutation is the flipping of a bit at a randomly chosen locus, or for larger alphabets, replacing a symbol at a randomly chosen locus with a randomly chosen new symbol.

Most applications of GA employ single-chromosome individuals, The genotype of an individual in a GA using bit strings is simply the configuration of bits in that individual's chromosome, and often there is no notion of phenotype in the context of GAs.

3.1.3. Elements of GA

GAs are constructed by the following elements: a population of chromosomes, selection according to a fitness criterion, crossover to produce new population and mutation to change a chromosome randomly.

A chromosome is usually encoded as a string of bits and each locus of a chromosome takes value either 0 or 1. Each chromosome is a potential solution in the entire search space of candidate solutions.

GA requires a fitness function that assigns a fitness score to each chromosome, depending on how well that chromosome solves the problem in consideration. The fitness

function depends on the problem at hand and its definition should be done correctly in order for GA to perform well.

There are three basic operators of GA: selection, crossover (recombination) and mutation. Selection operator selects chromosomes that will join the reproduction process. High fitness value gives a chromosome more chance of being selected for mating. Crossover operator randomly chooses a locus, and exchanges subsequences of randomly selected two chromosomes after that locus. GAs assume that high-quality parent candidate solutions from different regions in the space can be combined via crossover to, on occasion, produce high-quality offspring candidate solutions. Finally, mutation operator randomly changes the value of bits in a chromosome. Mutation is done for the chance of regeneration of lost good solutions that may not be obtained again with the crossover operator.

3.1.4. A Simple Genetic Algorithm

In a simple genetic algorithm (SGA), binary valued encoding is done, i.e. the chromosomes are encoded as strings of bits. The structure of a SGA can be seen in Figure 3.1. At the beginning of computation, a population is randomly generated. The objective function (fitness function) is then evaluated for each individual in the population. The optimization criteria are checked. If the optimization criteria are not met, creation of new generation is started. Selection is done according to the fitness values of the individuals, then parents are crossed to produce offspring and at last, all offspring will be mutated with a certain probability. The fitness of newly generated population is computed after replacing the parents with the offspring. This cycle is performed until the optimization criteria are met or a certain number of iterations achieved.

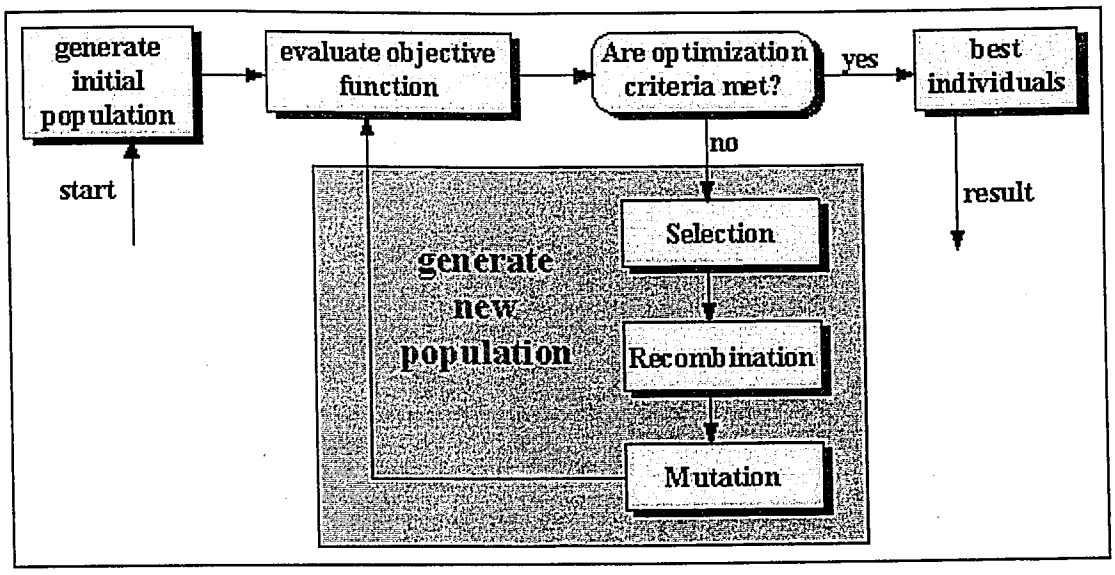

Figure 3.1. Structure of a simple genetic algorithm

The selection mechanism of a SGA is based on fitness-proportionate selection method, in which the number of times an individual is expected to reproduce is equal to its fitness divided by the average fitness of the population. A simple method of implementing fitness-proportionate selection is roulette-wheel selection [30], which is conceptually equivalent to giving each individual a slice of a circular roulette wheel equal in area to the individual's fitness, of whose details will be given in the implementation phase.

The crossover is done by randomly selecting two individuals from the mating pool after selection operation performed, and crossing them with a crossover probability. SGA uses one-point crossover mechanism, where a cross-site is randomly selected and the portions of each individual after that site is exchanged.

In SGA, the generation of new population is performed by completely replacing the old population with the new population obtained after selection, crossover and mutation.

There are many selection, crossover and reinsertion techniques that can be implemented in order to improve the SGA's performance. Some of these techniques that were used in our implementation will be explained in Section 4.

3.1.5. Schema Theory and Building Blocks

Despite the successful use of GAs in a large number of optimization problems, progress on the theoretical front end has been rather slow. A very clear picture of the working of GA has not yet emerged, but the schema theory and the building-block hypothesis of Holland [29] and Goldberg [30] capture the essence of GA mechanics.

A schema is a similarity template describing a subset of strings with similarities at certain positions. A schema represents a subset of all possible strings that have the same bits at certain string positions. For example the schema #1101#0 represents the set of strings {1110110, 1110100, 0110110, 0110100}, where each string is an instance of the schema. Here the sign "#" represents the "don't care" bit which can represent either 0 or 1. The number of fixed positions (0s and 1s) of a schema is its order. In our example, the order is five. Defining length of a schema is the distance between the outermost fixed positions, again five in our case. Any specific string is simultaneously an instance of 2^l possible schemata, where l is the length of the string. A fitness value can be assigned to a schema, which is the average fitness of the schema and calculated as the average fitness of instances of the schema, and this fitness value varies with the population's composition from one generation to another.

Consider a schema with k fixed positions. There are 2^k distinct schemata that generate a partitioning of all possible strings. Each such set of k fixed positions generates a schema competition, a survival competition among the 2^k schemata. Since there are 2^l possible combinations of fixed positions, 2^l distinct schema competitions are possible. Therefore, the execution of GA generates 2^l schema competitions and locates the best schema for each set of fixed positions. GAs search can be visualized as a search for the optimal string as a simultaneous competition among schemata to increase the number of their instances in the population. The schema with high fitness value and small defining length is called a building block and the winner of the competition is the optimal string of that schema. The notion that the strings with high fitness values can be located by sampling building blocks with high fitness values and combining the building blocks effectively is called the building-block hypothesis [29,30].

The genetic operators generate, promote and juxtapose building blocks to form optimal strings. Crossover tends to conserve building blocks whereas mutation tends to generate new building blocks. Selection provides the increase in representation of high fitness valued blocks from generation to generation. The building-block hypothesis assumes that the juxtaposition of good building blocks yields good strings.

A high average fitness value grows exponentially to win the competition, but a high average fitness value alone is not sufficient for a high growth rate. A schema must have a short defining length, because higher the defining length of a schema, the higher probability that the crossover point will fall between its fixed positions and an instance will be destroyed. Thus, the schemata with high fitness values and small defining lengths grow exponentially with time. This is the essence of the schema theorem first proposed by Holland [29], as the fundamental theorem of GAs. In (3.1), you can see the formal statement of the schema theorem.

$$N(h,t+1) \geq N(h,t)\frac{f(h,t)}{\bar{f}(t)}\left[1 - p_c \frac{\delta(h)}{l-1} - p_m o(h)\right] \tag{3.1}$$

where;

$f(h,t)$	: average fitness value of schema h in generation t
$\bar{f}(t)$	: average fitness value of the population in generation t
p_c	: crossover probability
p_m	: mutation probability
$\delta(h)$	: defining length of the schema
$o(h)$	: order of the schema h
$N(h,t)$	: expected number of instances of schema h in generation t
l	: the number of bit positions in a string
$p_c \frac{\delta(h)}{l-1}$	: the probability that an instance of the schema h is disrupted by crossover
$p_m o(h)$	: the probability that an instance is disrupted by mutation

3.2. Applications of GA in Molecular Biology

GAs are widely used in many areas of molecular biology. Some of these applications will be stated below.

3.2.1. Fragment Assembly of DNA

The sequencing of whole DNA molecule directly is not possible with today's technology. Instead, the fragments of DNA molecules obtained from the studies in laboratories (through DNA cutting, breaking and cloning) are sequenced (their nucleic acid sequence obtained) separately and then they should be combined in order to obtain the whole DNA molecule. The whole DNA sequence can be obtained by positioning fragments so that they align well with each other. The overall aligned sequence is the consensus sequence.

Parsons and co-workers [3] used GA in order to solve the fragment assembly problem. They have used two fitness functions, which both performed well for different encoding techniques and operators. The encoding used is the sorted-order representation. This representation provides a rather complex mapping from the individual to the permutation encoding. The two requisite properties for a legal ordering are that all fragments be present in the ordering and that there be no duplication in the ordering. Both uniform and two-point crossover methods are used for recombination, and they have found that uniform crossover does not perform well for this problem. They have also tried special operators such as edge-recombination, order crossover, and swapping instead of mutation.

Each position in a chromosome represented a fragment, and their goal was to find the optimum ordering and alignment of fragments that will give the highest fitness value. They have concluded that GA performs quite well with the appropriate representation and operator set both in terms of speed and solution quality.

3.2.2. Protein Folding Problem

Finding the 3-D geometry or tertiary structure of an arbitrary protein is vital to understanding the functionality of that protein. The prediction of this structure is known as the protein folding problem, and it is very difficult and has been labeled as one of the grand challenge problems for the scientific community.

Currently, biochemists use techniques such as MRI (magnetic resonance imaging) and X-ray crystallography on protein crystals in order to view the conformation of a protein. These techniques are expensive, in terms of equipment, computation and time. Additionally, both of these techniques require isolation, purification, and crystallization of the target protein, which may be difficult or impossible depending upon the particular protein under study, therefore faster, and cheaper methods are required for efficient studies on proteins.

Patton and co-workers [32], used GAs in order to solve this problem. Their approach to encoding an individual folded state for a protein was to represent a single relative move for each peptide in the protein. Thus each peptide has five possible values (up, down, right, left, and forward) encoded in seven bits rather than three, in order to reduce the number of illegal states searched. Thus, each peptide was represented with an ordered list of preferred directions.

An individual genotype is evaluated under each encoding by plotting the course encoded by the genotypic movement list in a 3-D lattice. After a protein is plotted, each occupied cell is tested for the presence of a hydrophobe. If one is present and no additional peptides (collision) are detected in the cell, all adjacent points are tested for the presence of a hydrophobe, and a single point is awarded each contact. At the same time, each cell is tested for the presence of multiple peptides, and if there are, they are penalized.

They have tried different operators and parameters in order to find the best performing algorithm. Two-point crossover and bit-mutation used as operators. The results of their work have shown that they have achieved very good performance compared to previous works on this problem.

3.2.3. Drug Design

GAs can be used in order to design new drugs. The central application task is the prediction of the biological activity of a novel compound given a collection of similar compounds with known activities. One approach can be stated in a form known as quantitative structure-activity relationship (QSAR) analysis, in which the activity is modeled by constructing a mathematical relationship between activity and the physicochemical properties of the series of compounds.

Rogers [33] has developed a genetic algorithm called genetic function approximation (GFA) algorithm in the area of QSAR modeling, which can be used by pharmaceutical companies for drug design.

QSAR models are represented as sums of linear or nonlinear terms. The model is (3.2). The terms are called basis functions, and are denoted ϕ_k, these are functions of one or more features, such as $(X_5\text{-}10)^2$, $\text{Sin}(X_1)*\text{Sin}(X_2)$, or X_{10}, where the X_i's are the feature measures. The model coefficients a_k are determined using least-squares regression or another suitable fitting technique. The linear strings of basis functions play the role of the DNA for the application of genetic algorithm.

$$F(X) = a_0 + \sum_{k=1}^{M} a_k \phi_k(X) \tag{3.2}$$

The initial QSAR models are generated by randomly selecting some number of features from the training data set, building basis functions from these features using the user-specified basis function types, and then constructing the genetic models from random sequences of these basis functions. Each model of the population, that is the individual is represented as a linear string of basis functions. F_1: {LOGP; DIPV_Y; VOL; $(\text{LOGP-5.1})^2$} is an example individual form standard QSAR data set.

The fitness function used during the evolution is derived from Freidman's lack-of fit scoring function, which is a penalized least-squares error measure. This measure balances the decrease in error as more basis functions are added against a penalty related to number

of basis functions and complexity of the model. The goal is to maximize the fitness function.

After initial population generation, genetic crossover is performed repeatedly. Two good models are probabilistically selected as parents, proportional to their fitness. Each is randomly cut into two sections. The cuts occur between the basis functions. A new model is created using the basis functions taken from a section of each parent. Optional mutation operators may alter the newly-created model. The model with the worst fitness is replaced by this new model. Algorithm stops when the average fitness of the models in the population stops improving. GFA analysis generates a population of solution models instead of presenting the user with a single best model.

The combinations of models discovered by GFA algorithm yields models, which are more predictive than the combinations of features discovered by either the forward-stepping regression, stepwise regression, or the neural-network technique. This is because the genetic algorithm specifically searches for combinations of features, which score well, rather than trying to identify individual feature. Tests on the algorithm showed that the GFA algorithm was able to meet or surpass the quality of models discovered through standard methods in use in QSAR, and to provide additional information, such as multiple models, automatic estimation of model size, and population statistics of feature use that are not available from other techniques. The GFA algorithm is proved to be better at discovering combinations of basis functions that take advantage of correlation only available in combination.

4. PROBLEM DEFINITION AND METHODOLOGY

In this section, the implementation details will be discussed. In our implementation, initially SGA is implemented and then different modifications to SGA's selection mechanism, and operators are performed in order to increase GA's performance. The fitness function of each individual is evaluated using regression and correlation analysis.

