Hybrid Genetic Algorithms for Flexible Ligand Docking

A dissertation submitted in partial fulfilment of the requirements for the M.Tech (Computer Science) degree of the Indian Statistical Institute, Kolkata.

By

Anindya Sen

under the supervision of

Dr. Ashish Ghosh

Machine Intelligence Unit

INDIAN STATISTICAL INSTITUTE
203, B.T. Road, Kolkata - 700035
21 July, 2004

Acknowledgement

I take pleasure in thanking Dr. Ashish Ghosh for providing guidance and encouragement throughout the dissertation period. His sound words of advice have helped me remain focussed in the face of many challenges.

Finally I take this opportunity to thank my classmates, family and friends without whose support and encouragement this dissertation would not have been possible.

Anindya Sen

Indian Statistical Institute,
203, Barrackpore Trunk Road,
Kolkata - 700 035

Certificate Of Approval

This is to certify that this thesis titled "Hybrid Genetic Algorithms for Flexible Ligand Docking" submitted by Anindya Sen towards partial fulfillment of requirements for the degree of M.Tech in Computer Science at Indian Statistical Institute, Kolkata embodies the work done under my supervision.

Dr. Ashish Ghosh
Machine Intelligence Unit,
Indian Statistical Institute,
Kolkata - 700108.

(External Expert)

Abstract

The present dissertation attempts to employ hybrid genetic algorithms (GAs) to solve the *flexible-ligand docking* problem i.e. predicting the binding conformation of a flexible ligand molecule into a rigid protein. Our hybrid GA scheme uses the concept of Lamarckian genetics to perform a local search about an individual, followed by replacing it with a better solution found in its neighborhood. Two local search schemes have been investigated and their performance relative to the standard GA have been compared. Preliminary results obtained on a set of three protein-ligand complexes have shown promising results.

Contents

Chapter 1

Introduction

1.1 Introduction to bioinformatics

Bioinformatics is an interdisciplinary science that seeks to uncover knowledge from a vast quantity of biological data using computational and informational approaches. Currently, the use of computers has revolutionized the field of biology and has helped in developing a better understanding of various processes taking place at the level of **genes** and **proteins**. A gene is a fundamental unit of inheritance and it codes for a protein. A protein, in turn, is responsible for various metabolic functions taking place in our body. Some of the challenging problems being faced today include gene **sequence analysis**, **protein structure prediction**, **drug docking** and **analysis of genetic and metabolic networks** [1]. In this dissertation, our focus is on attempting to solve the problem of docking of a drug molecule into a protein.

1.2 The molecular basis of life

The molecular basis of life is formed by complex biochemical processes that constantly produce and recycle molecules and do so in a highly coordinated fashion. The reactions comprising this *"molecular circuitry"* perform several important functions [1]:

- Construction of molecular components essential for life.
- Breakdown of molecules harmful to the cell.
- Storage and conversion of energy.
- Exchange of information in the cell or between cells.

The *"molecular circuitry"* has intended modes of operation that correspond to healthy states of an organism and aberrant modes that correspond to diseased states.

Molecular medicine aims to diagnose a disease by identifying its molecular basis, i.e., to discover what is wrong in its circuitry.

1.2.1 Metabolic reactions

Metabolic reactions take place spontaneously under physiological conditions (in aqueous solution, under room temperature and neutral pH). However, for a reaction to occur, it requires the presence of a specific molecule, called an enzyme to catalyze it. An enzyme speeds up the rate of the reaction tremendously, by rates of as much as 10^7. Typically, an enzyme is a protein molecule. Human beings are known to possess as many as 6000 enzymes.

To understand how an enzyme catalyzes a reaction, let us consider a protein called *dihydrofolate reductase* that turns dihydrofolate into tetrahydrofolate (see Figure 1.1). The surface of the protein shows a large and deep pocket (shown in red), called a **binding pocket/site** which is ideally formed in terms of geometry and chemistry, such as to bind to the **substrate** molecule, dihydrofolate, and present it in a conformation that is conducive for the desired chemical reaction to take place. The site where the reaction is catalyzed is called the **active site**.

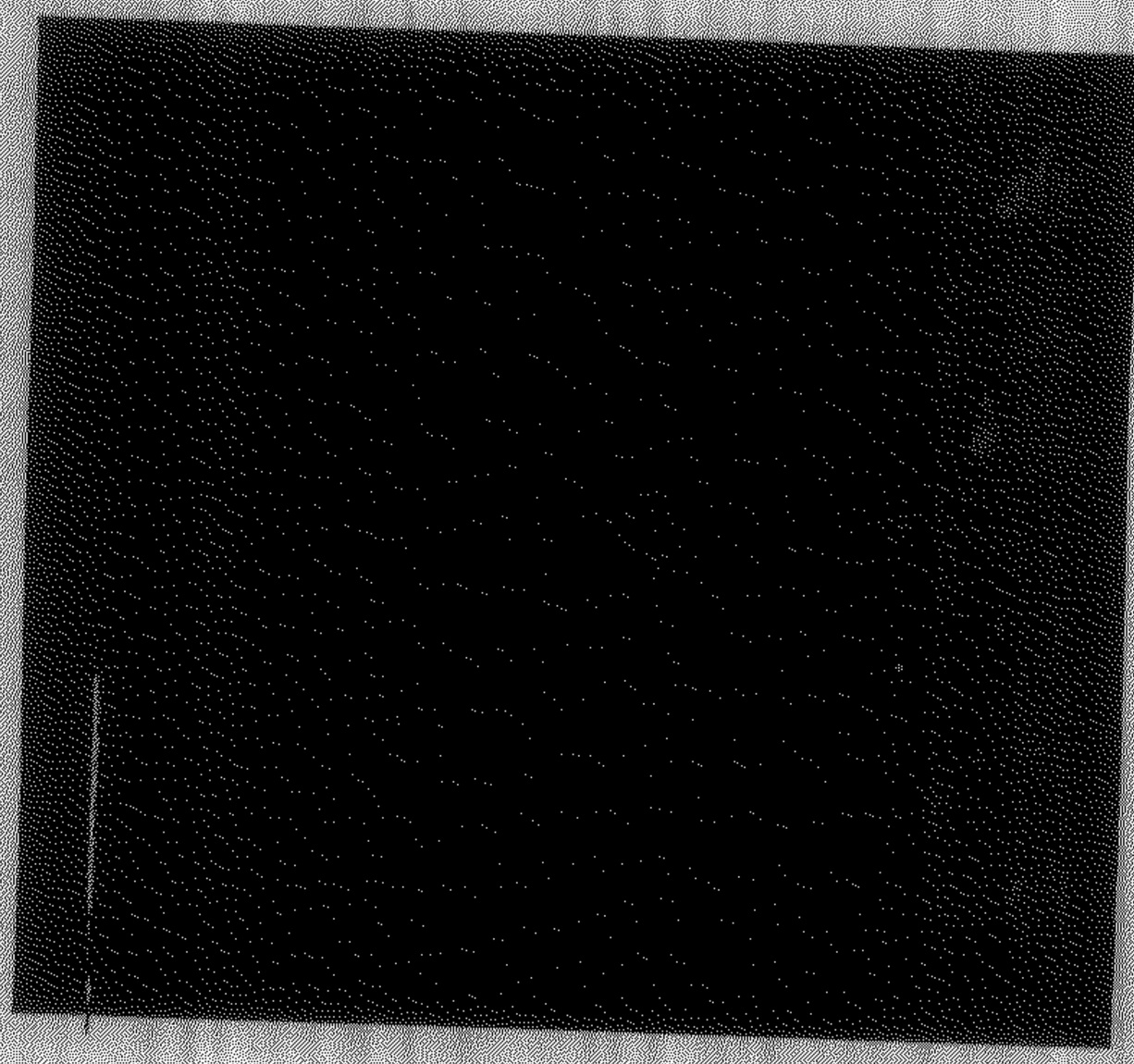

Figure 1.1: 3-D structure of DHFR.

