Antibiotic Molecular Design Using Artificial Bee Colony Algorithm

By

Shweta Mapari

Submitted to the graduate degree program in Chemical and Petroleum Engineering and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Chairperson: Dr. Kyle Camarda

Dr. Kevin Leonard

Dr. Arghya Paul

Date Defended: 15th January, 2019

The dissertation committee for Shweta Mapari certifies that this is the approved version of the following thesis:

Antibiotic Molecular Design Using Artificial Bee Colony Algorithm

Chairperson: Dr. Kyle Camarda

Date Approved: 1st February, 2019

Acknowledgements

First, I would like to thank my advisor, Dr. Kyle Camarda for his excellent guidance and help in improving my thesis. I am thankful for all the wonderful friends I made in KU. I became very good friends with the roommates I have had over past three years, Unnati, Sravanthi, Ravali, Rashmi and Sayali. I will dearly miss them. I would like to thank Sirisha and Apoorv for helping me understand how Openbabel and Pybel work. I am grateful to Martha for always being so supportive and helpful. Finally, I can’t thank my family enough for their unconditional love and support.

Abstract

Research is acutely needed to develop novel therapies to treat resistant infections. This project aims to design a drug molecule via a computer aided molecular design approach to provide lead candidates for the treatment of bacterial infections caused by *Staphylococcus aureus*. In a recently published WHO report, a list of bacteria which pose the greatest threat to human health was given. The purpose of this report was to identify the most important resistant bacteria at global level for which immediate treatment is required. *Staphylococcus aureus*, which is on this list, is a pathogen causing infections such as pneumonia and bone disorders. A methodology which determines the structures of candidate antibiotic molecules is described. The Artificial Bee Colony algorithm has been used for the first time for molecular design in this work.

It is necessary to predict physical and/or biological properties of compounds in order to design them. The prediction of properties is performed using Quantitative Structure Property Relationships (QSPRs). QSPRs are equations, which are developed using reported data for properties of interest by the method of regression analysis. This work applies connectivity indices and 3D MoRSE descriptors to develop QSPRs. The properties used in this work are minimum inhibitory concentration and Log *P* values. 3D MoRSE descriptors have been used for the first time for molecular design in this work.

The QSPRs are combined with structural feasibility and connectivity constraints to formulate an optimization problem, which is a mixed integer nonlinear program (MINLP). Because of the large number of potential chemical structures and the uncertainty in the structure-property correlations, stochastic algorithms are preferred to solve the resulting MINLP. One stochastic algorithm which has shown promise to solve these problems is the Artificial Bee Colony algorithm, which relies on principles of swarm intelligence to find near-optimal solutions efficiently. The Artificial Bee Colony algorithm described in this work is used to derive solutions which serve as lead compounds for a narrowed search for novel antibiotics.

Results show that the ABC algorithm is very effective in finding near optimal solutions to the MINLP, which is a combinatorial optimization problem. Molecular structures were obtained by optimizing objective function for individual property values and simultaneously for both the properties.

Contents

List of Figures:

List of Tables:

Chapter 1

Introduction

Computer Aided Molecular Design (CAMD) has become a very important tool for molecular design in recent decades. The traditional approach of molecular design involves synthesis of compounds with laboratory experimentation, which is very expensive and time consuming. On the other hand, CAMD methods give quicker results at low costs. While there is no substitute for experimental validation, CAMD methodology creates molecules that are promising candidates for experimental studies (Churi and Achenie, 1996). A CAMD framework helps to create a compound with desired property values. Hence, it has been extensively used in creating a wide range of chemicals such as polymers, refrigerants, solvents (Camarda and Maranas,1999, Duvedi and Achenie,1996, Karunanithi *et al.*, 2005). Some of the important applications of CAMD include drug discovery and pharmaceutical product design (Schneider and Fetchner, 2005, Kapetanovic, 2008, Siddhaye *et al.*, 2004). In this work, an antibiotic molecule has been designed using a CAMD approach.

1.1 Motivation

Antibiotics have been helping us combat against various bacterial infections for a long time. However, bacteria develop resistance towards these drugs, often by developing alternative metabolic pathways to circumvent inhibited ones. WHO recently published an article that describes a list of bacteria which have become resistant to multiple antibiotics, thus specifying the necessity for developing new drug molecules (WHO Report, 2017). *Staphylococcus aureus* is a multi-drug resistant bacterium, which belongs to this list. It is a gram-positive, sphere shaped bacteria which often skin infections, pneumonia, soft tissue, endovascular and bone disorders (Lowry, 1998). In order to treat the infections caused by this bacterium effectively, a novel pharmaceutical is required. The traditional approach of developing a new drug is expensive, time consuming and requires much trial and error. However, computer aided molecular design (CAMD) is a tool that allows the creation of a molecule having certain property values, using comparatively less amount of time and money (Achenie *et al.*, 2002).

1.2 Methodology

The general methodology of CAMD is presented in Figure 1.1. When solving the forward problem, property values need to be estimated from the chemical structure. Property prediction is done with the help of structure-property relationships. The property values used in developing such relationships are found experimentally or taken from the published data and then correlated using molecular descriptors. A wide range of molecular descriptors are tested in correlating property values. The molecular descriptor values are calculated for the specific group of compounds of interest, and these values are correlated with experimentally determined property values to form quantitative structure-property relationships (QSPRs). The correlation that shows the best fit and the best predictive capability is chosen for property prediction for similar compounds not in the correlating set.

When the inverse problem is solved, a molecular structure is obtained that possesses certain property values. This structure is determined by creating a molecular design problem within an optimization framework, which is solved using a suitable algorithm. This work seeks to solve antibiotic molecular design problems using the Artificial Bee Colony algorithm.