4.1. Problem Definition

In the MHC-Peptide problem, we have a set of MHC molecules with known amino acid sequences, and for each of these MHC molecules we have a set of peptide amino acid sequences that are known to bind to them. Our peptide sequences are eight to 10 residues long and our problem is to find the set of MHC amino acid positions, which plays important role in binding to a peptide for each peptide position separately and therefore the peptide motifs of the corresponding MHC molecules.

The set of MHC molecule positions for each peptide position will be represented in a form of equation (4.1), where $\hat{Y}$ value represents the peptide residue (amino acid), and X_m's represent the MHC residue at m^{th} position. Here b_m's are the parameters of the equation, b_0 is the constant term and VAR is the number of independent variables.

$$\hat{Y} = b_0 + \sum_{m=1}^{VAR} b_m X_m \tag{4.1}$$

In order to find the parameters, b_m a curve-fitting technique, regression analysis is performed. The "best fit" equation in the population is determined through correlation analysis. Both of these techniques will be stated in the following sections. The best set of positions, i.e. the best equation is the one that the strength of the equation is maximized, that is the error in curve-fitting is minimized.

4.2. Encoding

Although SGA uses binary encoding of chromosomes, in our implementation we have used integer encoding since it is more appropriate for our problem. In this encoding, each bit of a chromosome will be an integer from one to VART, i.e. the total number of MHC positions to be considered. Each chromosome will be composed of VAR MHC positions, where VAR is a parameter of our implementation. Each chromosome is of the form of equation (4.1) where X_ms range from one to VART as stated above. An example of a chromosome can be seen in Figure 4.1, where VART is 20 and VAR is six in this case.

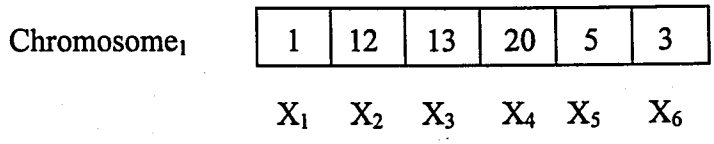

Figure 4.1. A chromosome with integer encoding

In our encoding, the ordering of bits in a chromosome does not matter, but duplicates are not allowed, since each bit represents a MHC position and should appear only once in a solution set.

4.3. Genetic Algorithm Implementation

Figure 4.2 shows the genetic algorithm used throughout our implementation phase. Initially a random population is generated and evaluated. Then the parents that will be involved in mating phase are selected according to their fitness. After generation of mating pool, the individuals are randomly paired for mating. The next step in the algorithm is the mating, where certain parts of paired individuals are exchanged with the hope of generating fitter individuals for next generation. After mating, mutation occurs with a certain probability that alters some of the bits of individuals. Then the new individuals are evaluated and reinserted into the next generation with a reinsertion mechanism. These steps will be explained in detail in the following sections.

```
Algorithm Genetic_Algorithm
    Initialize population;          // Random initialization of individuals
    Evaluate population;            // Regression and Correlation Analysis
    while not stopping condition reached
            Select mating pool;     // Selection Phase
            Recombination;          // Crossover Phase
            Mutation;               // Mutation Phase
            Evaluate population;    // Regression and Correlation Analysis
            Reinsertion;            // Determination of new population for
                                       next generation
    end while
end Genetic_Algorithm
```

Figure 4.2. Genetic algorithm

4.3.1. Initialization of the Population

The first step is the initialization of a population of individuals. As we have mentioned before, we have to choose VAR variables out of total VART variables for each individual. Each individual may contain some common variables with another individual, but two individuals cannot be exactly the same, i.e. no duplication at the initial phase allowed. This is for examining as many solutions as possible at the initial step. The number of individuals in the population is a parameter, labeled as POP. The initialization of the population is done at random, and we have a population like shown in Figure 4.3.

P_1	: 1 2 3 4 5 6	VART	: 10
P_2	: 5 6 7 8 9 10	VAR	: 6
P_3	: 2 5 7 1 9 10	POP	: 3

Figure 4.3. An example of a population of individuals

The parameters to our algorithm are given in Table 4.1. These are the parameters that will be optimized in order to increase the GA performance.

Table 4.1. Parameters of the genetic algorithm

Parameter Name	Definition
POP	Number of individuals in the population
VAR	Number of independent variables in an individual
VART	Total number independent variables available
M	Number of regression coefficients + the constant term
N	Number of input samples for algorithm training
Cross_prob	Crossover probability
Mutat_prob	Mutation probability

4.3.2. Regression and Correlation Analysis

After the initialization of our population, the next step is to evaluate the individuals' fitness values according to a fitness function. Our fitness function is the coefficient of determination, R^2, which is calculated through regression and correlation analysis.

As stated in the problem definition, we have a set of Y values where Y is the dependent variable to our problem and these values represent the hydrophobicity or size score of a peptide at a given position. Also we have a corresponding X_m set for each Y where X_m's are the independent variables to our problem and these values represent the hydrophobicty score or size of the MHC molecule at the m^{th} position. Here regression analysis is done to describe the nature of the relationship between the dependent variable's data and the independent variables' data, whereas the correlation analysis is done to investigate the strength of the relationship defined by regression. Regression analysis is concerned with the problem of estimating the value of one variable, called the dependent variable, on the basis of one or more other variables, called independent variables [34]. The strength of the relationship among the dependent and independent variables will be our fitness function, and our aim is to maximize the fitness function, and therefore maximize the relationship strength of our input data so that we can make powerful predictions.

4.3.2.1. Regression Analysis. Regression is a curve-fitting technique that gets a set of dependent and independent variables' data as input and produces an equation that best represents the input data. If the number of independent variable is one, than this is called simple regression. On the other hand, if we have more than one dependent variable, this is

called multiple regression. The solution technique of multiple regression is an extension of the simple one. Depending on the relationship among the dependent and independent variables, the regression can be either linear or nonlinear.

In our implementation, we looked for a linear relationship among our input data. Our regression equation was stated in (4.1). The regression model is (4.2). Here, $\hat{Y}_i$ is the predicted value and note that Y_i is the actual value of Y in (4.3), at input line i that ranges from one to N in our case. b_ms are the regression coefficients, b_0 is the constant term, and X_{mi} is the X value at position m of the MHC molecule at i^{th} line. Finally, e_i is the regression error defined in (4.3), which is the difference between the predicted value $\hat{Y}_i$ and the actual value Y_i.

$$\hat{Y}_i = b_0 + \sum_{m=1}^{VAR} b_m X_{mi} + e_i \tag{4.2}$$

$$e_i = Y_i - \hat{Y}_i \tag{4.3}$$

The approach almost universally adopted to find the line of best fit in regression analysis is to determine the values of b's, which minimize the sum of squared errors. This procedure is the method of least squares. The method is defined as (4.4). After applying this method, we have obtained the regression parameters and therefore the regression equation of (4.1).

$$Minimize \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i\right)^2 \tag{4.4}$$

The method of regression described above can further be extended for introducing nonlienar terms into the regression equation.

4.3.2.2. Correlation Analysis. After determining the regression coefficients, the next step is to determine the fitness of that equation. Our fitness function is the coefficient of determination, R^2, which is expressed as (4.5). The term SSR is the explained variation,

and the term SST is the total variation shown in (4.6). SSE in (4.6) is the unexplained variation and $\overline{Y}$ is the mean of the actual Y values.

$$R^2 = \frac{SSR}{SST}, \quad 0 \le R^2 \le 1 \tag{4.5}$$

$$\overset{SST}{\sum_{i=1}^{N}\left(Y_i - \overline{Y}\right)^2} = \overset{SSE}{\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2} + \overset{SSR}{\sum_{i=1}^{N}\left(\hat{Y}_i - \overline{Y}\right)^2} \tag{4.6}$$

R^2 shows that the regression equation derived explains R^2 per cent of the input data. The higher the R^2 value is, the better fit is made. One important feature of multiple regression analysis is that, as you increase the number of independent variables, R^2 value increases. To eliminate this, R^2 value is adjusted to Adj-R^2 as shown in (4.7). We have used Adj-R^2 as our fitness function throughout our implementation.

$$Fitness\_Function: \quad Adj - R^2 = 1 - \frac{\left(1 - R^2\right) * \left(N - 1\right)}{N - M} \tag{4.7}$$

4.3.2.3. Matrix Representation. In order to implement the regression analysis, a matrix representation is used. The dependent variable, Y's data and the independent variables', X's data are represented in matrix notation, and the unknown regression parameters are denoted as B matrix, and are calculated using matrix operations such as multiplication, transposition, and inversion. (4.8) shows the matrix representation of our regression equation.

$$Y_{Nx1} = X_{NxM} * B_{Mx1} \tag{4.8}$$

4.3.3. Selection

After the evaluation of the fitness values of individuals in the population, the next step is the selection of the mating pool, i.e. the set of individuals that will mate (recombine). Selection determines which individuals are chosen for mating and how many

offspring each selected individual produces. The purpose of selection is to emphasize the fitter individuals in the population in hopes that their offspring will in turn have even higher fitness. Selection has to be balanced with variation from crossover and mutation. Too strong selection means sub-optimal highly fit individuals will take over the population, reducing the diversity needed for further change and progress; on the other hand, too weak selection will result in too slow evolution.

Numerous selection schemes have been proposed in the GA literature. Some of the most common of these selection mechanisms together with the ones we have used in our implementation are explained below.

4.3.3.1. Roulette-wheel Selection. This is a stochastic algorithm and it is also called stochastic sampling with replacement. This selection mechanism is a fitness-proportionate selection. Fitness-proportionate selection is based on the expected number of times an individual will be selected to reproduce, and this number is the individual's fitness divided by the average fitness of the population.

In roulette-wheel selection, each individual is assigned a slice of circular roulette wheel. The size of the slice is proportional to the individual's fitness. The wheel is spun as much as the number of individuals in the population. On each spin, the individual under the wheel's marker is selected and put into the mating pool for recombination.

The roulette-wheel selection algorithm provides zero bias but it does not guarantee minimum spread. The algorithm can be seen in Figure 4.4. This algorithm is implemented in our study since it is the most widely used selection mechanism, also included in SGA.

```
Algorithm Roulette_Wheel_Selection
   Compute total_fitness;          // total fitness of population
   for each individual i           // from 1 to POP
         Compute sel_prob(i);      // Divide i^th individual's fitness to total fitness
                                     (selection probability of each individual)
   for each individual i           // from 1 to POP
         Choose r;                 // 0 ≤ r ≤ 1, floating point number
         sum initialized to 0;
         while sum<r
               sum=sum+sel_prob(i);   // add selection probability of each individual
         select i;             // Select individual whose addition to sum exceeds or
                                  becomes equal to r
   end for;
end Roulette_Wheel_Selection
```

Figure 4.4. Roulette-wheel algorithm

4.3.3.2. Stochastic Universal Sampling. This selection mechanism is also fitness-proportionate. Stochastic universal sampling is similar to roulette-wheel selection, but instead of selecting the individuals one by one, it selects all at once.

The algorithm can be seen in Figure 4.5. This algorithm provides zero bias and minimum spread (the range of possible actual values, given an expected value) by rather than spinning the roulette POP times to select POP individuals, spinning it once, but with POP equally spaced pointers. This algorithm is also implemented in our study.

```
Algorithm Stochastic_Universal_Sampling
      Compute total_fitness;          // total fitness of population
      for each individual i           // from 1 to POP
            Compute sel_prob(i);      // Divide i^th individual's fitness to total fitness (selection
                                        probability of each individual)
      dist = 1/POP;                   // distance between the POP pointers.
      Choose r;                       // 0 ≤ r ≤ dist, floating point number
      for each individual i           // from 1 to POP
            sum initialized to 0;
            while sum<r
                  sum=sum+sel_prob(i); // add selection probability of each individual
            select i;       // Select individual whose addition to sum exceeds or
                               becomes equal to r
            r=r+dist;       // move to next pointer
      end for;
end Stochastic_Universal_Sampling
```

Figure 4.5. Stochastic universal sampling algorithm

4.3.3.3. Tournament Selection. In this selection, a number Tour of individuals is chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated as often as individuals to choose. The algorithm parameter, Tour ranges between 2 – POP. Usually Tour is selected as two as we did in our implementation. We have also tried three. The algorithm is in Figure 4.6.

```
Algorithm Tournament_Selection
        for each individual i
                Select Tour individuals;
                Select the one with highest fitness;
        end for;
end Tournament_Selection
```

Figure 4.6. Tournament selection algorithm

4.3.3.4. Truncation Selection. This is an artificial selection method. In this selection, individuals are sorted according to their fitness. Only the best individuals are selected for parents. The parameter for truncation selection is the truncation threshold Trunc. Trunc indicates the proportion of population to be selected as parents and takes values ranging from 50 to 10 per cent. Individuals below the truncation threshold do not produce offspring. This kind of selection may easily cause loss of diversity, and therefore premature convergence. In order to avoid early convergence, the mutation rate should be kept high. This selection is not suitable for our implementation since it favors only the individuals whose fitness are high and eliminates individuals with low fitness, which may be the potential solutions in future generations, so we did not include it.

4.3.3.5. Rank Selection. In order to prevent too quick convergence, this method is an alternative. The individuals in the population are ranked according to fitness, and the expected value of each individual depends on its rank rather than on its absolute fitness. In some cases it might be important to know that one individual is far fitter than its nearest competitor, so this method is not applicable. This also holds for our case, since small differences in fitness values play important role for our solution.

4.3.4. Recombination

The completion of selection described in the previous section resulted in the population of individuals that will mate. Recombination phase is the step in which the individuals will be randomly divided into two groups and get crossed.

Before crossover operation is performed, the individuals in the mating pool are randomly paired, so that these selected pairs can be crossed at recombination phase. Each pair is crossed based on a crossover probability, cross_prob.

There are a number of different recombination mechanisms present in the literature. Here we will describe some of the most popular ones including the ones that we have implemented.