1.3 Role of drugs in curing diseases

Some metabolic reactions do not happen at their intended rate, resources that are needed are not present in sufficient amounts or waste products accumulate in the body, leading to diseases such as sickle cell anaemia, diabetes, Alzheimer's disease etc. In order to replace the aberrant processes with those that restore a healthy state, the effectiveness of the proteins responsible for the disease has to be controlled. A **drug** molecule (also called a **ligand molecule**) is a chemical compound, of typically small size, which modifies the enzymatic action of a protein by strongly binding to it.

An **inhibitor** is a drug molecule which binds to a protein more strongly than the substrate. Then the binding site of most enzyme molecules will contain drug molecules and cannot catalyze the desired reaction in the substrate. In some cases, the drug molecule binds to the protein for the remaining lifetime of the molecule. Such a drug is called a *suicide inhibitor*. The effect of the drug can be controlled by the time and dose it is administered.

1.4 Designing drugs

Current pharmaceutical research involves the following :

- Identification of **target** protein to which a drug binds.
- Identification of **potential** drug candidate.

Once the target protein has been identified, the search for drugs is governed by several procedures. The dominating paradigm has been the concept of molecular similarity based on the "**lock-and-key**" principle. According to this principle, a drug molecule fits into the binding pocket of the protein like a key inside a lock. Thus, molecules whose surfaces are similar in shape and chemical features should bind to the target protein with comparable strength. Drug design was based on comparing drug molecules, either intuitively or systematically with the help of computers, without using the 3D structure of the protein (about which little was known anyway).

However, as 3D structure of the target proteins became known, the focus of drug research shifted to a **rational** or **structure-based** approach, which involved prediction of the structure and binding affinity of the molecular complex involving a structurally resolved protein and a drug (ligand) molecule. With the advent of computer-based techniques to screening potential molecules for drug-likeness, also called **high-throughput screening**, the binding affinity of several hundred thousand drug candidates can be determined within a day. Rational based drug design has the potential to reduce the cost and time for developing drugs drastically.

1.5 The docking problem

The docking problem can be classified into three kinds, depending on which molecules participate in docking — protein-ligand, protein-protein and protein-DNA.

The **Protein-ligand Docking** problem can be stated as follows:

Given a protein molecule (receptor) and an arbitrary small ligand molecule, predict whether the ligand will bind to the protein, and if so, predict the geometry and affinity of the binding.

Here, the geometry of the configuration should be the configuration of the system that minimizes its free energy (potential + entropy).

1.5.1 Components of a docking procedure

A docking procedure consists of three interrelated procedures [2]:

- Identification of the binding site.
- Search algorithm to effectively sample search space.
- Scoring function.

The location of the binding site can be obtained from comparison with other known protein structures or biochemical constraints. Also, **cavity detection** programs can be used to identify binding sites on a computer.

The search algorithm must **efficiently** sample the search space of the ligand-protein complex, i.e. the translation, rotation, and conformation space of the ligand relative to the protein.

The role of scoring function is two-fold – as a target function for the search algorithm (a quantity to be optimized), and to give a ranking to the set of final solutions generated by the search.

1.5.2 Why is docking computationally difficult?

Several aspects make docking hard to solve [2, 3]:

- Large number of degrees of freedom — Includes:
  - relative orientation of the two molecules.
  - conformational flexibility of ligand.
  - partial flexibility of protein.

Typically, the conformational search space for a protein-ligand complex is of the order of billions, hence, ruling out exhaustive enumeration of all possible complex structures.

- Lack of accurate scoring function — Currently, popular scoring functions simplify many of the chemical interactions to enable quick computation, at the cost of accuracy. A full molecular dynamics based simulation is prohibitively expensive for docking of a large number of molecules for screening purposes.

Depending on the degrees of freedom available to the protein and ligand molecules, docking approaches can be classified into the following :

- Both protein and ligand rigid — This is computationally the simplest, as the number of degrees of freedom of the complex is minimum i.e. 6 (3 (translation) + 3 (rotation)).

- Rigid protein and flexible ligand — Ligand flexibility is modeled by rotatable bonds. The number of degrees of freedom equals (6+N), where N is the number of rotatable bonds in the ligand.

- Both protein and ligand flexible — This is computationally the most difficult as the search space is huge, even if protein flexibility is limited to typically a small fraction of all protein atoms.

In this dissertation, our focus will be on docking a *flexible* ligand into a *rigid* protein.

1.6 Application of computational drug-docking

Computational drug-docking has immense application in the pharmaceutical industry. Currently, the search for novel drugs, for diseases that afflict the human society, is very time-consuming and laborious, apart from being too expensive. If the computer is able to predict a subset of compounds, among those currently available, which qualify as potential drug candidates, the experimentation and testing aspects of the drug development phase can be targeted towards such promising compounds only. This will help save huge amounts of time, labor and money.

Another application is *de novo* ligand design in which a *combinatorial library* of *virtual* ligands are designed on a computer, followed by docking of these molecules to screen for candidate drugs. Time-efficient docking algorithms are required to screen a huge database of virtual ligands.

Chapter 2

Overview of evolutionary computation

2.1 Introduction

The term **evolutionary computation** (EC) refers to the study of the foundations and applications of heuristic techniques based on the principles of **Darwinian evolution** – survival of the fittest and descent with modification. These techniques are collectively called **evolutionary algorithms** (EAs).

2.2 EAs — nature-inspired computing

EAs mimic the process of natural evolution. Evolution is a process operating over **chromosomes** rather than over organisms. Chromosomes encode the structure of a living being. In addition, Darwinian principle of "survival of the fittest" or in other words, **natural selection** is the mechanism that relates chromosomes with the efficiency of the entity they represent, thus allowing those efficient organisms which are well-adapted to the environment to reproduce more often than those which are not. The evolutionary process takes place during the reproduction stage. Each of the selected chromosomes is subjected to a **variation** operator. Most common ones are *mutation* and *crossover*.

2.3 EAs as an optimization tool

EAs have been widely used in numerous combinatorial optimization problems involving large search spaces. The process of evolution searches only a very small portion of the potential space, but at the same time, develops better solutions to the problem at hand. Simulation of this method on the computer has enabled significant progress in solving several hard optimization problems encountered in biological domain, such

as drug docking, protein folding and biological sequence analysis. However, one characteristic feature that distinguishes EAs from classical optimization approaches, is that they do not guarantee convergence to the global optimum. In practice, the solutions provided by EAs are sufficiently close to the global optimum. An additional benefit of EAs over conventional approaches is that they can be parallelized easily, further increasing their efficiency.