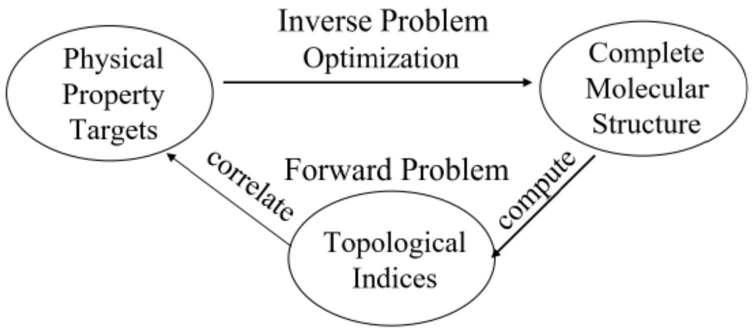


Figure 1.1: CAMD (Camarda and Sunderesan, 2005)

1.3 Molecular descriptors and Quantitative Structure Property Relationships

Molecular descriptors represent the structure of a molecule quantitatively. It is of prime importance that appropriate molecular descriptors are chosen to find an accurate correlation between property values and chemical structures. For a given data set of molecules, different molecular descriptors are calculated. A QSPR is derived for each property, correlating some number of descriptors to physical property values. The model which gives a high-quality fit, without over-fitting the data, is selected for molecular design and included in the molecular design optimization problem. The molecular descriptors present in this model are thus appropriate molecular descriptors for this class of molecules and for a given property. QSPRs in this work were obtained by correlating minimum inhibitory concentration (MIC) values and octanol-water partition coefficient values with connectivity indices and 3D MoRSE descriptors using the method of least squares.

1.4 Mathematical formulation of the design problem

The antibiotic design problem is an optimization problem, which contains both continuous and discrete variables with nonlinear constraints, resulting in a mixed integer nonlinear program (MINLP). Due to the non-convex nature of this problem, stochastic algorithms are preferred. We have used the Artificial Bee Colony (ABC) algorithm to find near optimal solutions to this problem. This is the first time, CAMD approach has been used for the design of an antibiotic molecule using the ABC algorithm.

1.5 Artificial Bee Colony algorithm

Different algorithms have been explored in the past to solve nonlinear optimization problems. However, a comparative study (Karaboga and Akay, 2009) has shown that ABC, an algorithm based on swarm intelligence, can perform better than other commonly used algorithms like genetic algorithms or simulated annealing. The ABC algorithm is based on the intelligent foraging behavior of a group of honeybees. The algorithm iterates through a set of phases until a near-optimal solution is obtained. More details on the ABC algorithm are given in Chapter Six.

1.6 Outline of the thesis

The introduction to the antibiotic design problem and motivation behind it are discussed in Chapter One. Background information on connectivity indices and 3-D MoRSE descriptors is given in Chapter Two. The development of QSPRs is discussed in Chapter Three. Chapter Four presents the mathematical formulation of the optimization problem intended for molecular design. The solution to the molecular design problem using the ABC algorithm is given in Chapter Five. Chapter Six contains the results and discussion of this work, followed by conclusions and future work in Chapter Seven.

Chapter 2

Molecular Descriptors

Molecular descriptors are numerical values which define attributes of a chemical structure. There are hundreds of different descriptors available, each of which describes a molecule's attributes differently. Some typical examples are molecular weight, total number of atoms in a molecule, or the number of single, double or triple bonds in a molecule. Sometimes, properties like melting point, boiling point, density can also function as descriptors (Terfloth and Gasteiger, 2018). In this work, two types of molecular descriptors are evaluated for use in QSPRs: connectivity indices and 3D-MoRSE descriptors.

2.1 Connectivity indices

Topological indices are a type of molecular descriptor which depends on the topology of molecule. There are many topological indices, including Randic's connectivity indices, Kier's shape indices and the Weiner index (Raman and Maranas, 1998). Molecular connectivity indices were introduced by Randic′ around 1975 (Randic′, 1975). Randic's connectivity indices, usually referred to as 'connectivity indices', have been tested in this work. They are a type of topological index, which provide information about the number and type of atoms present in a molecule as well as the bonding environment between adjacent atoms. They are calculated using simple formulae using concepts of graph theory. Numerous graph-theoretic approaches have been used previously in predicting the biochemical and biological properties of different types of chemicals successfully (Basak *et al*., 1987). One of these approaches is the use of connectivity indices, which have been used for prediction of biological properties successfully, and possess a huge potential in the process of finding drug leads (Dang and Madan, 1994, Galvez *et al*, 1995, Estrada *et al.*,2002).

For a successful molecular design, two hurdles must be overcome. The first one is developing the quantitative structure-property relationships with enough accuracy, and the second one is solving the optimization problem in a reasonable amount of time. Connectivity indices are easy to compute, and are able to correlate various properties of drugs such as solubility and density (Kier and Hall, 1976) quite well. Thus, they help with both of the hurdles mentioned previously quite well. They have been successfully implemented within an optimization framework for molecular design in the past. For example, connectivity indices were used for designing polymers (Camarda and Maranas, 1999), metal catalysts (Chavali *et al.*, 2004), soybean oil products (Camarda and Sunderesan, 2005), ionic liquids (McLeese *et al.*, 2010), pharmaceutical products (Siddhaye *et al.*, 2000) and selecting the best substituent for compound having desired fungicidal properties (Raman and Maranas, 1998). Hence, connectivity indices were considered as a good choice for molecular descriptors in this work.

2.1.1 Basic groups and molecular graphs

To calculate connectivity indices, a molecule is divided into small units called 'basic groups'. A basic group is defined as a non-hydrogen atom, along with its attached hydrogen atoms, in a particular valence state. The basic groups used in this work along with their corresponding atomic connectivity indices are shown in Table 2.1. A molecular graph is defined as the graph of a molecule where atoms and bonds correspond to vertices and edges respectively (Raman and Maranas, 1998). The basic groups are represented by vertices in a molecular graph, and the bonds are represented by edges. The conventional and molecular graph representations of n-butanol are shown in Figures 2.1 and 2.2 respectively. In the conventional representation, the numbers 1 to 5 shown underneath the groups indicate the corresponding basic groups, and the set of edges {(1,2), (2,3), (3,4), (4,5)} represent the bonds between the groups.