4.3.4.1. Single-point Crossover. This is the simplest form of recombination. In single point crossover, a single crossover position is chosen at random in the range from one to VAR-1, where VAR is the number of variables in an individual. The variables after the selected crossover position are exchanged between the individuals. We have applied a modified version of single-point crossover in our implementation. Since we do not allow duplication of variables in an individual, before selecting the cross-site, first we have to isolate the variables that are the same in both individuals. Then a cross-site is selected and the remaining elements of the individuals are crossed. Figure 4.7 shows the algorithm we have used, and Figure 4.8 shows an example of a single-point crossover.

```
Algorithm Single_point_Crossover
  Select mating pairs;              // Random pairing of individuals for mating
  for each mating pair i;           // i ranges from 1 to POP/2
    select prob;                    // probability of individual for crossing
    if prob ≤ cross_prob            // do crossover
        Separate common variables;  // not included in crossing
        Select a cross-site cs;
        Exchange two parents after cs;
    end if;
  end for;
end Single_point_Crossover
```

Figure 4.7. Single-point crossover algorithm

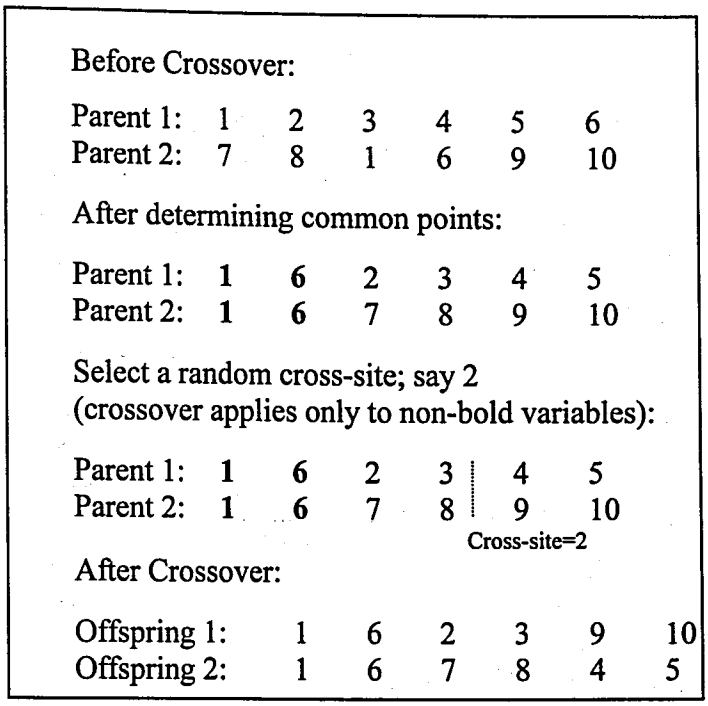

Figure 4.8. An example of a single-point crossover

4.3.4.2. Multi-point Crossover. For multi-point crossover, m crossover positions ranging from one to VAR-1 are chosen at random with no duplicates and sorted in ascending order. Then, the variables between successive crossover points are exchanged between two parents to produce two new offspring. The section between the first variable and the first crossover point is not exchanged between individuals. m is usually selected as two, this is called two-point crossover. The algorithm can be seen in Figure 4.9. Figure 4.10 illustrates this process on two binary encoded individuals where m is 3. We have implemented two-point crossover in our study.

```
Algorithm Two_point_Crossover
      Select mating pairs;        // Random pairing of individuals for mating
      for each mating pair i;     // i ranges from 1 to POP/2
            select prob;          // probability of individual for crossing
            if prob ≤ cross_prob        // do crossover
                  Separate common variables;    // not included in crossing
                  Select two cross-sites cs1, cs2;
                  Exchange parts between cs1 and cs2;
            end if;
      end for;
end Two_point_Crossover
```

Figure 4.9. Two-point crossover algorithm

Before Crossover:

Parent 1:	0	1	1	1	0	0	1	1	0	1	0
Parent 2:	1	0	1	0	1	1	0	0	1	0	1

Select cross-sites, say 2, 6 and 10 in this case where m=3;

After Crossover:

Offspring 1:	0	1	1	0	1	1	0	1	1	1	1
Offspring 2:	1	0	1	1	0	0	0	0	1	0	0
			cs: 2				cs: 6				cs: 10

Figure 4.10. An example of multi-point (3-point) crossover

4.3.4.3. Discrete Recombination. Discrete recombination makes every position in an individual a potential crossover point. A crossover mask, same length as the individual structure is created at random and the parity of the bits in the mask indicate which parent will supply the offspring with which bits. The algorithm shown in Figure 4.11 is implemented in our study. See Figure 4.12 for an example of this technique.

```
Algorithm Discrete_Recombination
    Select mating pairs; // Random pairing of individuals for mating
    for each mating pair i              // i ranges from 1 to POP/2
        select prob; // probability of individual for crossing
        if prob ≤ cross_prob  // do crossover
            Separate common variables;    // not included in crossing
            for offspring 1 and 2
                for each differing bits j
                    Select r; // r is 1 for parent 1 and 2 for parent 2
                    if r equal to 1
                        Select variable from parent 1;
                    else
                        Select variable from parent 2;
                end for;
            end for;
        end if;
    end for;
end Discrete_Recombination
```

Figure 4.11. Discrete recombination algorithm

Before Crossover:

Parent 1:	1	2	3	4	5	6
Parent 2:	7	8	1	6	9	10

After determining common points:

Parent 1:	**1**	**6**	2	3	4	5
Parent 2:	**1**	**6**	7	8	9	10

Generate Mask (for variable selection):

Sample 1:	1	1	2	1
Sample 2:	2	1	1	2

After Crossover:

Offspring 1:	1	6	2	3	9	5
Offspring 2:	1	6	7	3	4	10

Figure 4.12. An example of discrete recombination

4.3.5. Mutation

After recombination, offspring undergo mutation. Offspring variables are mutated with a low probability. The probability of mutating a variable, mutat_prob, is set inversely proportional to the number of variables. The more variables one individual has, as smaller is the mutat_prob.

Each variable in each individual has the chance of being mutated. Mutating a variable means in general, changing that variable with another one in the variable set. For every individual, the variable to change is chosen at random.

In our implementation, we have defined a mutation rate, for each individual's variables, we have generated a floating-point number *r*. If the variable's *r* is less than or equal to the mutation rate, another variable from the total variable set is selected and replaced by the variable in consideration. As mentioned before, we do not allow duplicates

of variables within an individual, therefore a check is made before exchange, and if the variable is already in the individual, then a new one is selected. This selection is done until a variable differing from all the other variables in the individual is reached. An example of mutation we have applied can be seen in Figure 4.13.

Before mutation:

Individual: 1 4 2 10 5 6

Assume variables 1, 10 and 6 are going to be mutated to 3, 7 and 1. Since 1 is already in the set, we choose another variable, say 8 and mutate.

After mutation:

Individual: 3 4 2 7 5 8

Figure 4.13. An example of mutation

4.3.6. Reinsertion

Once the offspring have been produced by selection, recombination and mutation of individuals from old population, the fitness of the offspring may be determined. The evaluation of offspring is done as described in section 4.3.2 using regression and correlation analysis.

The next step is to form the new population for the continuation of the search. There are different reinsertion techniques that will be stated below. All the techniques described are implemented in our study.

4.3.6.1. Pure Reinsertion. This reinsertion technique is the one described in SGA. Here, the offspring are replaced by their parents completely. That is, the new population is formed by the offspring only.

4.3.6.2. Elitist Reinsertion. In this technique, again the offspring form the new population, but a number of best parents replace the worst offspring. Elitism forces GA to retain some number of the best individuals at each generation. Such individuals can be lost by crossover or mutation. Many researchers have found that elitism significantly improves GA's performance. The number of parents to be kept depends on the implementation. Usually keeping the best parent is sufficient to achieve optimum results.

4.3.6.3. Rank-based Reinsertion. In ranked-based reinsertion, parents and offspring are ranked in descending order together and the best, that is the first half is selected as the new population. This kind of reinsertion may cause premature convergence since the diversity is minimized. To overcome this problem, the mutation rate should be kept high, but this also may not be the solution to this problem.

4.3.7. Stopping Criteria

GA continually performs the steps of selection, recombination, mutation, evaluation and reinsertion described in previous sections until a stopping criterion is met. Stopping criteria depends on the application. One would like to stop the algorithm whenever a sufficient result near optimum is achieved or continue until the algorithm finds an optimal solution. Usually the algorithm is terminated after a certain number of iterations where no more achievement in the results is obtained or the time is crucial.

Our algorithm stops whenever one of the following criteria is met:

- All the individuals are converged to the same fitness value, or
- no more improvement on the results can be obtained within a certain number of iterations, or
- a predefined number of iterations is achieved.

5. TEST RESULTS

5.1. Test Data Definition

In our tests on Class I MHC molecules, we have used the data set prepared by Zhang and co-workers [23]. They have obtained their data set from the MHCPEP [10] and Kabat [12] databases. This training data set is composed of 48 Class I MHC molecule sequence of which the crystallized structures of 9 are known, and the structure of the remaining 39 modeled. There is also another data set consisting of 20 MHC Class I molecule sequences that is used for testing the correctness of the applied technique. Both training and test data have a set of peptide sequences that bind to each of the corresponding MHC Class I molecules. For the test set, these peptide sequences are few in number, usually not more than six.

P_2, P_3, $P_{5/6}$ and P_Ω peptide sequence positions bind, respectively, to pockets B, D, C/D and F of the MHC Class I molecules as described in previous sections. We, therefore divide our data sets into four groups for each pocket. Note that Ω in P_Ω represents the last position of the peptide, and it usually refers to nine. The tests were done separately for each pocket and the corresponding peptide position in order to locate the peptide residue that can bind to each MHC molecule's pocket under study. Table 5.1 shows the MHC sequence positions that form each pocket together with the positions around these pockets that affect the binding of the peptide to that pocket. Residues at these positions are the ones that are polymorphic and within 6 Å of crystallographically observed positions of the corresponding peptide residue. The amino acids at these positions determine the peptide amino acid that will bind to that pocket.

Table 5.1. Polymorphic positions for each MHC pocket

Pocket	MHC Positions											
B	9	24	45	63	66	67	70	99	152	156	-	-
D	74	77	95	97	99	114	116	152	156	-	-	-
C/D	9	70	74	77	95	97	99	114	116	152	155	156
F	74	77	95	97	99	114	116	152	155	-	-	-

As an input to our algorithm, the hydrophobicity and size values for each of the 20 amino acids at positions stated in Table 5.1 of each MHC molecule are given. Initially three different hydrophobicity scales are tried; Scale A [35], Scale B [36], and finally Scale C [37]. All three scales are shown in Table 5.2. The size scale, which is also shown in Table 5.2 is developed by Tsai and co-workers [38]. The size is a measure of volume and the unit for volume is $Å^3$.

Table 5.2. Hydrophobicity Scales A, B and C, Size Scale BL– (C)

Amino acid	Scale A	Scale B	Scale C	Size Scale BL– (C) in $Å^3$
Phenylalanine (F)	2.8	2.45	3.7	191.9
Methionine (M)	1.9	1.68	3.4	167.0
Isoleucine (I)	4.5	2.46	3.1	163.9
Leucine (L)	3.8	2.32	2.8	164.0
Valine (V)	4.2	1.67	2.6	139.0
Cysteine (C)	2.5	2.1	2.0	103.3
Tryptophan (W)	-0.9	3.07	1.9	228.2
Alanine (A)	1.8	0.42	1.6	90.0
Threonine (T)	-0.7	0.36	1.2	121.5
Glycine (G)	-0.4	0.0	1.0	64.9
Serine (S)	-0.8	-0.05	0.6	95.4
Proline (P)	-1.6	0.67	-0.2	122.9
Tyrosine (Y)	-1.3	1.31	-0.7	197.0
Histidine (H)	-3.2	0.18	-3.0	160.0
Glutamine (Q)	-3.5	-0.79	-4.1	149.4
Asparagine (N)	-3.5	-0.3	-4.8	124.7
Glutamic acid (E)	-3.5	-0.87	-8.2	142.2
Lysine (K)	-3.9	-1.35	-8.8	167.3
Aspartic acid (D)	-3.5	-1.05	-9.2	117.3
Arginine (R)	-4.5	-1.38	-12.3	194.0

5.2. GA Performance Tests

5.2.1. Implementation Details

All the modules of our genetic algorithm are implemented in ANSI C and compiled both in Microsoft® Visual C++ 6.0 and Unix gcc environment. Therefore both Unix and Windows platforms are supported. Hardware configuration of the system used for

compilation and testing of programs is Pentium II 300 MHz CPU, 128 MB RAM, and 3 GB HDD.

5.2.2. System Parameters

In our implementation, our parameters are the generation of initial population, fitness function, population size, crossover rate, mutation rate, independent variable size, stopping criteria, selection type, crossover type, and reinsertion type.

Intially we have determined to randomly generate the population as mentioned in previous sections. Our fitness function was also clearly defined and not subject to any change during execution. After a series of runs, by keeping in mind that our individuals are rather small in length, we have decided to fix the crossover rate to 1.0, that is we cross every time. Another parameter to be kept constant is the stopping criteria, that we have mentioned in previous sections. Variable size selection is mentioned in the previous sections, where we have evaluated the test results of our algorithm.

In order to evaluate the performance of genetic algorithm and to find the best performing parameter set, we have decided to change the population size, mutation rate, selection type, crossover type and reinsertion type. In the following sections we will deal with different values of these parameters and their consequences on the results.

In all our tests we have used the data set for the second peptide position which is explained in detail in later sections. Table 5.3 shows the parameters that are kept constant during executions for this peptide positon's data set.

Table 5.3. Constant parameters for testing

Parameter	Value
N	1430
VART	20
VAR	6
cross_prob	1.0
Scale	C
Independent variable, Y	Hydrophobicity scores

5.2.3. Tests on Selection Type

As mentioned before we have implemented three different selection mechanisms; roulette-wheel selection, tournament selection and stochastic universal sampling. Their performance results are shown in Figure 5.1 and Figure 5.2. Figure 5.1 shows the average number of iterations each one takes in order to stop execution. If there is no change in the best parent within 200 iterations, the program stops. Figure 5.2 shows the percentage of the optimum results obtained. The executions are done using two-point crossover, and elitist reinsertion mechanisms.