2.4 Constituents of EAs

A variety of evolutionary algorithms have been proposed. The major ones being:

- **Evolution Strategies** (ES)
- **Evolutionary Programming** (EP)
- **Genetic Algorithms** (GA)
- **Genetic Programming** (GP)

While all of the above approaches give preference to the fitter individuals in a population, the differences between them can be attributed to the relative importance given to each of the variation operators e.g. GAs typically give more preference to the *crossover* operator whereas in ESs, ***mutation*** is the primary operator used.

2.5 Outline of an EA

An evolutionary algorithm is an iterative and stochastic process that operates on a set of individuals called **population**. Each individual represents a potential solution to the problem being solved. This solution is obtained by means of an encoding/decoding mechanism. Initially, the population is randomly generated (perhaps with the help of a construction heuristic). Every individual in the population is assigned, by means of a **fitness function**, a measure of its goodness with respect to the problem under consideration. This value is the quantitative information the algorithm uses to guide the search. **Selection** of individuals takes place according to a **selection strategy**, with fitter individuals having a higher probability of getting selected. The selected individuals are allowed to undergo **reproduction** through application of evolutionary operators, to generate a new population, and the cycle repeats. The steps involved in an EA are outlined in Algorithm 1.

Since, modified GAs have been used in the current problem, let us briefly review them first.

Algorithm 1 Evolutionary Algorithm

```
1: t ← 0 {initialize time}
2: Generate [P(0)]
3: while NOT Termination_Criterion [P(t)] do
4:    Evaluate [P(t)]
5:    P' (t) ← Select [P(t)]
6:    P"(t) ← Apply_Reproduction_Operators [P'(t)]
7:    P(t+1) ← Replace [P(t), P"(t)]
8:    t ← t + 1
9: end while
10: RETURN Best_Solution
```

2.6 Genetic algorithms

GAs are search techniques based on the mechanics of natural selection and evolution [4, 5]. The following are the defining characteristics of a GA.

2.6.1 Encoding

Each individual is encoded in a binary string, also called the *genotype*. The *phenotype* or actual interpretation can be got back by decoding individuals from the genotype.

However, the binary encoding of chromosome has been found to be inadequate for optimization problems in real-valued spaces. Hence, a **real-coded GA** is often used, in which a chromosome consists of a set of real-valued parameters.

2.6.2 Selection

A user-defined fitness function is used to score each solution. Based on the fitness scores, various selection methods are used:

- *Proportional selection* — Probability of selection varies directly with fitness value e.g. *Roulette wheel* selection.
- *Tournament selection* — Utilizes competition between each individual and a randomly chosen set of n other individuals in the population. In each competition, the individual with the higher fitness value is assigned a "*win*". After all competitions are completed, the individual are ranked with respect to the number of "*wins*", and the lowest scoring half of the population is removed. The remaining individuals become parents for the next generation.

2.6.3 Variation

The variation operators used are :

- **Crossover** – For crossover, individuals of the parent generation are randomly grouped pairwise. Then, one of the several crossover schemes are applied:

  - *Single-point* - One crossover point is selected, binary string from the beginning of the chromosome to the crossover point is copied from the first parent, the rest is copied from the other parent.
  - *Two point* - Two crossover points are selected, binary string from the beginning of the chromosome to the first crossover point is copied from the first parent, the part from the first to the second crossover point is copied from the other parent, and the rest is copied from the first parent again.
  - *Uniform* - Bits are randomly copied from the first or from the second parent.

- **Mutation** — Mutation is done by flipping a randomly chosen bit of a chromosome.

Both the variation operators are applied to selected individuals with a certain probability. Since, GAs lay more stress on the crossover operator, the probability of crossover (*pcross*) is typically taken to be high (e.g. 0.8) whereas mutation probability (*pmut*) is taken to be low (e.g. 0.05).

2.6.4 Elitism

Elistism preserves the best solution obtained in the current generation. Elitism has been found to be useful in problems involving function optimizations.

Chapter 3

GA-LS hybrids

3.1 Local search algorithms

A **local search** (LS) **algorithm is one that iteratively improves its estimate of the minimum (or maximum) by searching for better solutions in a local neighborhood of the current solution. The neighborhood of a LS algorithm is the set of solutions that can be reached from the current solution in a single iteration of the LS operator.** The termination of the algorithm is governed by a **stopping criterion** and the output of the LS algorithm is typically one of the local minimas of the objective function used.

3.2 Local search operators

Several local search operators have been proposed in the literature [6]. Broadly speaking, local search operators can be classified into two categories:

- Gradient-based Search
- Random Local Search

3.2.1 Gradient-based search

Gradient methods for local search require the use of the derivative of the objective function. The minimization is based on the steepest descent principle, wherein, the search moves along the local downhill gradient direction $-\nabla f(x)$ by a step proportion to the magnitude of the gradient at that point. Gradient-based methods have been used extensively in several problem domains, such as the learning algorithm used in multilayer perceptrons, called the *backpropagation* algorithm. The stopping criterion for search is governed by convergence to one of the local minima of the objective function, where the value of the gradient becomes 0 and no further moves are possible.

3.2.2 Random Local Search

Unlike gradient-based ones, random local search methods do not use the derivative information. For instance, the **Solis-and-Wets** operator, uses normally distributed steps to generate new points in the search space. A new point is generated by adding zero mean normal deviates to every dimension of the current point. If the value of the new point is worse than than the current point, a new point is generated by taking a step in the opposite direction from the new point. If neither point is better than the first point, another new point is generated.

The above operator automatically adjusts the variance of its normal deviates based on the rate at which better solutions are found. If the new solutions are better sufficiently often, the variance is increased to allow the algorithm to take larger steps. If poorer solutions are frequently generated, the variance is decreased to focus the search near the current solution. The algorithm is terminated, after a fixed number of iterations, or when the step size becomes smaller than a given threshold.

3.3 GA-LS hybrids

3.3.1 Why hybridize?

Two competing goals govern the design of global optimization methods. *Global reliability* is needed to ensure that every part of the domain is searched to provide a reliable estimate of the global optimum. *Local refinement* is important since the refinement of the current solution will often produce a better solution. Thus, any global optimization algorithm, such as a genetic algorithm, must try to achieve both goals using a combination of global strategy and local strategy.

A hybrid GA-LS approach possesses both the global optimality of the genetic algorithm and the convergence of the local search. In addition to finding better solutions than the GA, they also optimize more efficiently.

3.3.2 Models of hybridization

Several models of GA-LS hybrids have been proposed. They include the two-phase approach and the Lamarckian and Baldwinian models.

Two-phase approach

In this approach, the search process is divided into two phases. In the first phase, a standard GA is used to search for "good" solutions, i.e. solutions close to the global optimum. Once the population has converged to such a solution, search enters the second stage, wherein a local search is employed to search refine the solutions obtained during the first phase.

Lamarckian model

According to the Darwinian model of evolution, transfer of information is one-way, i.e. from the genotype to the phenotype. However, in those cases, where an inverse **mapping function** (from phenotype to genotype) exists, it is possible to finish a local search by *replacing* the individual with the result of the local search. This strategy is called the **Lamarckian Genetic Algorithm** (LGA). LGA is the based on the (discredited) assertion made by Jean Batiste de Lamark that phenotypic characteristics acquired during an individual's lifetime can become heritable traits. Figure 3.1 contrasts Lamarckian and Darwinian models of evolution. The local search process is shown on the left hand side and pure Darwinian mechanics is shown on the right.