Table 2.1: Simple and valence atomic connectivity index values for basic groups used in this work

Group	$\boldsymbol{\delta}$	$\boldsymbol{\delta^v}$	**Group**	$\boldsymbol{\delta}$	$\boldsymbol{\delta^v}$
$-CH_3$	1	1	>C=	3	4
-N<	3	5	=CH-	2	3
$-NH_2$	1	3	>CH-	3	3
=O	1	6	$>CH_2$	2	2
-OH	1	5	-F	1	7
-O-	2	6	>C<	4	4

CH_3-CH_2-CH_2-CH_2-OH

1 2 3 4 5

Figure 2.1: Conventional representation of n-butanol

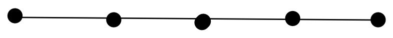


Figure 2.2: Molecular graph of n-butanol

2.1.2 Calculation of connectivity indices

A partitioned adjacency matrix (PAM) is used to store the molecular graph numerically for each molecule. A PAM is a symmetric square matrix that has order equal to the maximum number of possible vertices (basic groups) in the molecular graph. Each row and column represents a basic group. If we denote the row number by 'i' and column number by 'j', elements in the PAM can be denoted by the binary variable y_{ij}. If there is a bond between group 'i' and group 'j', then $y_{ij} = 1$, otherwise $y_{ij} = 0$. The calculation of molecular connectivity indices involves two more binary variables, w_i and y_{ijk}. w is a one-dimensional array containing binary variables, each of which represents the existence of a particular group in the molecule. y_{ijk} is an element in the three-dimensional matrix that has value one, if a triplet exists between i, j and k i.e. if $y_{ij} = 1$ and $y_{jk} = 1$.

The molecular connectivity indices of zeroth, first and second order are considered in this work. Each of these connectivity indices exist in two types: simple and valence. For calculating the simple and valence molecular connectivity indices, simple and valence atomic connectivity indices are used, respectively. Simple atomic connectivity indices (δ values) refer to the number of bonds which can be formed by a group with other groups. The δ^v values are atomic valence connectivity indices, which describe the electronic structure of each basic group. δ^v is calculated using the formula shown below in Equation 2.1, where Z^v denotes the number of valence electrons of a non-hydrogen atom, N_H denotes the number of hydrogen atoms bonded to it, and Z is the atomic number of the non-hydrogen atom in a basic group.

$$\delta^v = \frac{Z^v - N_H}{Z - Z^v - 1} \tag{2.1}$$

The connectivity indices are structural descriptors that contain significant information about the molecule, such as the number of hydrogen and non-hydrogen atoms bonded to each non-hydrogen atom, the electronic structure details of each atom, and features like branching. Therefore, even if two molecules have the same groups present in their structures but are connected differently, they will have different molecular connectivity indices. The zeroth order molecular connectivity indices, χ^0 and ${}^v\chi^0$ are sums over each basic group, hence they represent the identity of each basic group present in the molecular graph. The first order molecular connectivity indices, χ^1 and ${}^v\chi^1$ are sums over bond pairs present in the molecule and thus consider the connectivity in the molecule. The second order molecular connectivity indices, χ^2 and ${}^v\chi^2$ are sums over the triplets (three adjacently connected groups) present in the molecule and thus represent even better connectivity. The equations for calculating zeroth, first, and second order molecular connectivity indices based on the atomic connectivity indices are given in Equations 2.2-2.7. The square root sign was employed by Randic′ in the equations to avoid overlapping of indices belonging to different set of isomers (Randic′, 1975). It should be noted that stereoisomers cannot be distinguished by connectivity indices, as connectivity within the molecules remains same.

$$\chi^0 = \sum_{Vertices} \frac{w_i}{\sqrt{\delta_i}} \tag{2.2}$$

$$ {}^v\chi^0 = \sum_{Vertices} \frac{w_i}{\sqrt{\delta_i^v}} \tag{2.3}$$

$$\chi^1 = \sum_{Edges} \frac{y_{ij}}{\sqrt{\delta_i \delta_j}} \tag{2.4}$$

$$ {}^v\chi^1 = \sum_{Edges} \frac{y_{ij}}{\sqrt{\delta_i^v \delta_j^v}} \tag{2.5}$$

$$\chi^2 = \sum_{Triplets} \frac{y_{ijk}}{\sqrt{\delta_i \delta_j \delta_k}} \tag{2.6}$$

$$ {}^v\chi^2 = \sum_{Triplets} \frac{y_{ijk}}{\sqrt{\delta_i^v \delta_j^v \delta_k^v}} \tag{2.7}$$

The calculation of connectivity indices for n-butanol has been performed using the atomic connectivity indices from Table 2.1. n-butanol has five basic groups which are shown as five vertices, and the four connections between the five groups are shown by four edges in the molecular graph. The calculation for the six connectivity indices, made using Equations 2.2-2.7, is shown along with the final numerical values.

$$\chi^0 = \frac{1}{\sqrt{1}}+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{1}} = 4.1213$$

$$^{v}\chi^0 = \frac{1}{\sqrt{1}}+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{5}} = 3.5685$$

$$\chi^1 = \frac{1}{\sqrt{1\times2}}+\frac{1}{\sqrt{2\times2}}+\frac{1}{\sqrt{2\times2}}+\frac{1}{\sqrt{2\times1}} = 2.4142$$

$$^{v}\chi^1 = \frac{1}{\sqrt{1\times2}}+\frac{1}{\sqrt{2\times2}}+\frac{1}{\sqrt{2\times2}}+\frac{1}{\sqrt{2\times5}} = 2.0233$$

$$\chi^2 = \frac{1}{\sqrt{1\times2\times2}}+\frac{1}{\sqrt{2\times2\times2}}+\frac{1}{\sqrt{2\times2\times1}} = 1.4082$$

$$^{v}\chi^2 = \frac{1}{\sqrt{1\times2\times2}}+\frac{1}{\sqrt{2\times2\times2}}+\frac{1}{\sqrt{2\times2\times5}} = 1.1319$$

Thus, it can be seen how the molecular connectivity indices are used to describe connectivity of a molecule numerically. A quantitative structure property relationship (QSPR) derived using these indices is discussed in Chapter Three.