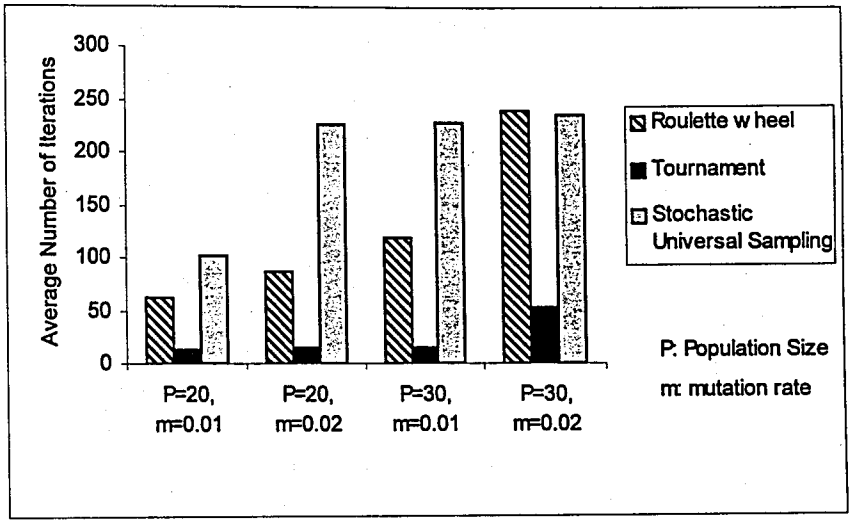

Figure 5.1. Average number of iterations for selection mechanisms

Figure 5.1 shows that as the mutation rate and population size are increased, the average number of iterations increases for every type of selection used. Here tournament selection seems to be the fastest method, but if we check Figure 5.2 we easily see that its predictive power is very low. Stochastic universal sampling is the slowest method but it has high predictive power. However it has the problem of convergence, i.e. it does not converge most of the time. Table 5.4 shows the convergence rates of the selection mechanisms discussed here.

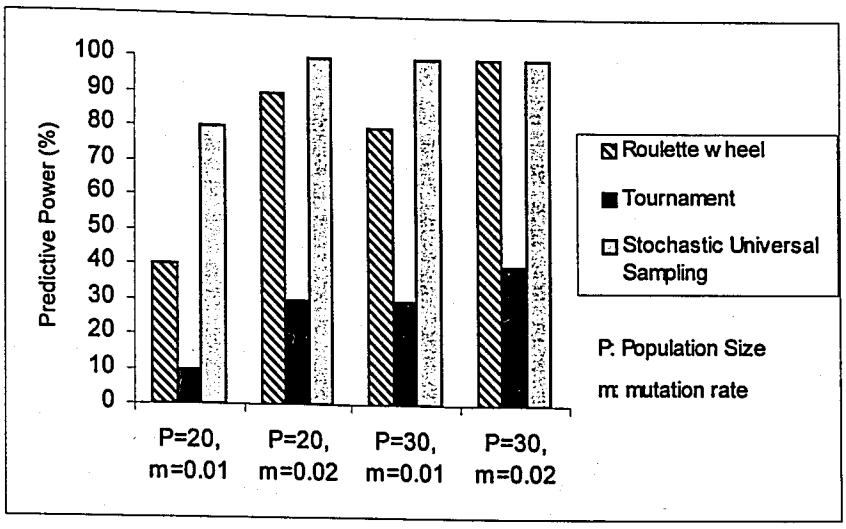

Figure 5.2. Predictive power for selection mechanisms

Table 5.4. Convergence rates for selection mechanisms

	Convergence Rate (%)			
Selection Mechanism	P:20, m:0.01	P:20, m:0.02	P:30, m:0.01	P:30, m:0.02
Roulette-wheel	100	100	90	0
Tournament	100	100	100	100
SUS	80	0	0	0

We have concluded that Roulette-wheel selection is better than the other two selection mechanisms on the average of the displayed criteria for our problem.

5.2.4. Tests on Crossover Type

For crossover, we have implemented three mechanisms; one-point, two-point and uniform crossover. We have decided to omit the results of uniform crossover and continue with one-point and two-point crossover because we did not find the results of uniform crossover promising for our problem. Figure 5.3 and Figure 5.4 show average number of iterations and predictive power of the two mechanisms respectively. The convergence rates are shown in Table 5.5. The selection mechanism used is roulette-wheel selection and the reinsertion mechanism is elitist reinsertion, which will be shown to be very efficient for our problem.

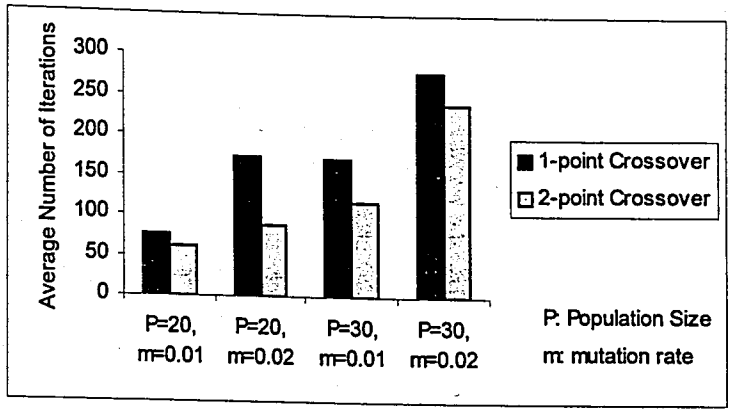

Figure 5.3. Average number of iterations for crossover mechanisms

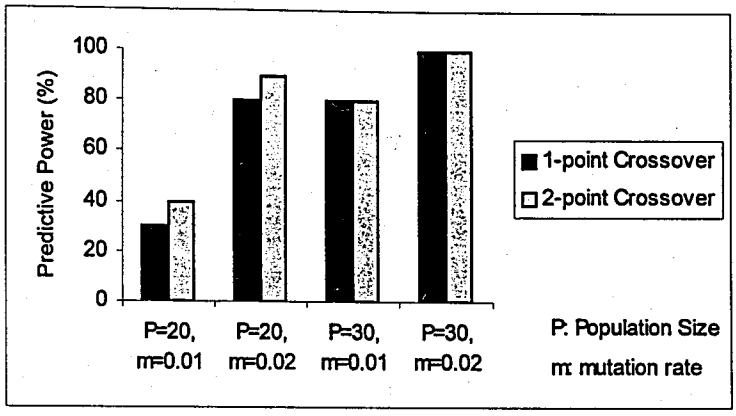

Figure 5.4. Predictive power for crossover mechanisms

Table 5.5. Convergence rates for crossover mechanisms

	Convergence Rate (%)			
Crossover Mechanism	P:20, m:0.01	P:20, m:0.02	P:30, m:0.01	P:30, m:0.02
One-point	100	90	70	0
Two-point	100	100	90	0

As it is clearly seen from the figures and convergence values, two-point crossover performs better in every instance for our problem, therefore we have selected two-point crossover as our crossover mechanism.

5.2.5. Tests on Reinsertion Type

For reinsertion, pure, elitist and rank-based reinsertion mechanisms are implemented. Figure 5.5 and Figure 5.6 show average number of iterations and predictive power for each mechanism respectively. The convergence rates are given in Table 5.6. Tests are done using roulette-wheel selection and two-point crossover.

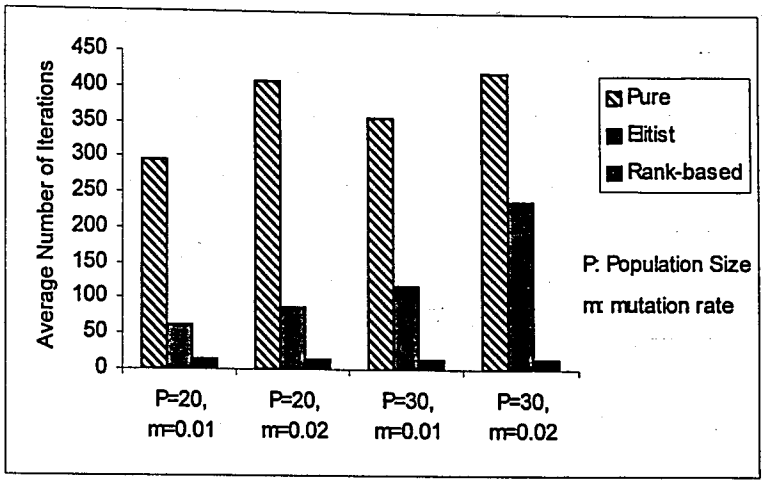

Figure 5.5. Average number of iterations for reinsertion mechanisms

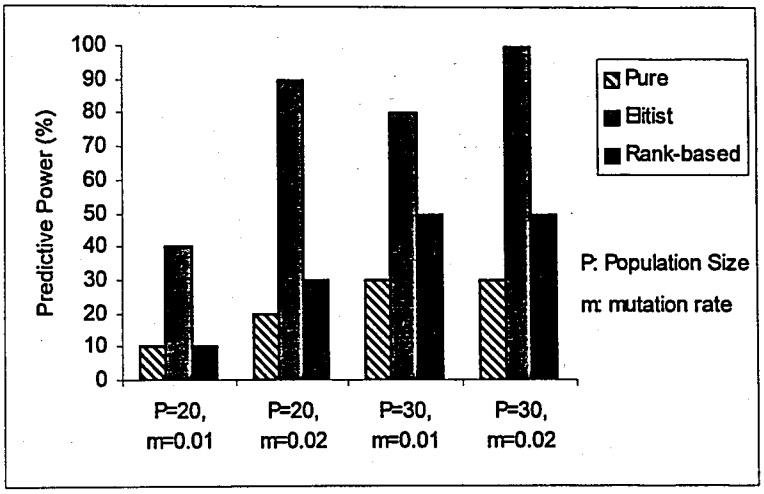

Figure 5.6. Predictive power for reinsertion mechanisms

Table 5.6. Convergence rates for reinsertion mechanisms

	Convergence Rate (%)			
Reinsertion Mechanism	P:20, m:0.01	P:20, m:0.02	P:30, m:0.01	P:30, m:0.02
Pure	10	0	0	0
Elitist	100	100	90	0
Rank-based	100	100	100	100

Pure reinsertion performs very bad in all aspects observed. Rank-based reinsertion stops at low iteration numbers but has low prediction power, and even increasing the mutation rate higher to stop premature convergence, it still converges soon to a local maxima. The best performing mechanism is the elitist reinsertion with low iteration numbers, high prediction power and high convergence rate.

5.2.6. Tests on Population Size

In order to see the effect of population size to the solution, we have chosen four population sizes; 10, 20, 30 and 40. The results are shown in Figure 5.7 and Figure 5.8. The convergence rates are given in Table 5.7. For these tests, roulette-wheel selection, two-point crossover and elitist reinsertion are used.

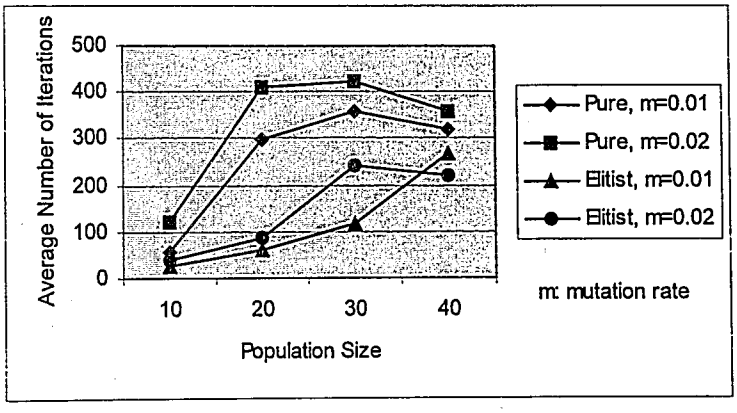

Figure 5.7. Average number of iterations for different population sizes

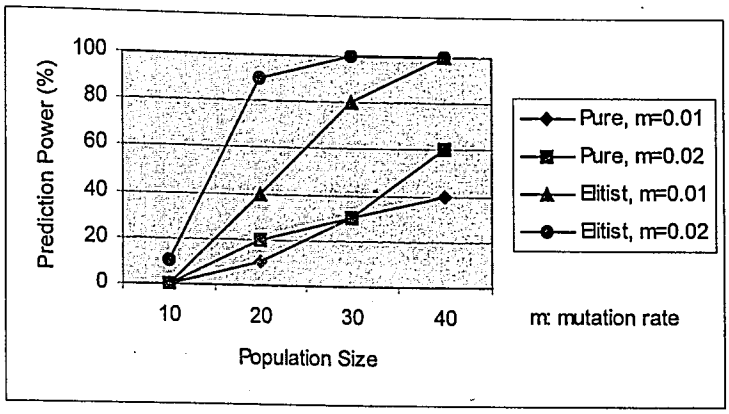

Figure 5.8. Predictive power of different population sizes

Table 5.7. Convergence rates for population size

	Convergence Rate (%)			
Population size	Pure, m:0.01	Pure, m:0.02	Elitist, m:0.01	Elitist, m:0.02
10	100	100	100	100
20	10	0	100	100
30	0	0	90	0
40	0	0	0	0

As the population size increases, the quality of results increase because the set of possible results observed at the same time increases, since this depends on the population size. However increasing the population size after a certain limit prevents the program's convergence and further increasing leads to a random search. From our results we can conclude that population size 20 or 30 are fine and 20 seems to be better on the average for all the criteria observed.

5.2.7. Tests on Mutation Rate

In order to test the effect of mutation rate on the solution, three mutation rates are tested. The results are shown in Figure 5.9, Figure 5.10, and Table 5.8. For these tests, roulette-wheel selection, two-point crossover and elitist reinsertion are used.