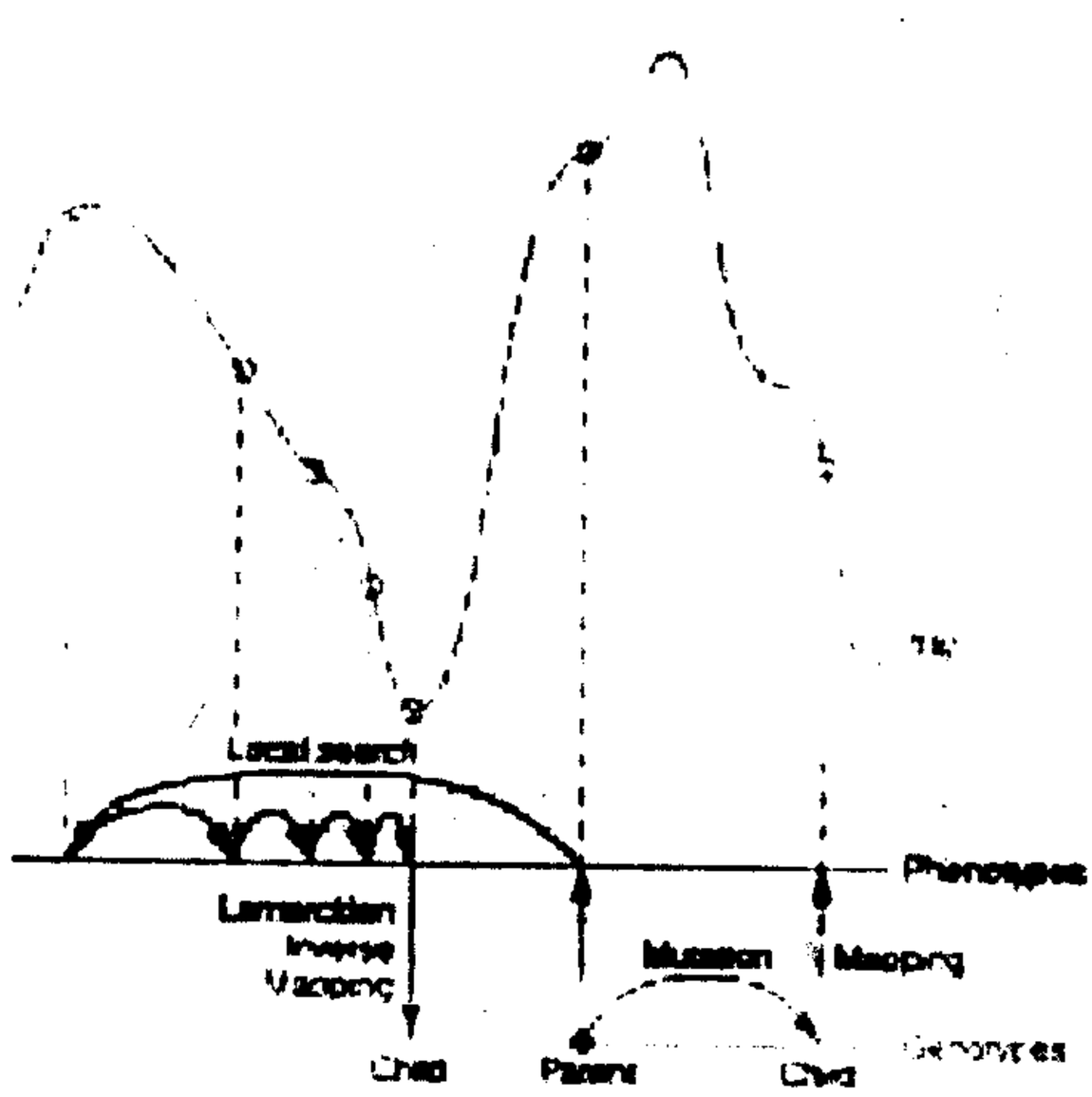

Figure 3.1: Lamarckian v/s Darwinian evolution

The LGA has been successfully used for the docking of flexible ligands into rigid proteins in the program AutoDock [7]. In AutoDock, the genotype consists of a string of real-valued genes: three Cartesian coordinates for the ligand translation, four variables for the ligand rotation, and a set of N angles for the N rotatable bonds in the ligand. After applying selection and crossover operators, a local search is performed on a fraction of the new population. The local search method is based on the Solis-and-Wets operator. At the end of the local search, the new chromosome replaces the old one.

Baldwinian model

In contrast to the Lamarckian model, Baldwinian model of evolution does not allow a parent to pass on its learned characteristics to its offspring; instead only the fitness is retained. Therefore, after learning, the chromosome is associated with a "learned" fitness that is not the same as its "inborn" fitness. Baldwinian learning has been shown to be able to direct genotypic changes.

Chapter 4

Approaches to drug docking

As briefly mentioned in Chapter 1, a docking method is composed of three components - identification of the binding site, search algorithm and scoring functions.

4.1 Identification of the binding site

Most docking methods rely on the binding site being predefined, so that the search space is limited to a comparatively small region of the protein. The binding site may be identified by comparison with the protein co-crystallized with a different ligand, or comparison with proteins of similar functions. In absence of such information, computer-based techniques rely on **cavity detection** programs [8].

4.2 Search algorithms

A variety of search algorithms have been applied to the docking problem, the prominent ones being matching algorithms for rigid body docking, incremental construction methods, simulated annealing and population-based methods such as genetic algorithms and evolutionary programming.

4.2.1 Matching algorithms

The earliest docking algorithms considered protein and ligand molecules as rigid bodies, in order to reduce the search space. One of the first, and currently, one of the most widely used software tools for molecular docking, is DOCK [9]. DOCK represents the ligand molecules as rigid spheres and tries to optimize their positions using a *distance-compatible matching* criterion. A set of atoms with given chemical properties are matched to complimentary atom positions within the binding pocket, and the interatomic distances between ligand atoms provide constraints on the solution set.

However, the utility of rigid-body docking algorithms is limited because ligand molecules and, to some extent, the protein molecules exhibit **conformational flexibility**. To account for this, later versions of DOCK programs use an **ensemble** of rigid structures, each representing a different conformation of the same ligand. Several other docking algorithms such as LIGIN [10] and FLOG [11] are also based on similar ideas.

4.2.2 Incremental construction

Incremental construction techniques use a piecewise assembly of the ligand within a defined binding pocket. Rigid fragments are generated from the ligand by breaking it at rotating bonds, to create a set of fragments to be used by the docking algorithm. The algorithm starts by placing one fragment (called **base** or **anchor** fragment) into the active site of the protein. Usually, the largest fragment is chosen as the base fragment. Then, the remaining parts of the ligand are added to the already placed fragment, iteratively. Several software tools use incrèmental construction techniques for docking purposes, most notable among them are FlexX [12] and LUDI [13].

4.2.3 Simulated annealing

Simulated annealing [14] is based on a model of cooling processes. In this model, the initial state of the system has random thermal motion or "high" temperature. This random motion is decreased over time (using a **cooling schedule**), until a final stable docked position is obtained. The random motion of the ligand allows for exploration of the local search space, and the decreasing temperature of the system acts to drive it to a minimum energy.