2.2 MoRSE descriptors

Connectivity indices work well for predicting the physicochemical properties of compounds (Kier, 2012). Many researchers have shown that in some cases, the 2D descriptors, even though they are based on topology of the molecule, possess some information about 3D structure (Petitjean, 1992, Estrada *et al*., 2001). However, Bath *et al.* (1995) stated that 2D descriptors are not enough for the proper representation of a molecule, hence 3D descriptors are necessary. Any molecule is a 3-dimensional entity and possesses a certain special arrangement of atoms. While estimating the properties for any structure, this fact should be taken into account, especially for biological properties which are closely dependent on the interaction of molecule with the biological target. 3D MoRSE descriptors have been used in the past to predict biological properties successfully (Schuur *et al.*, 1996). Hence, they are used in this work for the design of antibiotic molecules.

2.2.1 Background on 3D MoRSE descriptors

3D MoRSE stands for Molecular Representation of Structure based on Electron diffraction. These descriptors contain information about the 3D minimum energy structure of a molecule. In the 90's, 3D structure generation of a molecule became very easy with the advent of 3D structure generating programs like CORINA and CONCORD. The main question was how to represent the 3D structure? An easy answer to the problem would be to represent a molecule with atomic cartesian coordinates, where each atom would have a x, y and z coordinate. The number of parameters needed to store a molecule will then depend on the number of atoms present in a molecule. Ethane, which contains 8 atoms, will have 24 and caffeine, with 24 atoms, will have 72. Many methods used to find correlations between structures and property values, such as statistical or pattern recognition methods, need an equal number of variables for each molecule in the dataset.

The solution to this problem is to represent every molecule with the same number of variables despite the number of atoms present in it, which was achieved using the 3D MoRSE descriptors.

The experimental methods used for determining 3D structure were considered when a unique way to represent every molecule was to be decided. Electron diffraction was one of the methods used in the structural determination. The general molecular transform used in electron diffraction studies is given by Equation 2.8.

$$G(\vec{S}) = \sum_{i=1}^{N} f_i(2\pi\vec{r_i}\vec{S}) \tag{2.8}$$

where $G(\vec{S})$ is diffraction pattern, $\vec{S}$ is an observation point, $\vec{r_i}$ is the location of each atom, f_i is the form factor of atom *i* and *N* is number of atoms in the molecule.

Equation 2.8 was presented in a more useful form by Weirl which is presented below as Equation 2.9 (Weirl, 1931).

$$I(s) = K\sum_{i=2}^{N}\sum_{j=1}^{i-1} f_i f_j \int_0^{\infty} P_{ij}(r)\frac{\sin(sr)}{sr}dr \tag{2.9}$$

Where,

$$s = 4\pi\sin\left(\frac{\vartheta}{2}\right)/\lambda$$

ϑ is scattering angle, λ is wavelength, *I*(s) is intensity of scattered radiation, $P_{ij}(r)$ is probability distribution of the variation in the distance between atoms *i* and *j* due to vibrations, f_i is form factor of atom *i* and *K* is collection of instrument constants.

Gasteiger *et al.* (1996) further modified Equation 2.9 by considering the suggestions made by Saltzberg and Wilkins (1977).

$$I(s) = \sum_{i=2}^{N} \sum_{j=1}^{i-1} A_i A_j \frac{\sin(sr_{ij})}{sr_{ij}} \tag{2.10}$$

where *I(s)* is the scattered electron intensity, *A* can be any atomic property such as atomic number or atomic mass, r_{ij} are the atomic distances between *i* th and *j* th atoms, *s* represents scattering of electrons by the group of *N* atoms at point r_i and *N* is the number of atoms.

The value of parameter ‘s’ is considered at specific points within a certain range. The value of ‘s’ usually varies from 0 – 31 Å^{-1} with an increment of 1 unit (Gasteiger *et al*., 1996).

2.2.2 Calculation of 3D MoRSE descriptors

The calculation of 3D MoRSE descriptors is usually done by selection of weights. There are 2 main categories among which the 3D MoRSE descriptors are divided: unweighted descriptors and weighted descriptors. The unweighted descriptors have no weight, i.e. the values for the atomic property parameters in Equation 2.10, A_i and A_j are 1. On the other hand, the weighted descriptors are calculated by assigning some atomic property values to parameters A_i and A_j. Hence, 3D MoRSE descriptors provide a great advantage of selecting the atomic property based on the need of problem at hand. In commonly used software packages like DRAGON (Mauri *et al.*, 2006), E-dragon weighted descriptors are calculated for four different atomic properties: atomic mass, atomic van der Waals volume, atomic Sanderson electronegativity and atomic polarizability. It is worth noting that the numbering of 3D-MoRSE descriptors starts from 1, e.g. Mor01u represents an unweighted MoRSE descriptor having the scattering parameter, ‘s’ equal to 0 Å^{-1} as the scattering parameter starts from zero, Mor02u denotes unweighted MoRSE descriptor with ‘s’ equal to 1 Å^{-1} and so on.

Let us consider an example of methane molecule to better understand the calculation of MoRSE descriptors. The simplest MoRSE descriptor that can be considered is Mor01u, which is calculated using Equation 2.11.

$$I(0) = \sum_{i=2}^{N}\sum_{j=1}^{i-1} A_i A_j \frac{\sin(0 \times r_{ij})}{0 \times r_{ij}} = \sum_{i=2}^{N}\sum_{j=1}^{i-1} A_i A_j \frac{\sin(0)}{0} \tag{2.11}$$

Even though division by zero is undefined, the limit of the term, sin(θ)/θ as θ approaches zero equals 1. Hence, the term sin (0)/0 becomes 1. As the descriptors are unweighted, all the A_i and A_j values are equal to 1. So, the summation term results in a number, giving all the atomic pair combinations in a molecule containing 'N' atoms. It can be observed that this descriptor is a function of the number of atoms only. 3D-MoRSE descriptors with scattering parameters equal to 0 $Å^{-1}$ are always positive for all positive weights like atomic mass and van der Waals atomic volume. Mor01u for methane, which consists of 5 atoms, is equal to ${}^{5}C_{2}$ = 10.