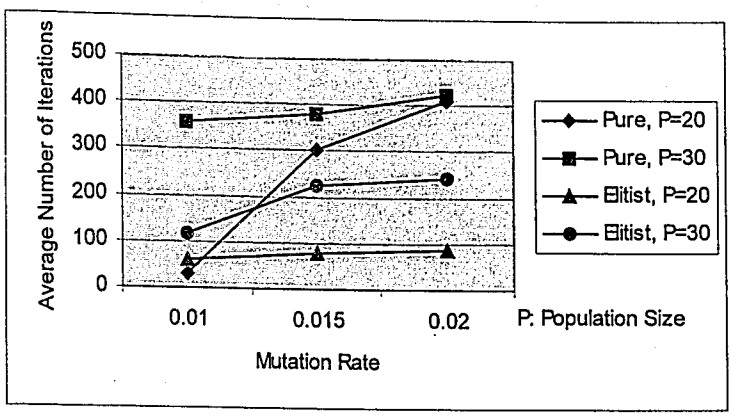

Figure 5.9. Average number of iterations for different mutation rates

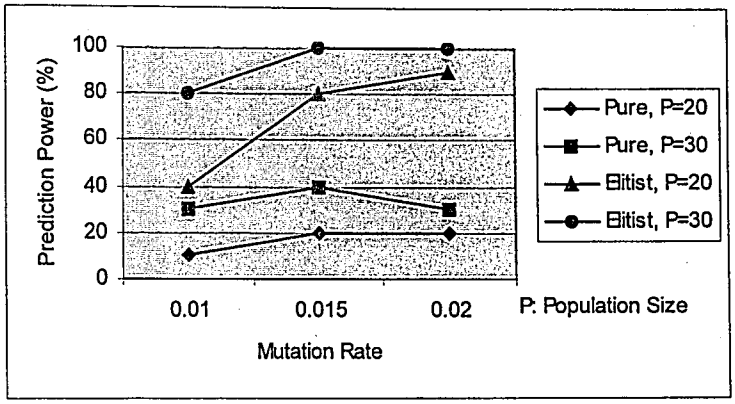

Figure 5.10. Predictive power of iterations for different mutation rates

Table 5.8. Convergence rates for mutation rate

	Convergence Rate (%)			
Mutation rate	Pure, P:20	Pure, P:30	Elitist, P:20	Elitist, P:30
0.01	10	0	100	90
0.015	0	0	100	0
0.02	0	0	100	0

As the mutation rate is increased, the average number of iterations increases and also predictive power increases. However the convergence problem occurs, i.e. as the mutation rate is increased, the program cannot converge. From the above results, 0.02 mutation rate is fine for our problem.

5.2.8. Convergence Graphs

After a series of performance tests, we have concluded that our problem best performs with Roulette-wheel selection, two-point crossover, elitist reinsertion, 0.02 as the mutation rate and 20 as the population size. Figure 5.11 shows the convergence graph for a single run using these parameters. In Figure 5.12 the average of the 20 individuals is displayed.

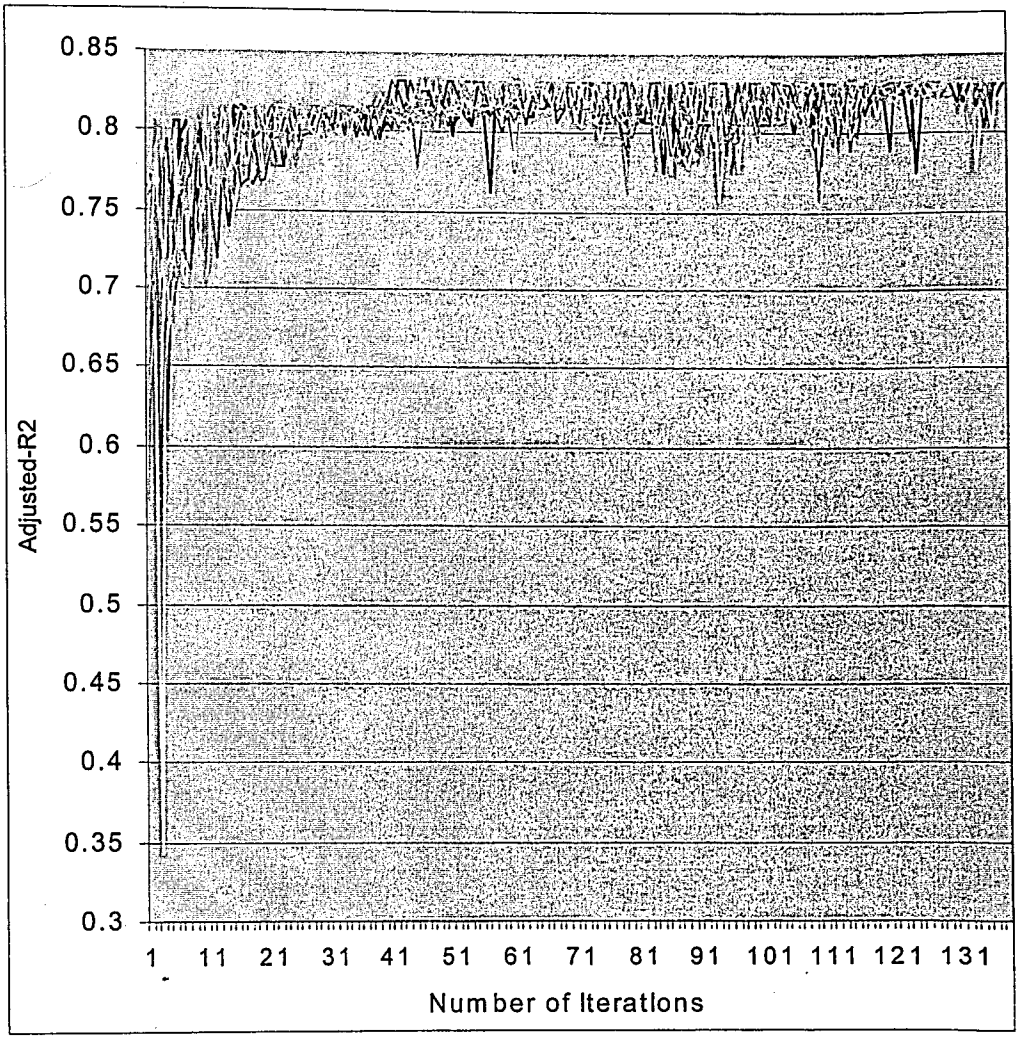

Figure 5.11. Convergence graph of P_2 for 20 individuals

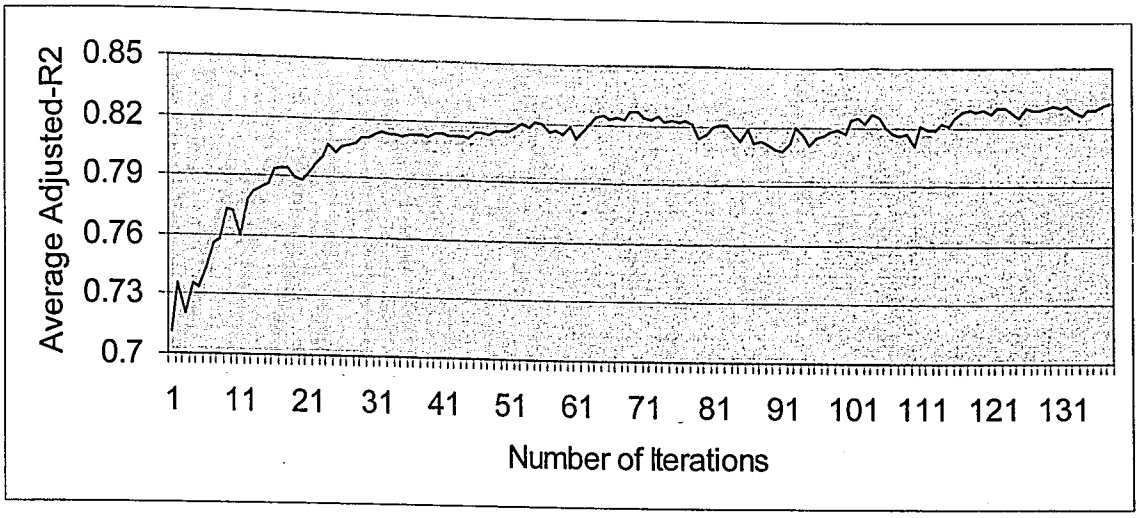

Figure 5.12. Average convergence graph of P_2

5.2.9. Numerical Comparisons to Exhaustive Search and SPSS

We have tested the correctness of our results using exhaustive search method. We were able to do exhaustive search because the problem we were dealing was not so large; but there are other receptors with higher number of variables that we are aiming to solve. By seeing the correctness of our results, we are now eligible to deal with larger problems of whose the exhaustive search will cost much both in space and time complexity. Table 5.9 shows a comparison of running times for exhaustive search and GA with optimized parameters, where running time of GA is given as the average of 10 runs.

Table 5.9. Running times for exhaustive search and GA

	VAR = 4	VAR = 5	VAR = 6
Exhaustive Search	40 minutes	1 hour, 5 minutes	2 hours, 15 minutes
GA	20 seconds	1 minute	4 minutes

Another comparison is done with the SPPS statistical software package. Different linear regression methods such as stepwise, forward and backward regression are tried. We have seen that SPSS sometimes cannot address the optimum solution that we are looking for. For P_2, the results of SPPS stepwise linear regression method are given in Table 5.10.

Table 5.10. Stepwise linear regression method with SPSS for P_2

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.834	.696	.695	2.9688
2	.868	.754	.754	2.6698
3	.879	.773	.773	2.5633
4	.893	.798	.797	2.4237
5	.902	.814	.814	2.3225
6	.910	.828	.828	2.2343
7	.915	.837	.836	2.1771
8	.918	.842	.842	2.1414
9	.920	.846	.845	2.1185
10	.923	.852	.851	2.0742
11	.926	.857	.856	2.0442
12	.927	.860	.858	2.0247
13	.928	.862	.861	2.0079
14	.928	.862	.861	2.0072
15	.930	.866	.864	1.9819
16	.932	.868	.867	1.9609
17	.933	.871	.870	1.9401
18	.934	.873	.872	1.9278
19	.935	.875	.873	1.9173

a Predictors: (Constant), V4
b Predictors: (Constant), V4, V8
c Predictors: (Constant), V4, V8, V12
d Predictors: (Constant), V4, V8, V12, V7
e Predictors: (Constant), V4, V8, V12, V7, V6
f Predictors: (Constant), V4, V8, V12, V7, V6, V15
g Predictors: (Constant), V4, V8, V12, V7, V6, V15, V14
h Predictors: (Constant), V4, V8, V12, V7, V6, V15, V14, V9
i Predictors: (Constant), V4, V8, V12, V7, V6, V15, V14, V9, V5
j Predictors: (Constant), V4, V8, V12, V7, V6, V15, V14, V9, V5, V11
k Predictors: (Constant), V4, V8, V12, V7, V6, V15, V14, V9, V5, V11, V13
l Predictors: (Constant), V4, V8, V12, V7, V6, V15, V14, V9, V5, V11, V13, V20
m Predictors: (Constant), V4, V8, V12, V7, V6, V15, V14, V9, V5, V11, V13, V20, V18
n Predictors: (Constant), V4, V8, V12, V6, V15, V14, V9, V5, V11, V13, V20, V18
o Predictors: (Constant), V4, V8, V12, V6, V15, V14, V9, V5, V11, V13, V20, V18, V2
p Predictors: (Constant), V4, V8, V12, V6, V15, V14, V9, V5, V11, V13, V20, V18, V2, V19
q Predictors: (Constant), V4, V8, V12, V6, V15, V14, V9, V5, V11, V13, V20, V18, V2, V19, V16
r Predictors: (Constant), V4, V8, V12, V6, V15, V14, V9, V5, V11, V13, V20, V18, V2, V19, V16, V7
s Predictors: (Constant), V4, V8, V12, V6, V15, V14, V9, V5, V11, V13, V20, V18, V2, V19, V16, V7, V3

For six variables, SPSS results are V4, V8, V12, V7, V6, V15, where these correspond to MHC positions 45, 63, 66, 70, 9_s (size of position 9), 67_s (size of position 67) with an adjusted R^2 value 0.828. Our GA selects 45, 63, 67, 9_s, 45_s and 67_s with an adjusted R^2 value 0.832, which will be explained in detail in the following sections.

We have concluded that optimum regression line generation with genetic algorithm techniques is much more efficient than traditional regression analysis methods and the exhaustive search. In the remaining sections, tests on different data sets for different pocket positions will be done using GA with roulette-wheel selection, two-point crossover, elitist reinsertion, point mutation with 0.02 rate, and population size 20.

5.3. Tests on Pocket B and P_2

The specificity of the P_2 side-chain is determined primarily by 10 MHC polymorphic residues. These residues were stated in Table 5.1. Table 5.11 shows the 20 MHC Class I molecules and the corresponding peptide residues at P_2 for testing. The data set used for training is given in Table 5.12. This table shows the 46 MHC Class I molecules and the corresponding peptide residues at P_2 together with their number of occurrences at that position. Residues for the 20 MHC test molecules and 46 training MHC molecules for pocket B are shown in Table 5.13 and Table 5.14 respectively.