Simulated annealing has been applied to numerous docking procedures, such as an early version of AutoDock [7].

4.2.4 Genetic algorithms

Molecular docking is a difficult optimization problem, due to the combinatorial explosion caused by the many degrees of freedom of the molecules. Thus, it requires efficient sampling across the entire range of positional, orientational, and conformational possibilities. GAs fulfill the role of global search particularly well, and have found successful application in drug docking.

Jones et al.[15] introduced a program GOLD (Genetic Optimization for Ligand Docking) which incorporated full ligand flexibility with partial flexibility of the protein. Each chromosome encoded the internal coordinate of both the ligand and protein active site, and a mapping between hydrogen-bonding sites. Fitness was determined by summing the hydrogen-bonding sites, the pairwise interaction energy between the ligand and protein atoms, and the ligand steric and torsional energies.

4.3 Scoring function

Given a molecular docking algorithm, a means is required to identify the best of a set of possible solutions. A scoring function provides an estimate of the strength of interaction between the protein and the ligand, and can be used to rank the possible solutions. There exist primarily three kinds of scoring functions:

- Free energy perturbation (FEP)
- Empirical scoring function
- Knowledge-based scoring function

4.3.1 Free energy perturbation

These are the most rigorous approaches for calculating binding affinities as they deal with **molecular dynamics** simulations, considering, for example, quantum-mechanical effects and solvent molecules. However, these methods are less practical for solving drug docking problems as they require huge computational resources. Execution time per protein-ligand conformation varies from minutes to hours of computation time.

4.3.2 Empirical scoring function

Empirical scoring functions are the most common means of estimating a binding affinity and have become quite popular in drug-docking studies. The basic assumption underlying such approaches is that the overall binding free energy can be partitioned into a set of components. Thus, the free energy of binding is written as a sum of terms, accounting for various effects such as **hydrogen bonding**, **torsional effects**, and interactions such as **van der Waals**, **electrostatic**, and **hydrophobic**. Empirical scoring functions were pioneered by Böhm [13]. A typical function is given in Equation 4.1

$$\begin{aligned} E &= E_{bond} + E_{phi} + E_{tor} + E_{impr} \\ &\quad + E_{vdW} + E_{elec} + E_{H} + E_{sol} \end{aligned} \tag{4.1}$$

where E_{bond}, E_{phi}, E_{tor}, E_{impr}, E_{vdW}, E_{elec}, E_H and E_{sol} refer to the bond length potential, bond angle potential, torsion angle potential, improper torsion angle potential, van Der Waals pair interactions, electrostatic potential, hydrogen bonds and solvent interactions respectively. This particular energy function has been used in the program CHARMm [16].

The weighting of each of the interaction terms is obtained by fitting a regression model to a test set of ligand-protein complexes.

4.3.3 Knowledge-based scoring function

These functions are based on statistical analysis of pairing frequencies of atoms within known ligand-protein complexes, with the underlying assumption that the more favorable the interaction is, the greater is the frequency of occurrence. Thus, the total interaction is approximated by the sum of pairwise potentials between the atoms. Several such functions have been developed in recent years, e.g. PMF [17], BLEEP [18], and DrugScore [19].

Chapter 5

A GA-LS algorithm for flexible docking

In this dissertation, an attempt is made to tackle the "*flexible ligand docking problem*" using a hybrid GA-LS algorithm modeled on the principle of Lamarckian genetics. Further, we have tried to modify the local search operator used, in order to maintain the diversity of the solutions throughout the run of the algorithm.

In the following sections, the details of the docking procedure are presented.

5.1 Input parameters

The input to the algorithm consists of several files containing the structures of the protein and the ligand, a file containing the parameters governing various aspects of the GA-LS run, and the approximate location of the binding site region.

The protein structure information is obtained from the Brookhaven Protein Data Bank (PDB) format. The 3-D structure of the ligand molecule is stored in the Tripos MOL2 format. This file contains, besides the coordinates of the atoms, the information about charges on each atom, the type of bonds and whether the bond is rotatable or not (to calculate changes in atom coordinates due to torsional rotation about those bonds). In addition, the scoring function requires the use of two files containing data about the physical and chemical aspects of the ligand atoms and the protein residues like VDW radii, atomic weight, hydrogen bonding character, hydrophobicity etc.

The GA-LS parameter file contains all the parameters required during the run of the algorithm e.g. population size, crossover and mutation probabilities, maximum number of function evaluations, various parameters controlling the local search (kind of operator, step sizes for each of the translation, rotational and torsional parameters) etc.

The location of the binding site is specified by the approximate coordinates of the active site and a *bounding box* within which the GA-LS algorithm will limit its search.

This helps to reduce the search complexity as the algorithm is not required to search the entire protein molecule.

5.2 Preprocessing stage

After extracting the coordinates of the protein and the ligand from their respective files, the first task is to place the ligand at the binding site. This is because the translation values encoded in the chromosome are defined w.r.t. binding site. Placement of the ligand is accomplished by translating all the ligand atoms by an amount equal to the difference in coordinates between the binding site and the mean coordinate of all ligand atoms.

Next, for each rotatable bond, we maintain a list of all the ligand atoms "affected" by rotation of this bond. Since, the rotation of a bond only modifies the relative position between any pair of atoms connected to *opposite* ends of the bond, we obtain two sets, one containing all atoms connected to one end of the bond and the other, of atoms connected to the opposite end. This is done by using a breadth-first search (BFS) on the graph defined by considering the atoms as nodes and bonds as edges. The set of "affected" atoms is taken to be the set having smaller cardinality (to reduce computation time).

The search algorithm for our drug docking problem is composed of two components — the global genetic component and the local search component.

5.3 Genetic component

The current problem uses a **real-coded** GA to encode the conformation of the ligand relative to the protein. Each chromosome is composed of a string of N real valued genes: three Cartesian coordinates for the ligand translation, three for rotation, and the remaining $(N-6)$ for torsional angles of the rotatable ligand bonds.

A user-defined parameter, *popsize*, gives the total number of individuals in the population. Initially, for each individual, the values for the translation parameters, x, y and z, are randomly chosen within the limits specified by the binding site. The angle parameters are chosen to uniformly lie within a range of 0^0 and 360^0.

Thereafter, each generation of the GA run consists of several stages:

- Mapping
- Fitness evaluation
- Selection
- Crossover
- *Local Search*

- Mutation
- Elitism

It may be noted that the local search stage is not exactly a part of the GA component. Local search is discussed in detail in section 5.4.

5.3.1 Mapping

Mapping translates from an individual's genotype to its phenotype. In the current problem, the task of the mapping function is to convert the translation, rotation and torsional parameters into the actual coordinates of the ligand atoms.

The coordinates of each atom are obtained by applying the following set of transformations. First, the ligand atoms affected by torsional rotation are rotated about their corresponding rotatable bonds. Next, the whole molecule is rotated about the X, Y and Z axes. Since, the rotation axes must pass through the ligand molecule, the ligand molecules are initially translated, so that the origin of the coordinate system coincides with the center of the ligand molecule. After the rotation, the atoms are translated back to their original location. Finally, the translation operator is applied to each of the ligand atoms to obtain the final coordinates of the molecule.