The second unweighted descriptor, Mor02u, has the scattering parameter value equal to one. Please note that A_i = A_j = 1 for unweighted descriptors. The calculation is given in Equation 2.12.

$$I(1) = \sum_{i=2}^{N}\sum_{j=1}^{i-1} 1 \times 1 \times \frac{\sin(1 \times r_{ij})}{1 \times r_{ij}} = = \sum_{i=2}^{N}\sum_{j=1}^{i-1} \frac{\sin(r_{ij})}{r_{ij}} \tag{2.12}$$

As the scattering parameter is no longer 0, we need to know the distance between any two atoms within a molecule to calculate further MoRSE descriptors. As the methane molecule is highly symmetric and has repeating interatomic distance values, we can write the expression for Mor02u as shown in Equation 2.13.

$$Mor02u = 4\frac{\sin(1.085)}{1.085} + 6\frac{\sin(1.772)}{1.772} = 6.578 \tag{2.13}$$

For calculating each further unweighted MoRSE descriptor value, the scattering parameter is increased by 1. The calculation of weighted MoRSE descriptors is done in the same way as unweighted MoRSE descriptors except the A_i and A_j values are not 1, but some atomic property values (Devinyak, 2014).

Thus, 3D MoRSE descriptors provide a unique way to represent 3D structure of a molecule, a definite improvement over connectivity indices which only consider the connections within a molecule. 3D MoRSE descriptors inherently account for the bond angles and bond lengths present within atoms of a molecule, thus possessing better characteristics of the real molecular structure. They can even help in avoiding ambiguity for certain cases. For example, let us consider two molecules, 3-methyl heptane and 4-methyl heptane. They both have same values for zeroth and first order connectivity indices. Hence, they contribute the same information despite being two different molecules. The values of 3D MoRSE descriptors in this case would be different, as the change of methyl position from 3 to 4 would change the interatomic distance for atoms in the methyl group with the rest of the atoms in the molecule.

Overall, connectivity indices and 3D MoRSE descriptors are two sets of promising molecular descriptors for predicting physicochemical and biological properties of drug molecules. These molecular descriptors are correlated with property values of a group of compounds for obtaining quantitative structure-property relationships. Their development process and statistical analysis will be discussed in the following chapter.

Chapter 3

Development of Quantitative Structure-Property Relationships (QSPRs)

Quantitative structure-property relationships play a very important role in CAMD. QSPRs are equations that relate property values to molecular descriptors. Therefore, if we know the structure of a molecule belonging to a particular class of compounds, its property value can be predicted using a QSPR. The main advantage of QSPRs is that once they are verified with a certain accuracy and prediction power, they can be used for predicting properties of new molecules even before they are synthesized. Hence, they are employed in this work for property prediction of newly designed molecules. The formulation of a molecular design problem containing these QSPRs is discussed in Chapter Four.

QSPRs are developed using experimental property data for a set of similar compounds. They are used to find out how the structure of a molecule should be changed in order to get a desired property value or to predict properties of compounds when experimental measures are unavailable. They have applications in different areas like combinatorial chemistry, molecular design and mainly, screening of large molecular databases. Day by day, as the need for new medicines is growing, the need to find potential drug candidates without doing extensive experimental work is on the rise. QSPRs help eliminate a majority of possible compounds, and narrow down a full database to a few promising candidates, which are considered for further studies. Thus, they play a crucial role in the drug discovery process.

The QSPRs developed in this work are for the minimum inhibitory concentration (MIC) of certain antibiotics and for the octanol/water partition coefficient i.e. Log *P* values. Andrews (2001) has defined MIC as the lowest concentration of an antimicrobial that will inhibit the visible growth of a microorganism after overnight incubation. MIC is considered the ‘gold standard’ to test susceptibility of bacteria to antibiotics, and hence it was chosen in this work for the antibiotic design (Andrews 2001). Another chosen property is the Log *P* value, which is logarithm of the octanol/water partition coefficient of the compound. A drug molecule has to pass through numerous biological barriers before it reaches the target site, which involves passing through different lipophilic and lipophobic membranes. An octanol/water system mimics biological tissues, and therefore gives a good estimate as to how effectively a drug would pass through these barriers. The two properties provide two important characteristics for antibiotic design. The new molecule should not only be potent enough to exhibit antibacterial activity, but also possess enough affinity towards both lipophilic and lipophobic environments in order to exhibit a biological effect. Other properties like density, solubility and toxicity are also relevant for drug design and can be included in the molecular design formulation depending on the scope of application.

Quinolones represent a class of broad-spectrum antibiotics that is effective against a wide range of bacteria. The development of quinolones began in 1960s with the discovery of ‘nalidixic acid’. Since then, this class of antibiotics has grown to become one of the most prescribed antibiotics in the world. One of the important changes made to the quinolone structure during the development process was inclusion of a fluorine atom at the C6 position. Because of this addition, quinolones are often referred as fluoroquinolones. Fluoroquinolones are used for treating infections caused by various gram negative and positive bacteria, *staphylococcus aureus* being one of them (Aldred *et al.*, 2014).

3.1 Development of QSPRs using connectivity indices

QSPRs are developed for correlating MIC and Log *P* values with the connectivity indices of fluoroquinolones listed in Tables 3.1 and 3.2 respectively. They are developed according to the flowchart shown in Figure 3.1. The development process for the QSPRs is described below in detail.

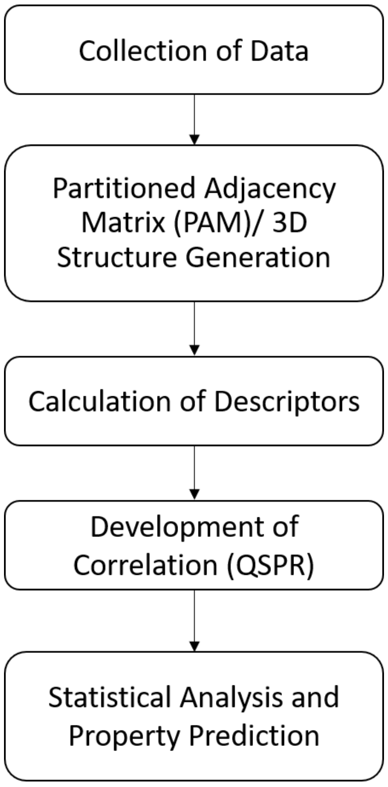

Collection of Data
Partitioned Adjacency Matrix (PAM)/ 3D Structure Generation
Calculation of Descriptors
Development of Correlation (QSPR)
Statistical Analysis and Property Prediction

Figure 3.1: Flowchart for QSPR development.