Table 5.11. Peptide residues at P_2 of 20 Class I MHC molecules for testing

MHC	Residues at P_2	MHC	Residues at P_2
B*42	P(2)	Cw*0102	I(1)
B*5103	A(1), G(1), F(1)	B*18	D(1)
B*5501	P(4)	B*3901	H(2), R(1)
B*5502	P(5)	H-2Dk	R(2)
B*5601	P(3)	A*0203	I(1), V(1)
H-2Dq	P(1)	A*0209	L(2), I(2), T(1)
Cw*0801	A(1), S(1), T(1)	A*32	L(1)
Cw*1601	A(2)	B*5701	S(4), A(1)
Cw*0201	I(1)	Cw*0401	F(1)
Cw*0304	A(1)	Cw*0602	R(3), Q(1), A(1)

Table 5.12. Peptide residues at P_2 for 46 Class I MHC molecules for training

MHC	Residues at P_2	MHC	Residues at P_2
B*0702	P(25)	A*0204	L(10), I(6)
B*3501	P(80)	A*0207	L(3), I(3)
B*5101	P(11), A(5), G(3)	A*0205	L(8), I(3), V(3)
B*5102	P(5), A(2)	A*0206	L(7), I(6), V(2)
B*5301	P(10)	A*0214	L(3), V(2), A(2), Q(2)
H-2Ld	P(19)	A*0301	L(42), V(24), T(11), I(9)
B*40012	E(6)	A*1101	V(23), T(13), S(6)
B*4002	E(28)	A*2601	V(2), L(2), T(1), I(1)
B*4006	E(8)	A*6801	V(21), T(11), L(9), S(5), A(3)
B*4402	E(27)	A*6901	T(10), L(4), A(3), V(2), S(2)
B*44031	E(7)	A*0101	T(16), S(11), I(6), A(6), V(5)
B*3701	D(10), E(4)	B*5801	T(5), A(2)
H-2Kk	E(20), D(2)	A*2902	E(20)
B*14	R(14)	A*31012	V(15), T(7), L(6), R(6), M(3), A(3)
B*3801	H(8)	A*3301	V(9), T(7), L(6), M(4), S(3), I(2)
B*2701	R(6)	A*3302	M(2), I(2), V(1), S(1), Y(1)
B*2702	R(10)	B*1501	Q(9), L(6)
B*2703	R(22)	B*52011	Q(4), M(2), G(2)
B*2704	R(15)	A*2401	Y(36), F(12), W(10)
B*2705	R(80), K(4)	H-2Kd	Y(36), F(7)
B*2706	R(19)	Cw*0702	Y(6), R(1), K(1)
A*0201	L(244), I(46), M(43), V(27)	H-2Kb	S(6), V(5), R(4), G(4), A(3), I(3), Q(2), E(2), L(2), Y(1), P(1), K(1), D(1), C(1), W(1), N(1)
A*0202	L(8)	H-2Dd	G(29)

Table 5.13. Amino acids at pocket B for the 20 MHC Class I test molecules

	Amino acids at Positions:									
MHC	9	24	45	63	66	67	70	99	152	156
B*42	Y	S	E	N	I	Y	Q	Y	V	D
B*5103	Y	A	T	N	I	F	N	Y	E	L
B*5501	Y	A	E	N	I	Y	Q	Y	E	L
B*5502	Y	A	E	N	I	Y	Q	Y	V	L
B*5601	Y	A	E	N	I	Y	Q	Y	V	L
H-2Dq	E	S	Y	I	I	A	N	Y	A	Y
Cw*0801	Y	A	G	E	K	Y	Q	Y	T	L
Cw*1601	Y	A	G	E	K	Y	Q	Y	A	Q
Cw*0201	Y	A	G	E	K	Y	Q	Y	E	W
Cw*0304	Y	A	G	E	K	Y	Q	Y	E	L
Cw*0102	F	S	G	E	K	Y	Q	C	E	R
B*18	H	S	T	N	I	S	N	Y	V	L
B*3901	Y	S	E	N	I	C	N	Y	V	L
H-2Dk	E	S	D	E	I	A	N	S	A	D
A*0203	F	A	M	E	K	V	H	Y	E	W
A*0209	F	A	M	E	K	V	H	Y	V	L
A*32	F	A	M	E	N	V	H	Y	V	L
B*5701	Y	A	M	E	N	M	S	Y	V	L
Cw*0401	S	A	G	E	K	Y	Q	F	E	R
Cw*0602	D	S	G	E	K	Y	Q	Y	E	W

Table 5.14. Amino acids at pocket B for 46 MHC Class I training molecules

	Amino acids at Positions:									
MHC	9	24	45	63	66	67	70	99	152	156
B*0702	Y	S	E	N	I	Y	Q	Y	E	R
B*3501	Y	A	T	N	I	F	N	Y	V	L
B*5101	Y	A	T	N	I	F	N	Y	E	L
B*5102	Y	A	T	N	I	F	N	Y	E	L
B*5301	Y	A	T	N	I	F	N	Y	V	L
H-2Ld	E	S	Y	I	I	A	Q	Y	A	Y
B*40012	H	T	K	E	I	S	N	Y	V	L
B*4002	H	T	K	E	I	S	N	Y	V	L
B*4006	H	T	K	E	I	S	N	Y	V	L
B*4402	Y	T	K	E	I	S	N	Y	V	D
B*44031	Y	T	K	E	I	S	N	Y	V	L
B*3701	H	S	T	E	I	S	N	S	V	D
H-2Kk	H	S	Y	N	I	A	N	Y	D	D
B*14	Y	S	E	N	I	C	N	Y	E	L
B*3801	Y	S	E	N	I	C	N	Y	V	L
B*2701	H	T	E	E	I	C	K	Y	V	L
B*2702	H	T	E	E	I	C	K	Y	V	L
B*2703	H	T	E	E	I	C	K	Y	V	L
B*2704	H	T	E	E	I	C	K	Y	E	L
B*2705	H	T	E	E	I	C	K	Y	V	L
B*2706	H	T	E	E	I	C	K	Y	E	L
A*0201	F	A	M	E	K	V	H	Y	V	L
A*0202	F	A	M	E	K	V	H	Y	V	W
A*0204	F	A	M	E	K	V	H	Y	V	L
A*0207	F	A	M	E	K	V	H	C	V	L
A*0205	Y	A	M	E	K	V	H	Y	V	W
A*0206	Y	A	M	E	K	V	H	Y	V	L
A*0214	Y	A	M	E	K	V	H	Y	V	L
A*0301	F	A	M	E	N	V	Q	Y	E	L
A*1101	Y	A	M	E	N	V	Q	Y	A	Q
A*2601	Y	A	M	N	N	V	H	Y	E	W
A*6801	Y	A	M	N	N	V	Q	Y	V	W
A*6901	Y	A	M	N	N	V	Q	Y	V	L
A*0101	F	A	M	E	N	M	H	Y	A	R
B*5801	Y	A	T	E	N	M	S	Y	V	L
A*2902	T	A	M	Q	N	V	Q	Y	V	L
A*31012	T	A	M	E	N	V	H	Y	V	L
A*3301	T	A	M	N	N	V	H	Y	V	L
A*3302	T	A	M	N	N	V	H	Y	V	L
B*1501	Y	A	M	E	I	S	N	Y	E	W
B*52011	Y	A	T	E	I	S	N	Y	E	L
A*2401	S	A	M	E	K	V	H	F	V	Q
H-2Kd	V	A	F	Q	R	A	D	F	D	Y
Cw*0702	D	S	G	E	K	Y	Q	S	A	L
H-2Kb	V	E	Y	E	K	A	N	S	E	L
H-2Dd	V	E	Y	E	R	A	N	A	A	D

5.3.1. Test Data Preparation for Pocket B and P_2

The data set consists of 1430 input lines, represented by N. These lines are a combination of data obtained from Table 5.12 and Table 5.14. Each line's first element,

that is the independent variable Y represents the hydrophobicity or size value of the peptide residue at P_2 for the MHC of that line. The rest of the input line is the set of MHC residues stated in Table 5.14 represented by the dependent variables Xi, where i ranges from one to 20 represented by VART. First 10 values of X's are the hydrophobicity scores for the corresponding MHC sequence and the remaining 10 values are the sizes of these residues. A diagram showing the input data is in Figure 5.13. Our aim is to find the set of X's that will best represent Y in the form of an equation Y=aX+b, where b is the constant term and a's are the parameters of the equation.

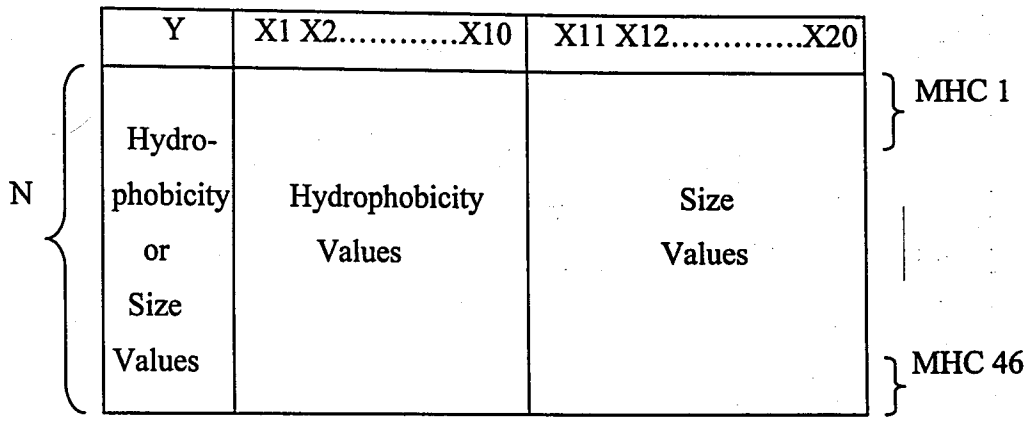

Figure 5.13. Schema for input data

5.3.2. Tests Using Hydrophobicity Scale A

By using the input data whose format is shown in Figure 5.13, GA is applied many times and a set of best equations are generated for different number of independent variables, starting with VAR=1. Our runs on different VAR sizes have shown that VAR=6; that is six independent variables out of 20 were enough to explain the independent variable Y. We have stopped increasing the variable count as soon as there is no more than one per cent increase in adjusted R^2 value by adding an new variable. We have decided that increasing the number of independent variables after six did not contribute very much to the equation's success. The maximum adjusted R^2 value that can be obtained using all 20 variables is 0.778 and 0.752 with only six variables. Predictions on different variable sizes are checked, and it is found that increasing variable size after six does not improve the

prediction accuracy. Table 5.15 shows adjusted R^2 value changes by adding new variables one by one.

Table 5.15. Adjusted R^2 values for P_2 with different VAR sizes (Scale A)

Independent Variable Size	Adjusted R^2 Value
1	0.587
2	0.692
3	0.717
4	0.730
5	0.741
6	0.752
7	0.755

The regression line and the corresponding adjusted R^2 value obtained with six variables are as follows:

$$Y^{(P2)} = -8.65 + 0.041 * X_{9\_s} + 0.431 * X_{45} + 0.039 * X_{152\_s} + 0.590 * X_{63} - 0.011 * X_{156\_s} - 0.494 * X_{66} \quad (5.1)$$

$$Adj - R^2 = 0.752 \quad (5.2)$$

In the above equation, "_s" means the size of the residue at that location. In order to determine P_2 residues for a given MHC molecule, the above equation should be applied. This equation determines Y with 75 per cent accuracy as shown by the adjusted R^2 value. During prediction, the hydrophobicity value obtained from equation (5.1) is assigned to its nearest amino acid. The results of predictions have shown that nine exact matches are obtained out of 20. Only 45 per cent of test data could be explained with Scale A, so we have moved to Scale B for better predictions.

5.3.3. Tests Using Hydrophobicity Scale B

Six independent variables out of 20 were enough to explain the independent variable Y since again increasing the number of independent variables after six did not contribute very much to the equation's success. The maximum adjusted R^2 value that can be obtained

using all 20 variables is 0.793 and 0.764 with only six variables. Table 5.16 shows adjusted R^2 value changes by adding new variables one by one.

Table 5.16. Adjusted R^2 values for P_2 with different VAR sizes (Scale B)

Independent Variable Size	Adjusted R^2 Value
1	0.554
2	0.703
3	0.730
4	0.742
5	0.754
6	0.764
7	0.769

The equation and the corresponding adjusted R^2 value obtained with six independent variables are as follows:

$$Y^{(P2)} = 1.535 - 0.470 * X_{66} + 0.012 * X_{152\_s} + 0.373 * X_9 - 0.025 * X_{24\_s} + 0.621 * X_{63} + 0.154 * X_{99} \quad (5.3)$$

$$Adj - R^2 = 0.764 \quad (5.4)$$

This equation determines Y with 76 per cent accuracy as shown by the adjusted R^2 value. The results of predictions have shown that five exact matches are obtained out of 20. During predictions, the hydrophobicity value obtained from equation (5.3) is assigned to its nearest amino acid shown in Table 5.2 according to Scale B. Only 25 per cent of test data could be explained with Scale B, so we have moved to Scale C for better predictions.

5.3.4. Tests Using Hydrophobicity Scale C

Tests using Scale C have shown that six variables are enough to explain the training data with 83 per cent accuracy. The computed adjusted R^2 value for six variables is 0.832, whereas maximum is 0.873 with 20 variables. Table 5.17 shows adjusted R^2 value changes by adding new variables one by one.

Table 5.17. Adjusted R^2 values for P_2 with different VAR sizes (Scale C)

Independent Variable Size	Adjusted R^2 Value
1	0.695
2	0.754
3	0.781
4	0.797
5	0.820
6	0.832
7	0.838

The equation and the corresponding adjusted R^2 value obtained for six independent variables are as follows:

$$Y^{(P2)} = -28.840 + 0.075 * X_{45\_s} + 0.083 * X_{67\_s} - 0.220 * X_{66} + 0.553 * X_{45} - 0.891 * X_{67} + 0.032 * X_{9\_s} \quad (5.5)$$

$$Adj - R^2 = 0.832 \quad (5.6)$$

According to the physico-chemical properties of the amino acids, the ones having similar properties can replace each other easily. Groups of amino acids where an amino acid can replace another one in the same group are shown in Table 5.18. Our predictions are done based on that table.

Table 5.18. Amino acid grouping for replacement

Grouping	Amino acids
1	L, I, M, V
2	Y, F
3	K, R
4	E, D
5	Q, N
6	S, T, A

Predictions on the 46 molecules used in the algorithm are shown in Table 5.19. Results on 20 test MHC molecules are shown in Table 5.20. During predictions, the hydrophobicity value obtained from equation (5.5) is assigned to its nearest amino acid

shown in Table 5.2 according to Scale C, or to the amino acids that it can replace. Also variation and standard deviation (σ) of the equation are calculated and predictions are done within one σ distance.

Table 5.19. Predictions for 46 Class I MHC Molecules (Scale C)

MHC	P_2	Prediction		MHC	P_2	Prediction	
B*0702	P(25)	P	√	A*0204	L(10), I(6)	L	√
B*3501	P(80)	Y	σ	A*0207	L(3), I(3)	L	√
B*5101	P(11), A(5), G(3)	Y	σ	A*0205	L(8), I(3), V(3)	I	√
B*5102	P(5), A(2)	Y	σ	A*0206	L(7), I(6), V(2)	I	√
B*5301	P(10)	Y	σ	A*0214	L(3), V(2), A(2), Q(2)	I	√
H-2Ld	P(19)	N	*√	A*0301	L(42), V(24), T(11), I(9)	C, V	√
B*40012	E(6)	E	√	A*1101	V(23), T(13), S(6)	C, V	√
B*4002	E(28)	E	√	A*2601	V(2), L(2), T(1), I(1)	C, V	√
B*4006	E(8)	E	√	A*6801	V(21), T(11), L(9), S(5), A(3)	C, V	√
B*4402	E(27)	E	√	A*6901	T(10), L(4), A(3), V(2), S(2)	C, V	√
B*44031	E(7)	E	√	A*0101	T(16), S(11), I(6), A(6), V(5)	F	σ
B*3701	D(10), E(4)	E	√	B*5801	T(5), A(2),S(5)	Y	σ
H-2Kk	E(20), D(2)	Q	X	A*2902	E(20)	P	X
B*14	R(14)	R	√	A*31012	V(15), T(7), L(6), R(6), M(3), A(3)	P	σ
B*3801	H(8),K(2)	R	√	A*3301	V(9), T(7), L(6), M(4), S(3), I(2)	P	σ
B*2701	R(6)	R	√	A*3302	M(2), I(2), V(1), S(1), Y(1)	P	σ
B*2702	R(10)	R	√	B*1501	Q(9), L(6)	Y	X
B*2703	R(22)	R	√	B*52011	Q(4), M(2), G(2)	N	√
B*2704	R(15)	R	√	A*2401	Y(36), F(12), W(10)	P	σ
B*2705	R(80), K(4)	R	√	H-2Kd	Y(36), F(7)	P, T	σ
B*2706	R(19)	R	√	Cw*0702	Y(6), R(1), K(1)	Y	√
A*0201	L(244), I(46), M(43), V(27)	L	√	H-2Kb	S(6), V(5), R(4), G(4), A(3), I(3), Q(2), E(2), L(2), Y(1), P(1), K(1), D(1), C(1), W(1), N(1)	H	√
A*0202	L(8)	L	√	H-2Dd	G(29), Y(5),P(1),F(1),R(1),A(1)	Y	√

√: Exact match
X: Mismatch
σ: Within one standard deviation (σ=2.2)
*: Special case explained in text

Out of 46 MHC training molecules, we are able to predict 32 of them exactly. During our search we have found that N or Q is the third position for the H-2Ld and crystal structures show that instead of P molecule at the second position, N or Q at the third position binds to pocket B. Also there are 43 matches within one σ distance.