5.3.2 Fitness Evaluation

The role of the scoring function to evaluate the binding affinity between the protein and the ligand atoms is very crucial for successful docking. Hence, in our current endeavor, we have used the well-known scoring function, **SCORE** [20], to rank the solutions provided by the GA. Basically, SCORE is a *linear* empirical method for estimating the absolute binding affinity of the protein-ligand complex with known three-dimensional structure. The model takes into account several factors such as Van der Waals (VDW) interaction, metal-ligand bonding, hydrogen bonding, desolvation and deformation effects. The model parameters were obtained from regressional analysis upon a training set composed of 170 complex structures from the PDB. The output of score is expressed in the negative logarithm of the dissociation equilibrium constants, i.e. pK_d. The pK_d values range from 1.54 to 13.96, covering over 12 orders of magnitude.

Since, our GA was designed keeping minimization of objective function in mind, the fitness value is taken to be the negative of the value returned by SCORE.

5.3.3 Selection and Variation

Proportionate selection is used to decide which individuals will reproduce. Two-point crossover is used with the crossover points chosen only at gene boundaries. This is to prevent large and random changes in the value of the genes. Since we are using

real variables, bit-flip mutation can't be used. Instead, mutation is performed by adding a random real number having a Cauchy distribution to the variable, as shown in Equation 5.1.

$$C(\alpha, \beta, x) = \frac{\beta}{\pi(\beta^2 + (x - \alpha)^2)}; \alpha \geq 0, \beta > 0, -\infty < x < \infty. \tag{5.1}$$

Generational replacement of population is performed. Also, a user-selected flag is used to decide whether elitism will be applied. If true, the worst individual among those obtained after application of selection and variation operators is replaced by the best individual in the previous generation. The termination of the run is decided by checking if the number of function evaluations performed till now, exceeds a user-defined maximum value. The final output of the algorithm is a set of K top-ranked protein-ligand complexes.

5.4 Local Search Component

For the local search, we have used the Lamarckian model in which a chromosome is replaced by a better chromosome in its neighborhood. At each generation, a user-defined fraction of the population undergoes such a local search. The individuals undergoing local search are chosen from the population in an unbiased manner.

Two search strategies have been implemented and their performance compared. One of them is the Solis-Wets random operator, described in chapter 3. The other one is inspired from the mechanics of simulated annealing and attempts to overcome the problem of getting trapped in local minima by a self-adaptation mechanism. The algorithm is described below:

5.4.1 Self-adaptive LS

In this method, the *self-adaptation* of the local search is governed by a temperature parameter. This temperature determines the degree, by which, uphill moves will be allowed. At the start of each local search stage, the temperature parameter is set (Equation 5.2).

$$temperature = \frac{1}{|maxFitness - minFitness|} \tag{5.2}$$

where $maxFitness$ and $minFitness$ are the maximum and minimum values of the fitness function, respectively. As the temperature is inversely proportional to the spread of fitnesses within the population, when the latter converges, the former rises. Therefore, each individual in the population will be more "nervous" and will try to move away from its initial position, exploring the search space. Eventually, the fitnesses will spread, lowering the population temperature. However, in this method,

we don't allow the best individual to perform a local search to maintain the overall best fitness.

Chapter 6

Results and discussion

This chapter describes the results of running the GA-LS algorithm described in Chapter 5 for docking of several well known ligands to their corresponding proteins. Three different methods (GA, LGA with Solis-Wets local search (LGA-SW), and LGA with self-adaptive local search LGA-SA) have been employed for docking of each complex. Results of all the three methods have been compared and critically analyzed. In cases where high deviation from the experimentally-determined complex structures have been found, we have tried to analyze the causes for failure.

6.1 Test complexes

For our current docking experiments, we chose three protein-ligand complexes having varying degrees of ligand flexibility:

- β-Trypsin/Benzamidine
- Streptavidin/Biotin
- DHFR/Methotrexate

The pdb code, resolution, number of rotatable bonds of the ligand molecule and the energy of the crystal structure for the above complexes is given in Table 6.1.

The 3D structures of the protein-ligand complexes were obtained from the Brookhaven Protein Data Bank (PDB). The ligand structure was obtained in the Tripos Mol2 format.

6.2 Search-specific parameters

In all of the methods, the population size (*popsize*) was taken to be 50. Increasing the value of *popsize* was found not to significantly affect the results obtained, in addition to requiring more computational cost.

Table 6.1: Protein-ligand complexes used in docking experiments

Protein-ligand complex	PDB Code	Resolution (A^0)	No. of rotatable bonds of ligand	Energy (kcal mol^{-1})
β-Trypsin/ Benzamidine	3ptb	1.7	0	-7.86
Streptavidin/Biotin	1stp	2.6	5	-8.86
DHFR/methotrexate	1dra	1.7	7	-13.64

Since, the number of function evaluations per generation is not constant for the LGA dockings, the maximum number of function evaluations in the entire run (*maxeval*) was fixed prior to the start of the algorithm. In our docking experiments, we have taken *maxeval* to be vary between 30,000 and 50,000. The choice of such a value was governed by the maximum affordable computational time for a single docking while at the same time ensuring that the final solution set is fairly converged.

The parameters for the variation operators were taken to be 0.80 and 0.02, for the crossover and mutation operators respectively. The parameters for the Cauchy distribution used in mutation were $\alpha = 0$ and $\beta = 1$.

A number of parameters were used to control the exploitation abilities of the local search. For the search based on the Solis-Wets method, the local search frequency, i.e. the fraction of population undergoing local search at every generation, was chosen to be 0.1. Since, translation is more sensitive to large step-sizes than either orientation or torsional rotation, the initial step size for translation was taken to be $0.2A^0$, and for both rotation and torsion, it was 5^0. Another important parameter for the above method is the maximum number of local moves per individual. Higher the value, more is the exploitation of the search space, however, it can also lead to premature convergence. For the present experiment, a value of 10 was found to be a good compromise. The maximum number of consecutive successes or failures before doubling or halving the search step, ρ, was 4.

For the self-adaptive local search described in 5.4.1, the only parameter needed for the controlling the search is the search frequency, which was chosen to be 0.5; since, each individual performs only a single energy evaluation, a higher value (compared to the Solis-Wets method) is acceptable.

6.3 Docking results

For each of the three protein-ligand complexes, the search algorithms of chapter 5 were run, resulting in a total of nine dockings. The output was obtained in the form of a PDB file containing the coordinates of the protein and ligand atoms, together with the computed **root-mean-square-deviation** (rmsd) between the docked structure and the coordinates determined from X-ray crystallography. On an average, a single docking took 1.5 hours (using *maxeval* = 50,000) on a Pentium IV machine, the

bulk of which was accounted for by the scoring calculations. The details of docking simulations on individual complexes is presented below:

6.3.1 β-Trypsin/Benzamidine

The experimentally obtained complex structure (see Figure 6.1 (a)) shows that benzamidine binds tightly in a specific pocket of trypsin. The coordinates of the binding site was taken to be $(0, 1.5, 0)$ and the bounding box had dimensions of $10 \times 10 \times 10$. All the algorithms were run for $30,000$ function evaluations. The results of binding are shown in Table 6.2 and the final complexes for the GA, LGA-SW and LGA-SA are shown in Figure 6.1 (b),(c) and (d) respectively.