3.1.1 Step one: Data collection

The chemical structures and their corresponding property values are taken from the literature. Andrews has reported the MIC values of different classes of antibiotics including fluoroquinolones against various bacteria. The data was collected for the MIC values from this paper for fluoroquinolones tested against *staphylococcus aureus* (Andrews, 2001). The MIC values are reported in mg/L. They are converted into mol/L before correlating the data. As the MIC values in mol/L are of the order 10^{-6} to 10^{-8}, negative logarithm of all the MIC values is considered. The

MIC values are shown in Table 3.1. The experimental values for Log *P* were also obtained from literature (Pérez *et al.*, 2002, Bermejo *et al.*, 1999) and are given in Table 3.2.

Table 3.1: MIC values used as data (Andrews, 2001)

Fluoroquinolone	**MIC (mg/L)**
Ciprofloxacin	0.12
Enoxacin	0.5
Fleroxacin	0.5
Gatifloxacin	0.03
Grepafloxacin	0.03
Levofloxacin	0.12
Lomefloxacin	0.5
Moxifloxacin	0.06
Nalidixic Acid	130
Norfloxacin	0.25
Ofloxacin	0.25
Pefloxacin	0.25
Rufloxacin	1
Sparfloxacin	0.03
Trovafloxacin	0.015

Table 3.2: Log *P* values used as data (Pérez *et al*., 2002, Bermejo *et al*., 1999)

Fluoroquinolone	**Log *P***
Norfloxacin	-1.52
Pefloxacin	0.26
N-Ethylnorfloxacin	0.37
N-Propylnorfloxacin	1.05
N-Butylnorfloxacin	1.48
N-Pentylnorfloxacin	2.11
N-Hexylnorfloxacin	2.71
N-Heptylnorfloxacin	3.22
Ciprofloxacin	-1.1
N-Methylciprofloxacin	0.15
Enrofloxacin	0.53
N-Propylciprofloxacin	1.07
N-Butylciprofloxacin	1.55
N-Pentylciprofloxacin	2.06
N-hexylciprofloxacin	2.56
N-Heptylciprofloxacin	3.02

3.1.2 Step two: Creation of data structure for storing bonds

As we are trying to relate the structure of a chemical with property values, which are numerical in nature, it is necessary to find a way to present the chemical structure in numerical way as well. For the mathematical interpretation of chemical structure, a data structure is created to store all the bonds between groups of a molecule. One example of such data structure is Partitioned Adjacency Matrix, which has been discussed in Chapter Two.

3.1.3 Step three: Calculation of connectivity indices

The calculation of connectivity indices is performed using Equations 2.2 to 2.7 from Chapter Two. In this work, the online software E-Dragon calculated connectivity indices for all molecules (Tetko *et al*., 2005). The process of forming a PAM separately for each molecule in the data set is tedious. Hence, we used the software E-Dragon to calculate the connectivity indices. The input was .sdf (structure-data file) file of the compound for which connectivity indices are to be calculated and the output was the numerical values of the connectivity indices.

3.1.4 Step four: Developing the QSPRs

The QSPRs are developed as correlations between property values and zeroth, first and second order connectivity indices. As we are considering only zeroth, first and second order connectivity indices in this work, the QSPR will have the form shown in Equation 3.1, where P is the property value, C_1, C_2, C_3, C_4, C_5, C_6 are the coefficients, C_7 is the constant, χ^n is nth ordered simple connectivity index and ${}^v\chi^n$ is nth ordered valence connectivity index.

$$P = C_1\chi^0 + C_2{}^v\chi^0 + C_3\chi^1 + C_4{}^v\chi^1 + C_5\chi^2 + C_6{}^v\chi^2 + C_7 \tag{3.1}$$

R is a statistical software and program which is widely used for data analysis and statistical computing (R Core Team, 2017). The method of linear regression is used to obtain the QSPRs in R. The correlations for MIC and LD-50 values, developed using the connectivity indices, are shown in Table 3.3.

Table 3.3: QSPRs using connectivity indices

Property	**Correlation**	**R²**	**Q²**
MIC	$-\log(MIC) = -0.494\chi^0 - 1.788\chi^1 + 2.687\chi^2 + 0.338\,{}^v\chi^0 + 3.51\,{}^v\chi^1 - 3.647\,{}^v\chi^2 - 3.935$	0.87	0.54
Log P	$\log(P) = 29.499\chi^0 - 28.163\chi^1 + 6.995\chi^2 - 23.903\,{}^v\chi^0 + 18.48\,{}^v\chi^1 - 2.991\,{}^v\chi^2 - 66.943$	0.99	0.97

3.1.5 Step five: Statistical analysis and property prediction

After developing a mathematical model that correlates property values to molecular descriptors, a statistical analysis is performed to check quality of fit and predictive ability of the model. R^2 is the correlation coefficient, which denotes how well the data fits the mathematical model. For a model M, generated for data set that consists of variables $x_1, x_2, \ldots, x_n$ which describe a value y, the correlation coefficient R^2 is calculated as shown in Equation 3.2 (Draper, 1966).

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(y_i^{fit} - y_i\right)^2}{\sum_{i=1}^{N}(y_i - y_{mean})^2} = 1 - \frac{RSS}{SS}$$

(3.2)

Where, N denotes total data points, y_i^{fit} is the y value predicted by model M, y_i is given data point, y_{mean} is the mean of all y observations, ***RSS*** is **R**esidual **S**um of **S**quares and ***SS*** is **S**um of **S**quares.

Q^2 is the prediction quotient that shows how well the model predicts the correlated property value. Leave-One-Out Cross-Validation (LOOCV) method was used to evaluate the predictive ability of the model. In LOOCV method, one observation is left out of the data set and the remaining data is used to correlate the descriptors to property values. The resulting model is used for predicting property value for the left-out observation. This process is repeated for each observation in the data set. Upon completion, the predictions are used to calculate the predicted residual sum of the square errors (*PRESS*). The calculation of Q^2 is shown in Equation 3.3 (Efron, 1983).

$$Q^2 = 1 - \frac{\sum_{i=1}^{N}\left(y_i^{pred} - y_i\right)^2}{\sum_{i=1}^{N}(y_i - y_{mean})^2} = 1 - \frac{PRESS}{SS}$$

(3.3)

Where N denotes total data points, y_i^{pred} is predicted value for the left-out data point, y_i is given data point, y_{mean} is the mean of all y observations, ***PRESS*** is **PRE**dicted residual **S**um of **S**quares and ***SS*** is **S**um of **S**quares.