Table 5.20. Predictions for 20 Class I MHC test molecules (Scale C)

MHC	Actual Bindings	Prediction	
B*42	P(2)	P	√
B*5103	A(1), G(1), F(1)	P	σ
B*5501	P(4)	P	√
B*5502	P(5)	P	√
B*5601	P(3)	P	√
H-2Dq	P(1)	N, Q	X
Cw*0801	A(1), S(1), T(1)	A	√
Cw*1601	A(2)	A	√
Cw*0201	I(1)	A	σ
Cw*0304	A(1)	A	√
Cw*0102	I(1)	A	σ
B*18	D(1)	E	√
B*3901	H(2), R(1)	D, K	√
H-2Dk	R(2)	R	√
A*0203	I(1), V(1)	L	√
A*0209	L(2), I(2), T(1)	L	√
A*32	L(1)	C, V	√
B*5701	S(4), A(1)	F	X
Cw*0401	F(1)	Y	√
Cw*0602	R(3), Q(1), A(1)	Y	X

√: Exact match
X: Mismatch
σ: Within one standard deviation (σ=2.2)

The results of predictions have shown that 14 exact matches are obtained out of 20. There are three mismatches and the other three predictions are within one standard deviation. After this low prediction accuracy obtained, we have checked the MHCPEP [10] database for new peptides that can bind to the above test data in order to increase the accuracy of our predictions. Table 5.21 shows the actual bindings of the above 20 test MHC molecules obtained from MHCPEP [10]. The MHC molecule Cw*0201 could not be found in the latest archive, so we have omitted that molecule.

Table 5.21. New predictions for 20 Class I MHC test molecules (Scale C)

MHC	Actual Bindings	Prediction	
B*42	P(1), L(1)	P	√
B*5103	P(18), A(7), G(3), F(1)	P	√
B*5501	P(2)	P	√
B*5502	P(5)	P	√
B*5601	P(3)	P	√
H-2Dq	P(2), I(1), T(1)	N, Q	*√
Cw*0801	A(1), S(1), T(1)	A	√
Cw*1601	A(2)	A	√
Cw*0201	---	A	---
Cw*0304	A(1)	A	√
Cw*0102	A(5), I(1), T(1), L(1), C(1)	A	√
B*18	E(10), P(2), D(1), R(1), Y(1), V(1)	E	√
B*3901	R(1)	D, K	√
H-2Dk	R(2)	R	√
A*0203	L(13), I(3), V(1), M(1)	L	√
A*0209	L(4), I(1), A(1), T(1)	L	√
A*32	L(2), I(2)	C, V	√
B*5701	A(3), S(2), F(1), P(1), M(1), T(1), R(1)	F	√
Cw*0401	Y(9), F(1)	Y	√
Cw*0602	Q(4), A(3), R(2)	Y	σ

√: Exact match
σ: Within one standard deviation (σ=2.2)
*: Special case explained in text

As you can see from Table 5.21 we are able to predict 18 out of 19 MHC molecules exactly. For B*5103 we were always predicting P, but Zhang and co-workers [23] depicted in their study that this molecule can bind A, G, and T only. After our search in the updated database archive, we have seen that actually there are 18 peptides that show P at their second position for B*5103. Same corrections are done on Cw*0102, B*18, A*0203, B*5701 and Cw*0401 which are shown to bind A in five peptides, E in 10 peptides, L in 13 peptides, F in one peptide, and Y in nine peptides respectively. There remained one mismatch after correction, and our predictions on the remaining three molecules show high physiological similarities to the actual bindings. H-2Dq in Table 5.21 is shown to bind to P at position two but we always predict that as an N or Q so we do encounter a mismatch for this case as for H2-Ld in Table 5.19. During our search we have found that N or Q is the third position for this MHC molecule also and crystal structures show that instead of P molecule at the second position, N or Q at the third position binds to pocket B so our

prediction is correct. Also within one σ distance we are able to predict all the molecules accurately. Our prediction accuracy for exact matches is 95 per cent, and with one standard deviation it is 100 per cent.

5.3.5. Size Prediction

After predicting the peptide at P_2 using the predicted hydrophobicity scores, we have decided that Scale C performs best, and we will continue to use it all through the remaining tests. Here we have tried to obtain the peptide at P_2 by predicting the size values of amino acids at P_2. The equation for size prediction with six variables is as follows:

$$Y^{(P2)} = 179.580 + 0.906 * X_{70\_s} + 2.771 * X_{156} - 0.297 * X_{67\_s} + 14.463853 * X_{99} - 4.445 * X_{45} - 1.196 * X_{24\_s} \quad (5.7)$$

$$Adj - R^2 = 0.652 \quad (5.8)$$

Table 5.22. Predictions for 20 Class I MHC test molecules using size information

MHC	Actual Bindings	Prediction	
B*42	P(1), L(1)	P	√
B*5103	P(18), A(7), G(3), F(1)	P	√
B*5501	P(2)	H	X
B*5502	P(5)	H	X
B*5601	P(3)	H	X
H-2Dq	P(2), I(1), T(1)	V	√
Cw*0801	A(1), S(1), T(1)	E	X
Cw*1601	A(2)	P	X
Cw*0201	---	P	---
Cw*0304	A(1)	E	X
Cw*0102	A(5), I(1), T(1), L(1), C(1)	I, L	√
B*18	E(10), P(2), D(1), R(1), Y(1), V(1)	E	√
B*3901	R(1)	R	√
H-2Dk	R(2)	E, Q	X
A*0203	L(13), I(3), V(1), M(1)	E, Q, V	√
A*0209	L(4), I(1), A(1), T(1)	I, L	√
A*32	L(2), I(2)	I, L	√
B*5701	A(3), S(2), F(1), P(1), M(1), T(1), R(1)	A	√
Cw*0401	Y(9), F(1)	Y	√
Cw*0602	Q(4), A(3), R(2)	P	X

√: Exact match
X: Mismatch

As the Table 5.22 shows, by using predicted size information from equation (5.7), we are able to find 11 exact matches out of 19. Accuracy is 58 per cent.

5.4. Tests on Pocket F and P_Ω

The specificity of the P_Ω side-chain is determined primarily by nine MHC polymorphic residues. These residues are stated in Table 5.1. The data set for training is in Table 5.23. Table 5.24 shows the 20 MHC Class I molecules and the corresponding peptide residues at P_Ω for testing. This table shows the 48 MHC Class I molecules and the corresponding peptide residues at P_Ω and their number occurrences at that position. Residues for the 20 MHC test molecules and 48 MHC molecules for pocket F are shown in Table 5.25 and Table 5.26 respectively.

Table 5.23. Peptide residues at P_Ω of 48 Class I MHC molecules for training

MHC	P_Ω	MHC	P_Ω
B*0702	L(15),V(5),I(5)	A*0207	I(3),V(3)
B*3501	L(37),Y(26),M(7),I(6),F(4)	A*0205	V(8),L(3),I(3)
B*5101	V(7),I(7),L(3)	A*0206	V(7),L(7),I(3)
B*5102	V(4),I(3)	A*0214	L(4),V(3)
B*5301	L(3),F(2),W(2),I(2)	A*0301	K(50),R(24),Y(22)
H-2Ld	L(10),F(8)	A*1101	K(39),R(21),Y(6)
B*40012	L(6)	A*2601	V(4),L(1),A(1)
B*4002	L(13)	A*6801	R(30),K(13)
B*4006	V(5)	A*6901	R(10),V(7),L(4),K(2)
B*4402	Y(14),F(8)	A*0101	Y(54)
B*44031	F(5),Y(2)	B*5801	F(5),W(2)
B*3701	I(9),L(6),T(3)	A*2902	Y(9),L(6),F(3)
H-2Kk	I(19),L(2),V(1)	A*31012	R(32),K(13)
B*14	L(13),G(3),V(2)	A*3301	R(24),K(10)
B*3801	L(4),Y(3),F(3)	A*3302	R(7)
B*2701	F(2),L(2)	B*1501	Y(9),F(4)
B*2702	Y(5),F(3)	B*52011	I(4),L(2),V(2)
B*2703	R(11),K(5)	A*2401	L(34),F(15),I(10)
B*2704	R(6),K(3),L(2),A(2)	H-2Kd	L(31),I(9),V(3)
B*2705	K(27),R(26),L(12)	Cw*0702	Y(6),L(2)
B*2706	L(5),R(3),K(2)	B*0801	L(29),I(5),F(4),V(3)
A*0201	V(147),L(130),I(47),A(29),M(17)	H-2Db	I(22),L(18),M(12),V(3)
A*0202	V(6),L(2),I(2)	H-2Kb	L(25),M(4),V(4),I(2)
A*0204	V(8),L(6),I(3)	H-2Dd	L(17),I(6),M(3),Y(3),F(3)

Table 5.24. Peptide residues at P_Ω of 20 Class I MHC molecules for testing

MHC	P_Ω	MHC	P_Ω
B*42	L(1),Y(1)	Cw*0102	L(1)
B*5103	V(1),I(1),F(1)	B*18	A(1)
B*5501	A(3),T(1)	B*3901	I(1),V(1),M(1)
B*5502	A(3),V(2)	H-2Dk	L(1),G(1)
B*5601	A(3)	A*0203	L(3)
H-2Dq	M(1)	A*0209	L(2),V(2),A(1)
Cw*0801	I(2),V(1)	A*32	R(1)
Cw*1601	L(2)	B*5701	W(3),F(2)
Cw*0201	W(1)	Cw*0401	F(1)
Cw*0304	L(1)	Cw*0602	Y(3),L(1),V(1)

Table 5.25. Amino acids at pocket F for 20 MHC Class I test molecules

	Amino acid Positions								
MHC	74	77	95	97	99	144	116	152	155
B*42	D	S	L	S	Y	N	Y	V	Q
B*5103	Y	N	W	T	Y	N	Y	E	Q
B*5501	D	S	W	T	Y	N	L	E	Q
B*5502	D	S	W	T	Y	N	L	V	Q
B*5601	D	S	W	T	Y	N	L	V	Q
H-2Dq	F	S	I	R	Y	E	F	A	H
Cw*0801	D	S	L	R	Y	N	F	T	Q
Cw*1601	D	S	L	W	Y	D	S	A	Q
Cw*0201	D	N	L	R	Y	D	S	E	E
Cw*0304	D	S	I	R	Y	D	Y	E	Q
Cw*0102	D	S	L	W	C	D	Y	E	Q
B*18	Y	S	L	R	Y	D	S	V	Q
B*3901	D	S	L	R	Y	N	F	V	Q
H-2Dk	F	D	I	R	S	E	F	A	R
A*0203	H	D	V	R	Y	H	Y	E	Q
A*0209	H	D	V	R	Y	H	Y	V	Q
A*32	D	S	I	M	Y	Q	D	V	Q
B*5701	Y	N	I	V	Y	D	S	V	Q
Cw*0401	D	N	L	R	F	N	F	E	Q
Cw*0602	D	N	L	W	Y	D	S	E	Q

Table 5.26. Amino acids at pocket F for 48 MHC Class I training molecules

	Amino acid Positions								
MHC	74	77	95	97	99	114	116	152	155
B*0702	D	I	L	I	Y	D	Y	E	Q
B*3501	Y	S	I	R	Y	D	S	V	Q
B*5101	Y	N	W	T	Y	N	Y	E	Q
B*5102	Y	N	W	T	Y	N	Y	E	Q
B*5301	Y	N	I	R	Y	D	S	V	Q
H-2Ld	F	N	L	W	Y	E	F	A	Y
B*40012	Y	S	L	R	Y	N	Y	V	Q
B*4002	Y	S	L	S	Y	N	Y	V	Q
B*4006	Y	S	W	T	Y	N	Y	V	Q
B*4402	Y	N	I	R	Y	D	D	V	Q
B*44031	Y	N	I	R	Y	D	D	V	Q
B*3701	Y	D	I	R	S	N	F	V	Q
H-2Kk	F	N	F	R	Y	E	Y	D	R
B*14	D	S	L	W	Y	N	F	E	Q
B*3801	Y	N	L	R	Y	N	F	V	Q
B*2701	Y	N	L	N	Y	H	D	V	Q
B*2702	D	N	L	N	Y	H	D	V	Q
B*2703	D	D	L	N	Y	H	D	V	Q
B*2704	D	S	L	N	Y	H	D	E	Q
B*2705	D	D	L	N	Y	H	D	V	Q
B*2706	D	S	L	N	Y	D	Y	E	Q
A*0201	H	D	V	R	Y	H	Y	V	Q
A*0202	H	D	L	R	Y	H	Y	V	Q
A*0204	H	D	V	M	Y	H	Y	V	Q
A*0207	H	D	V	R	C	H	Y	V	Q
A*0205	H	D	L	R	Y	H	Y	V	Q
A*0206	H	D	V	R	Y	H	Y	V	Q
A*0214	H	D	L	R	Y	H	Y	V	Q
A*0301	D	D	I	I	Y	R	D	E	Q
A*1101	D	D	I	I	Y	R	D	A	Q
A*2601	D	N	I	R	Y	Q	D	E	Q
A*6801	D	D	I	M	Y	R	D	V	Q
A*6901	D	D	V	R	Y	H	Y	V	Q
A*0101	D	N	I	I	Y	R	D	A	Q
B*5801	Y	N	I	R	Y	D	S	V	Q
A*2902	D	N	I	M	Y	R	D	V	Q
A*31012	D	D	I	M	Y	Q	D	V	Q
A*3301	D	D	I	M	Y	Q	D	V	E
A*3302	D	D	I	M	Y	Q	D	V	E
B*1501	Y	S	L	R	Y	D	S	E	Q
B*52011	Y	N	W	T	Y	N	Y	E	Q
A*2401	D	N	L	M	F	H	Y	V	Q
H-2Kd	F	S	F	R	F	Q	F	D	Y
Cw*0702	D	S	L	R	S	D	S	A	Q
B*0801	D	S	L	S	Y	N	Y	V	Q
H-2Db	F	S	L	Q	S	L	F	A	H
H-2Kb	F	D	I	V	S	Q	Y	E	R
H-2Dd	F	D	L	W	A	W	F	A	R

Tests on P_Ω resulted in very low adjusted R^2 value. Maximum R^2 that can be obtained using all 18 variables (same as P_2: nine for hydrophobicity scores and nine for the size scores) is 0.701. The predictions are found to be very poor, therefore in order to increase prediction power, we have divided the train data into two logical groups using physic-chemical properties of amino acids.