As can be seen from the results, all the three methods are able to approximately identify the location of the binding site though not the actual orientation. The rmsd is highest for the GA, and lowest for LGA-SA, showing the effect of local search in fine-tuning the results.

6.3.2 Streptavidin/Biotin

Biotin binds very tightly to streptavidin due to the formation of hydrogen bonds and van der Waals interactions (see Figure 6.2 (a)). In our simulation experiments, the binding site was $(-11, 2, -10)$ with the same bounding box as above. All the algorithms were run for $50,000$ function evaluations. The rmsd's obtained for the runs are shown in Table 6.2 and the final complexes are shown in Figure 6.2 (b),(c) and (d).

6.3.3 DHFR/Methotrexate

Dihydrofolate Reductase (DHFR) is an enzyme important for the process of cell growth in organisms. Thus, an inhibitor for DHFR can be useful to reduce the rate of cell growth. Methotrexate is such an inhibitor and is used as an anticancer drug in chemo therapy. In our simulation experiments, the binding site was $(20.62.65)$ and the bounding box had dimensions of $20 \times 20 \times 20$. All three algorithms were run for $50,000$ function evaluations. Unfortunately, due to the high degree of freedom (13), none of the algorithms were able to find a suitable conformation (having rmsd $< 10.4^0$). The reasons for failure of the search algorithms have been analyzed in section 6.4.

6.4 Analysis of results

From the results obtained above, it can be said that a satisfactory docking was obtained in the first 2 test cases by all the methods, with the local search based ones showing a slightly better performance. However, it can be observed that the above

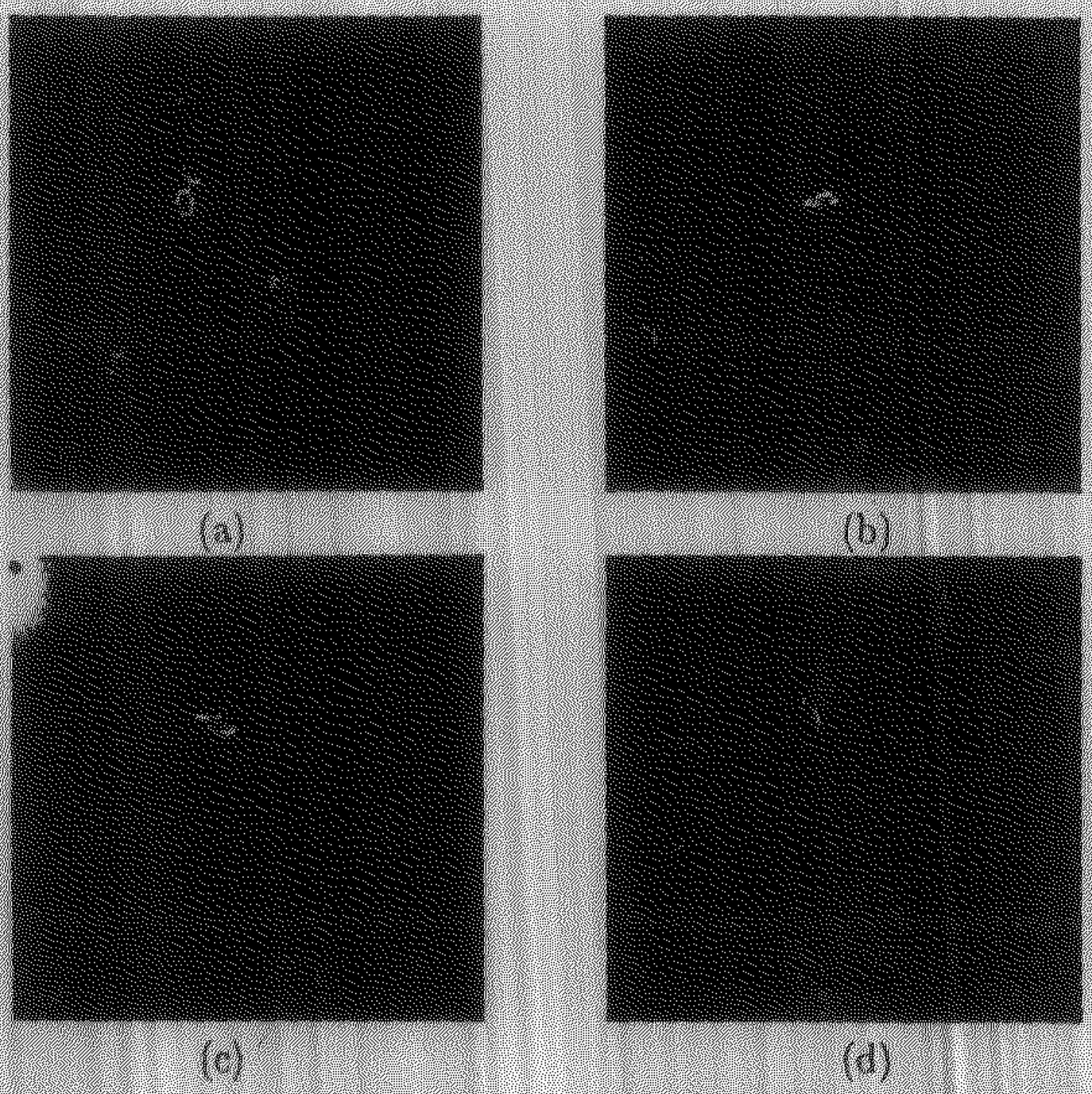

Figure 6.1: Docking results for β-Trypsin/Benzamidine. (a) Complex structure (b) Docking using GA (c) Docking using LGA-SW, and (d) Docking using LGA-SA.

Table 6.2: Results of docking experiments performed on 3 test complexes

Protein-ligand complex	GA		LGA-SW		LGA-SA	
	Score	RMSD	Score	RMSD	Score	RMSD
β-Trypsin/ Benzamidine	-5.74	6.66	-5.3	6.3	-5.71	6.36
Streptavidin/Biotin	-5.6	9.23	-6.9	8.56	-6.9	9.46
DHFR/methotrexate	-9.9	17.1	-10.2	18.6	-10.31	17.64