A special package named 'cvq2' is installed in R for calculating Q^2. One of the advantages of this package is that it generates a linear model for the given data while calculating the Q^2 value and provides the conventional correlation coefficient R^2 along with Q^2. The R^2 and Q^2 values for MIC and LD-50 models generated with connectivity indices are given in Table 3.3.

For both parameters, R^2 and Q^2, values closer to 1 imply good data fit and predictability for the model. For the MIC model, the R^2 value is 0.87, indicating moderately good quality of fit for the given data. However, the Q^2 value is 0.54, indicating a low quality of prediction for the correlation. For the Log *P* model, both R^2 and Q^2 values are greater than 0.95, indicating an excellent data fit and predictability. This behavior can be explained with the nature of molecular descriptors used. Connectivity indices consider the molecular connectivity information which has been proven to be very good for predicting physicochemical properties, however not as good for predicting the biological activity which involves the shape, geometry and orientation of molecule with respect to the binding target.

3.2 QSPRs for predicting MIC and Log *P* using 3D MoRSE descriptors

The process used for development of MIC and Log *P* QSPRs using 3D MoRSE descriptors is described below.

3.2.1 Step one: Data collection

The MIC and Log *P* data is same as the data used for generating QSPRs with connectivity indices.

3.2.2 Step two: 3D structure generation

3D MoRSE descriptors depend on the interatomic distances within a molecule. Hence, we need to find spatial coordinates of atoms to get the interatomic distances. With the availability of programs like CORINA (Sadowski *et al*., 1994) and Openbabel (O′Boyle *et al*., 2011), this can be achieved very easily. In Openbabel, the input can be supplied in the SMILES (Weininger, 1987) notation format to find the 3D coordinates of atoms. Once we know the positions of all atoms, the distance formula can be used to get distance between any two atoms.

3.2.3 Step three: Calculation of 3D MoRSE descriptors

3D MoRSE descriptors are calculated using Equation 2.10 in Chapter Two. We have used the software E-Dragon for the calculation of 3D MoRSE descriptors (Tetko *et al*., 2005). The input is provided as .sdf file of the molecule and the output is numerical values of desired descriptors.

3.2.4 Step four: Developing the QSPRs

In this work ten unweighted MoRSE descriptors were considered for correlation development. The simplest MoRSE descriptors are the unweighted MoRSE descriptors. The QSPRs were developed as a correlation between the property values and unweighted MoRSE descriptors. An experiment was conducted to determine the number of descriptors that should be included in the correlations for MIC and Log *P* values to avoid overfitting of the data. Different models were created for different numbers of descriptors used in the correlation in R by using method of linear regression. We started with the first five unweighted MoRSE descriptors and continued until ten. The R^2 and Q^2 values calculated for MIC and Log *P* models are listed in Tables 3.4 and 3.5 respectively.

Table 3.4: Selection of number of unweighted MoRSE descriptors for MIC model

Number of MoRSE descriptors used	**R^2**	**Q^2**
First 5	0.8058	-0.045
First 6	0.8577	0.2150
First 7	0.9640	0.7336
First 8	0.9869	0.9209
First 9	0.9894	0.9216
First 10	0.99	0.69

Table 3.5: Selection of number of unweighted MoRSE descriptors for Log *P* model

Number of MoRSE descriptors used	**R^2**	**Q^2**
First 5	0.9853	0.9667
First 6	0.9898	0.9656
First 7	0.9935	0.97
First 8	0.9963	0.9843
First 9	0.9983	0.985
First 10	0.9984	0.9806

3.2.5 Step five: Statistical analysis and property prediction

Statistical information is used to make the decision about the number of descriptors to be included in the correlations. For the MIC model, the R^2 values show an increase from 0.81 to 0.99 indicating better models for data fitting with increase in number of descriptors. The Q^2 values show a vast difference when changing from five to ten descriptors, showing an increase until nine descriptors and a sudden drop at ten. This results from the overfitting of data at ten descriptors. So, overall the MIC model with nine descriptors is best, showing very good data fitting and predictive capability. For Log *P* models, the data fit is very good for all, as R^2 does not change very much with an increase in the number of descriptors. The predictive capability also varies within a very small range indicating the highest value for nine descriptors. Hence, the Log *P* model with nine descriptors is chosen to be the best after considering the R^2 and Q^2 values. The resulting correlations for MIC and LD-50 values along with their statistics are shown in Table 3.6.

Both the models show very good quality of fit and predictive capability for the given data. The predictive capability is better for Log *P* than MIC. MIC is dependent on the biological activity, thus introducing an element of uncertainty in the predictions. However, it is no surprise that the predictability for Log *P* is excellent as the information contributed by 3D descriptors is more than enough for predicting a physical property.

Table 3.6: QSPRs using 3D MoRSE descriptors

Property	**Correlation**	**R^2**	**Q^2**
MIC	$-\log(MIC) = -0.002 * Mor01u + 0.09 * Mor02u - 0.082 * Mor03u - 0.127 * Mor04u - 0.634 * Mor05u - 0.413 * Mor06u + 0.19 * Mor07u - 0.581 * Mor08u - 0.237 * Mor09u - 2.509$	0.99	0.92
Log P	$\log(P) = 0.022 * Mor01u + 0.011 * Mor02u + 0.624 * Mor03u - 0.41 * Mor04u + 2.312 * Mor05u - 2.227 * Mor06u + 0.497 * Mor07u - 0.465 * Mor08u + 0.589 * Mor09u - 11.375$	0.99	0.99

The goal of this work is to design an antibiotic molecule having certain properties. In the molecular design process, these property values are calculated by QSPRs. Hence, it is very important that the appropriate QSPR model is chosen for property prediction. The QSPR model selection for molecular design is dependent on the kind of molecular descriptors used for model development. For example, let us consider MIC models developed in this chapter. Both the models are developed using the same data. However, the statistics show that the MIC model developed using 3D descriptors is much better and reliable compared to the one with connectivity indices. This difference arises due to different nature of molecular descriptors used. Connectivity indices are based on the concept of a molecular graph whereas, 3D MoRSE descriptors are based on interatomic geometric distances. As 3D QSARs work well for correlating the activity values, 3D

MoRSE descriptors provide a better model for predicting MIC. For Log *P*, both the molecular descriptors show very good results, however slightly better predictive capability is observed with the 3D descriptors.