First group contains 33 training and 15 test molecules, second group contains 12 training and four test molecules.

The equation of the first group with six independent variables is as follows:

$$Y^{(P\Omega)}\_eq1 = -26.788 - 0.005 * X_{97\_s} + 0.152 * X_{116\_s} - 0.004 * X_{152\_s} - 0.057 * X_{114} + 0.224 * X_{116} + *X_{114\_s} \quad (5.9)$$

$$Adj - R^2 = 0.976 \quad (5.10)$$

Predictions for training molecules are in Table 5.27 for test molecules.

Table 5.27. Predictions at P_Ω of 15 Class I MHC test molecules

MHC	P_Ω		MHC	P_Ω	
B*42	L, Y	√	B*3901	I.V.M	√
B*5103	V,I,F	√	H-2Dk	L,G	√
B*5501	A,T	X	A*0203	L	√
B*5502	A,V	X	A*0209	L,V,A	√
B*5601	A	X	A*32	R	√
H-2Dq	M	√	Cw*0304	L	√
Cw*0801	I,V	√	Cw*0401	F	√
Cw*0102	L	√			

√: Exact match
X: Not included in the training MHC molecule set

Predictions have shown that we are able to predict the test data with 80 per cent accuracy with first equation.

The equation of the second group with four independent variables is as follows:

$$Y^{(P\Omega)}\_eq2 = -20.520 + 0.145 * X_{152\_s} - 0.121 * X_{74} - 0.026 * X_{77} + 0.012 * X_{74\_s} \quad (5.11)$$

$$Adj - R^2 = 0.564 \quad (5.12)$$

Predictions for training molecules are in Table 5.28 for test molecules.

Table 5.28. Predictions at P_Ω of four Class I MHC test molecules

MHC	P_Ω		MHC	P_Ω	
B*5701	W,F	√	Cw*0602	Y,L,V	√
Cw*1601	L	X	B*18	A	√
√: Exact match X: Not included in the training MHC molecule set					

Predictions have shown that we are able to predict the test data with 75 per cent accuracy with second equation.

5.5. Comparisons with Another Technique

Our results showed that we can predict second and ninth positions of a peptide sequence by using one or two regression lines. Zhang and co-workers [23] managed to explain the data only by dividing it into very small groups. For second position, they have divided the data into twelve and in addition to this, they proposed many conditions in order for the division to hold. We have managed to explain their data with a single regression line with 95 to 100 per cent accuracy. They have also divided the data for ninth position into 4 groups and they could not be able to explain one of the groups and called it as being nonspecific. We are able to explain ninth position with two regression lines, and one of our groups was the one that is found to be nonspecific by them. We have explained the first group with 80 per cent and second group with 75 per cent accuracy.

6. CONCLUSIONS

With today's technology and present methods, it is very difficult to identify peptide motifs. As it is described by Zhang and co-workers [23], researchers usually divide the data set into several small groups in order to predict the peptide motifs of a given test data. They have managed to explain all the data for the second peptide position by dividing the data set into 12 groups. They have applied the same method to ninth position by dividing it into four parts.

In our technique of regression analysis using GAs, we have managed to explain all the data for second peptide position with only one regression line with six MHC sequence positions. We have predicted with 95 per cent accuracy all the test data for second position. This value becomes 100 per cent if we look for one standard deviation away from our predictions. For position nine, we have divided the set into two because one part of the set has a high variation. The first group predictions resulted in 80 per cent accuracy. The second group predictions resulted in 75 per cent accuracy. These predictions imply that there is a linear relationship between the MHC molecules' sequences and the peptide sequences. Our technique makes possible the ligand design for a given receptor and can be applied to other problems rather than MHC receptors.

In addition to peptide motif discovery, we are also able to determine the positions of MHC molecules that are the determining positions for the peptide to bind to them. Applying this technique to other receptor-ligand problems makes it possible for us to determine the positions of the receptors of which are important in binding to the ligand.

GAs are very efficient in determining the optimum regression line. There are many application areas in molecular biology that GA is applied in regression analysis such as drug design. Since our problem size is small, it can be solved with exhaustive techniques but the problem size can be extended for other receptor-ligand problems and therefore exhaustive search may not be possible or it may be very time and space consuming. We are planning to attack such problems in near future, and we believe that GA will perform very well on larger problems.

After doing a series of performance optimization tests, we have concluded that our problem is best solved with roulette-wheel selection, two-point crossover, elitist reinsertion, and with population size 20, mutation rate 0.02. This set finds the optimum result 90 per cent of the time with 90 iterations on the average and it converges all the time.

This project enabled us to find the MHC binding motifs of peptides out of all the possible peptide sequences. Knowing this method will allow us to select the candidate peptides that are the ones that have the motif for that specific MHC molecule, so that they can be used to develop a vaccine against a virus.

As a future work, we aim to organize this technique into a software package that will provide great help to the people doing research on vaccine design.

REFERENCES

1. Russel, S. and P. Norvig, *Artificial Intelligence – A modern Approach*, Prentice Hall, New Jersey, 1995.

2. Forrest, S., "Genetic Algorithms: Principles of Natural Selection Applied to Computation", *Science*, Vol. 261, pp.872-878, August 1993.

3. Parsons, R. J., S. Forrest and C. Burks, "Genetic Algorithms, Operators, and DNA Fragment Assembly", *Machine Learning*, Vol. 21, pp. 11-33, 1995.

4. Hunkapiller, T., R. Kaiser, B. Koop and L. Hood, "Large-Scale Automated DNA Sequence Determination", *Science*, Vol. 254, pp. 59-67, 1991.

5. Shulze-Kremer, S., *Genetic Algorithms and Protein Folding*, 1995, http://igd.rz-berlin.mpg.de/~steffen/bcc/111.html

6. Mitchell, M., *An Introduction to Genetic Algorithms*, The MIT Press, England, 1996.

7. Bailey, T. L. and C. Elkan, "Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization", *Machine Learning*, Vol. 21, pp. 51-80, 1995.

8. Gaur, A. and C. G. Fathman, "Immunotherapeutic strategies directed at the trimolecular complex", *Advan. Immunol.*, Vol. 56, pp. 219-265, 1994

9. Zhang, C., J. L. Cornette, J. A. Berzofsky and C. Delisi, "The organization of human leukocyte antigen class I epitopes in HIV genome products: implications for HIV evolution and vaccine design", *Vaccine*, Vol. 15, pp. 1291-1302, 1997

10. Rhodes, D. and J. Trowsdale, *Genetics and Molecular Genetics of the MHC*, 2000, http://www.path.cam.ac.uk/~immuno/mhc/mhc.html

11. Robinson, J., A. Malik, P. Parham, J. G. Bodmer, S. G. E. Marsh, " IMGT/HLA database – a sequence database for the human major histocompatibility complex", *Tissue Antigens*, Vol. 55, pp. 280-287, 2000.

12. Brusic, V., G. Rudy, A. Kyne and L.C. Harrison, "MHCPEP, a database of MHC-binding peptides: update 1997", *Nucleic Acids Research*, Vol. 26, pp. 368-371, 1998.

13. Korber, B., C. Brander, J. Moore, P. D'Souza, B. Walker, R. Koup, B. Haynes and G. Myers, *HIV Molecular Immunology Database*, 1996, http://hiv-web.lanl.gov/immuno/

14. Kabat, E. A., T. T. Wu, H. M. Perry, K. S. Gottesman and C. Foeller, *Sequence of Proteins of Immunological Interest*, 2000, http://immuno.bme.nwu.edu/sequence_name.html

15. Korber, B., J. P. Moore, C. Brander, B. D. Walker, B. F. Haynes and R. Coup, *A database of MHC Ligands and Peptide Motifs*, 1999, http://www.uni-tuebingen.de/uni/kxi/

16. Brusic, V., *FIMM, a database of functional molecular immunology*, 2000, http://sdmc.hrdl.org.sg/fimm

17. Branden, C. and J. Tooze, *Introduction to Protein Structure*, Garland Publishing, Inc., New York and London, 1991.

18. Parham, P., "Getting into the groove", *Nature*, Vol. 342, pp. 617-618, 1989.

19. Alberts, B., D. Bray, J. Lewis, M. Raff, K. Roberts and J. D. Watson, *Molecular Biology of the Cell*, Garland Publishing, Inc., New York and London, 1994.

20. *The Immune Responses (Adaptive Immunity)*, 1999, http://www/cat/md/us/courses/bio141/lecguide/unit3/u3iab.html

21. Bjorkman, P. J., M. A. Saper, B. Samraoui, W. S. Bennett, J. L. Strominger and D. C. Wiley, "The foreign antigen binding site and T cell recognition regions of class I histocompatibility antigens", *Nature*, Vol. 329, pp. 5120-518, 1987.

22. Matsumura, M., D. H. Fremont, P. A. Peterson and I. A. Wilson, "Emerging principles for the recognition of peptide antigens by MHC class I molecules", *Science*, Vol. 257, pp. 927-934, 1992.

23. Zhang, C., A. Anderson and C. DeLisi, "Structural Principles that Govern the Peptide-binding Motifs of Class I MHC Molecules", *J. Mol. Biol.*, Vol. 281, pp. 929-947, 1998.

24. Baringa, M., "Getting some backbone: How MHC binds peptides", *Science*, Vol. 257, pp. 880-881, 1992.

25. Parham, P., "Deconstructing the MHC", *Nature*, Vol. 360, pp. 300-301, 1992.

26. Rechenberg, I., *Evolutionsstrategie: Optimierung techniser Systeme nach Prinsipien der biologische Evolution [Evolutionary Strategy: Optimization of Technical Systems According to the Principles of Biological Evolution]*, Frommann Holzboog Verlagm Stuttgart, Germany, 1973.

27. Schwefer, H. P., "Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie" [Numerical Optimization of Computer Models by Means of the Evolutionary Strategy], *Interdisciplinary Systems Research*, Vol. 26, Birkäuser, Basel, Switzerland, 1977.

28. Fogel, L.J., A. J. Owens and M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, 1966.

29. Holland, J. H., *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arborm Mich., 1975.

30. Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.

31. Setubal J. and J. Meidanis, *Introduction to Computational Molecular Biology*, PWS Publishing Company, Boston, 1997.

32. Patton, A. L., W. F. Punch III and E. D. Goodman, "A Standard GA Approach to Native Protein Conformation Prediction", *Proceedings of the Sixth International Conference on Genetic Algorithms*, University of Pittsburgh, 15-19 July 1995, pp. 574-581, Morgan Kaufmann Publishers Inc., San Francisco, 1995.

33. Rogers, D., "Development of the Genetic Function Approximation Algorithm", *Proceedings of the Sixth International Conference on Genetic Algorithms*, University of Pittsburgh, 15-19 July 1995, pp. 589-596, Morgan Kaufmann Publishers Inc., San Francisco, 1995.

34. Harnett, D. L. and K. S. Ashok, *Statistical Methods for Business and Economics*, Addison-Wesley, Reading, Massachusetts, 1991.

35. Kyte, J. and R. Doolittle, "A simple method for displaying hydropathic character of a protein", *J. Mol. Biol.*, Vol. 157, pp. 105-132, 1982.

36. Edelman, J. "Quadratic Minimization of Predictors for Protein Secondary Structure. Application to Transmembrane alpha-Helices", *J. Mol. Biol.*, Vol. 232, pp. 165-191, 1993.

37. Engelman, D. M., T. A. Steitz and A. Goldman, "Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins", *Annu. Rev. Biophys. Biophys. Chem.*, Vol. 15, pp. 321-353, 1986.

38. Tsai, J., R. Taylor, C. Chothia and M. Gerstein, "The Packing Density in Proteins: Standard Radii and Volumes", *J. Mol. Biol.*, Vol. 290, pp. 253-5266, 1999.

REFERENCES NOT CITED

Black, J. and J. F. Bradley, *Essential Mathematics for Economists*, John Wiley & Sons Ltd., England, 1994.

Chatterjee, B. and B. Price, *Regression Analysis by Example*, John Wiley & Sons, Inc., New York, 1991.

Filho, J. L. R. and P.C. Treleaven, "Genetic-Algorithm Programming Environments", *IEEE Computer*, Vol. 27, pp. 28-43, June 1994.

Johnson, R. A., *Miller & Freund's Probability & Statistics For Engineers*, Prentice Hall, New Jersey, 1995.

Miller, A. J., *Subset Selection in Regression*, Chapman and Hall, London, 1990.

Sen, A. and M. Srivastava, *Regression Analysis Theory, Methods, and Applications*, Springer-Verlag, New York, 1990.

Shields, P. C., *Elementary Linear Algebra*, Worth Publishers Inc., New York, 1969.

Stuart, M., *Major Histocompatibility Complex*, 1997
http://www.kcom.edu/faculty/chamberlain/Website/MSTUART/lect6.htm