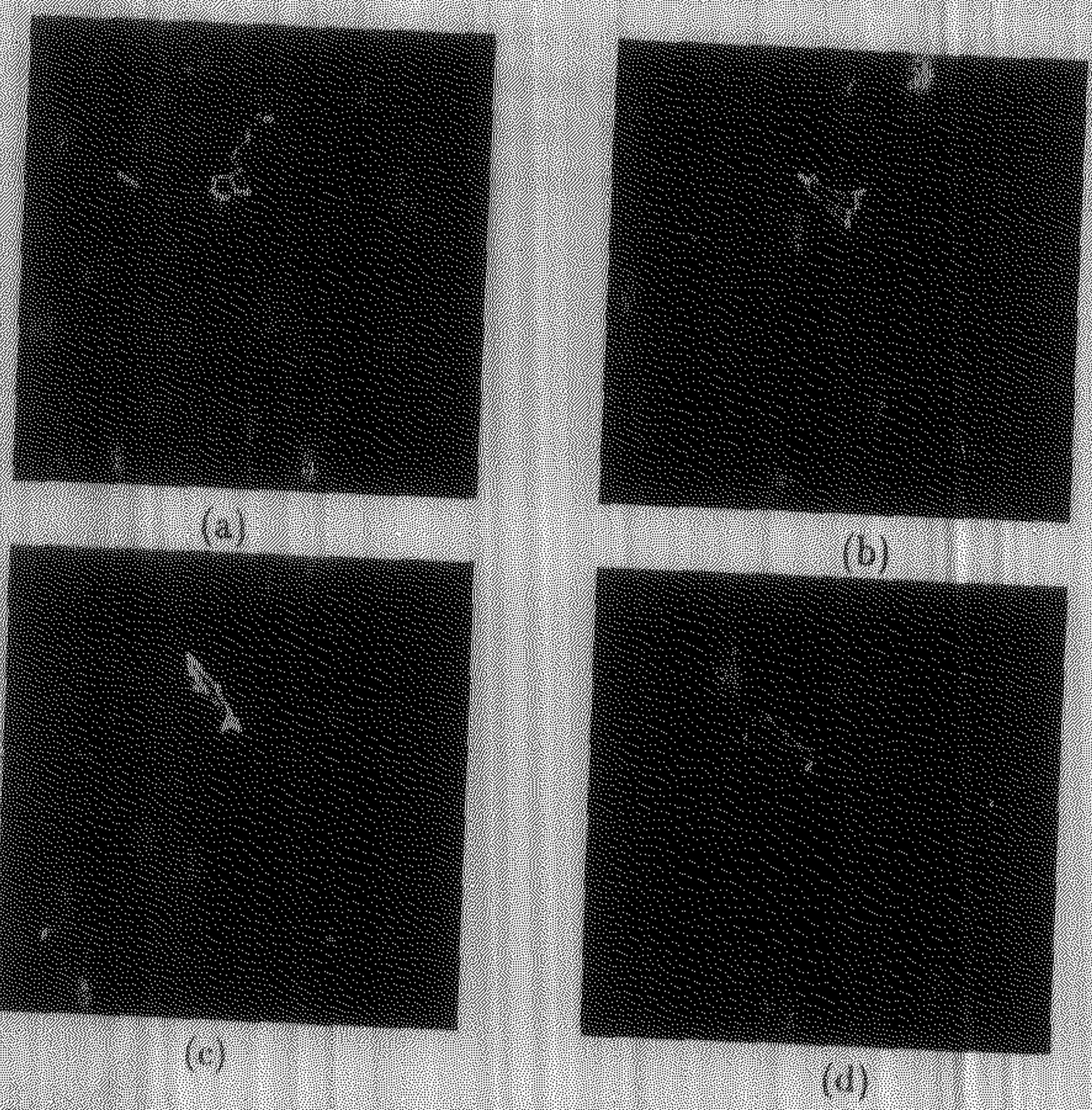

Figure 6.2: Docking results for Streptavidin/Biotin. (a) Complex structure (b) Docking using GA (c) Docking using LGA-SW, and (d) Docking using LGA-SA.

implementations have been able to obtain only moderate success in docking problems involving ligands with high number of degrees of freedom (> 10). The failure of the algorithms in test case 3 can be attributed to several reasons. First, the number of function evaluations were far less as compared to the algorithms reviewed in chapter 4 e.g. in [7], the maximum number of function evaluations were set to (1.5×10^6). However, in our current situation, such a high number was computationally infeasible. since function evaluation (done by SCORE) was fairly slow. Increasing the number of evaluations in test case 3 to $1,00,000$ gave us marginally improved results. Secondly, the choice of the scoring function is also believed to have affected the results. As pointed out in [20], SCORE makes a few assumptions about the ligand structure (e.g. the similarity between the modeled structure and the actual structure). In cases where the above conditions are violated, SCORE is not able to accurately predict the binding affinity. A more robust and fast scoring function is thus required.

Chapter 7

Conclusions and Future Work

This dissertation attempts to investigate the application of hybrid genetic algorithms to tackle the problem of molecular docking of a flexible ligand into a rigid protein. It is believed that the hybrid GAs will be more successful in solving the docking problem by combining global exploration of the search space along with local exploitation of neighboring solutions. Preliminary results obtained on simulation of three protein-ligand test complexes have provided positive leads. While methods using a GA-LS hybrid has shown certain improvement over pure GAs, a full study taking into account various parameters controlling the search process needs to be done. Also another interesting direction of work is the design of a good scoring function, which is very crucial for successful docking.

Bibliography

[1] Thomas Lengauer (Ed.). *Bioinformatics — From Genomes to Drugs.* Wiley-VCH, GmbH,Weinheim, 2002.

[2] Vladimir Sobolev Brendan J. McConkey and Marvin Edelman. The performance of current methods in ligand-protein docking. *Current Science.* 83:845–856. 2002.

[3] George N. Philips Jr. Miguel L. Teodoro and Lydia E. Kavraki. Molecular docking : A problem with thousands of degrees of freedom.

[4] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[6] William E. Hart. *Adaptive Global Optimization With Local Search.* PhD thesis, University of California, San Diego, 1994.

[7] David S. Goodsell et al. Garret M. Morris. Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function. *Journal of Computational Chemistry,* 19:1639–1662, 1998.

[8] G. Patrick Brady Jr. and Peter F. W. Stouten. Fast prediction and visualization of protein binding pockets with pass. *Journal of Computer-Aided Molecular Design,* 14:383–401. 2000.

[9] J. M. Oatley et al. I. D. Kuntz. A geometric approach to macromolecular-ligand interactions. *Journal of Molecular Biology,* 161:269–288. 1982.

[10] R. C. Wade et al. V. Sobolev. Molecular docking using surface complementarity. *Proteins,* 25:120–129. 1996.

[11] S. K. Underwood et al. M. D. Miller. Flog - a system to select quasi-flexible ligands complementary to a receptor of known three-dimensional structure. *Journal of Computer-Aided Molecular Design.* 8:153–174. 1994.

[12] B. Kramer et al. M. Rarey. Predicting receptor-ligand interactions by an incremental construction algorithm. *Journal of Molecular Biology*, 261:470–489. 1996.

[13] H. J. Bohm. Ludi: Rule-based automatic design of new substituents for enzyme inhibitor leads. *Journal of Computer-Aided Molecular Design*, 6:593–606. 1992.

[14] E. H. L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley, Chichester, U.K., 1989.

[15] Peter Willet et al. Gareth Jones. Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology*, 267:727–748. 1997.

[16] R. E. Bruccoleri et al. B. R. Brooks. Charmm: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4:187–217, 1983.

[17] I. Muegge and Y.C. Martin. A general and fast scoring function for protein-ligand interactions: A simplified potential approach. *Journal of Medicinal Chemistry*, 42:791–804, 1999.

[18] Roman A. Laskowski et al. John B. O. Mitchell. Bleep - potential of mean force describing protein-ligand interactions: I. generating potential. *Journal of Computational Chemistry*, 20:1165–1176, 1999.

[19] M. Hendlich H. Gohlke and G. Klebe. Knowledge-based scoring function to predict protein-ligand interactions. *Journal of Molecular Biology*, 295:337–356, 2000.

[20] L. Liu et al. R. Wang. Score: A new empirical method for estimating the binding affinity of a protein-ligand complex. *Journal of Molecular Modeling*, 4:379–394. 1998.