Based on these results, MIC and Log *P* models derived using 3D MoRSE descriptors are chosen for the molecular design. Using these property models, an optimization problem has been formulated. It is discussed in the next Chapter.

Chapter 4

Formulation of the Design Problem

The QSPRs developed in the previous chapter provide a means of predicting property values of fluoroquinolones. These QSPRs along with other equations are put into an optimization format to create a molecular design problem. Previously, optimization has been used in the design of various chemicals and products. For example, Eslick *et al.* (2009) designed polymethacrylate dental materials, Achenie and Sinha (2003) designed cleaning solvents in the printing industry, Sinha *et al.* (1999) designed environmentally benign solvents and, Gebreslassie and Diwekar (2015) designed a solvent for extracting acetic acid from waste process stream by using optimization methods. In this chapter, an antibiotic design problem is formulated using the optimization framework.

Molecular design has evolved over the years. In the beginning, group contribution methods were mainly used for developing QSPRs and ultimately for molecular design. Group contribution methods consider the presence of certain blocks or groups of atoms present in a molecule and thus account for their contribution towards property values (Gani *et al.,* 1991). Even though group contribution methods were successful at predicting certain properties, a better approach for representing molecular structure was provided by topological indices which are based on the topology of a molecule (Bicerano, 2002). An example is connectivity indices, which are used in this work for developing correlations. Connectivity indices not only consider the presence of certain groups in a molecule, but also the way they are connected, thus they provide better information than group contribution methods. An even better approach for representing molecules is to use their 3D structure, as molecules are three dimensional entities and molecular descriptors contributing 3D structural information can be considered as the ones that come closest to representing the actual molecular structure. An example is 3D MoRSE descriptors, which are used in this work for property prediction and in molecular design (Gasteiger *et al.*, 1996).

When the QSPRs described above are combined with constraints to ensure a stable, bonded molecule, an optimization problem is derived. This problem consists of an objective and a set of constraints. The goal of an optimization problem is to find a solution that either minimizes or maximizes a given objective while staying in the feasibility region created by a set of constraints. An increase in the number of constraints makes the feasibility region smaller, and often reduces the computational time to find a solution. Thus, the number of constraints is an important factor for solving an optimization problem in a time-efficient manner. The constraints used in this work include structural feasibility and connectivity constraints, which are typically used in molecular design. More details about these constraints and objective are given below.

4.1 Objective

This work is an example of a small drug molecule design, with the molecule possessing desired physical and biological property values. The user can select a drug class, and the properties that are relevant to the application. As this work deals with antibiotic design, the chosen drug class is fluoroquinolones and the chosen properties are MIC and Log P for reasons mentioned in Chapter Three. The calculation of these properties is done using the QSPRs developed in the previous chapter. The desired property values are called target property values. The objective function for the optimization problem is stated in Equation 4.1, where $P_{i,target}$ is the target value of property i, $P_{i,predicted}$ is the predicted value of property i using QSPRs and w_i is scaling factor used for adjusting relative importance of each property.

$$Min \quad S = \sum_{i \in properties} w_i \left| \frac{P_{i,target} - P_{i,predicted}}{P_{i,target}} \right| \tag{4.1}$$

The scaling factor gives the flexibility to design a molecule by giving more weight to one property over others. As the predicted property values approach target values, the objective function approaches zero. Hence, the objective function needs to be minimized. Sometimes, the objective function is written in linear or convex form to simplify the method for finding solution, as nonlinear or non-convex problems are comparatively difficult to solve. Once the objective is converted to a linear or convex form, the problem can be solved with a deterministic method to find the global optimum. This is not required in this work, since a stochastic algorithm like Artificial Bee Colony can find near-optimal solutions to nonlinear, non-convex problems.

4.2 Constraints

The structural feasibility and connectivity constraints are important for molecular design. The structural constraints ensure that a stable molecule is formed. This means the valency of every group is satisfied, only one type of bond exists between any two groups, and the total number of basic groups present in a molecule is kept constant. The connectivity constraints are known as network flow constraints. They ensure that all the basic groups which form a molecule are bonded together as a single unit. The network flow constraints are based on following considerations:

1) The groups present in a molecule can be considered nodes of a molecular graph. One existing node is chosen as the source and is provided N units. N units are equal to the total number of groups present in the molecule.
2) These N units must be distributed among all other nodes such that every other node retains exactly one unit.

3) Units can only flow between nodes connected by edges.

These rules are satisfied with a connected graph. These constraints, if included in a molecular design problem, end up producing feasible molecular structures that are connected as a complete unit. Even though the structure feasibility and connectivity constraints are not explicitly present in the optimization problem when we solve using a stochastic algorithm, they are accounted for by replacing a group with another group having the same valency. Thus, newly formed molecules always have stability and a well-connected structure, as no infeasible solutions are considered.

The QSPRs used for predicting properties are included as constraints. The calculation of property values for a candidate molecule is done by using molecular descriptors. Hence, the equations for calculating 3D MoRSE descriptors are included as constraints. As this work deals with antibiotic design, it is very important that the pharmacophore should remain intact. The pharmacophore is the part of a drug molecule that is responsible for the desired biological effect. So, another constraint was implicitly added, which did not allow the pharmacophore structure to change.

4.3 Optimization problem type and size

Optimization problems are broadly categorized into two groups based on the nature of equations in a problem. A problem having linear equations is called a linear problem. On the other hand, a problem containing any nonlinear equations is a nonlinear problem. These two types of problems can be further divided into two types based on the nature of variables used in the problem. If all the variables in the problem are continuous, the problem is simple linear program (LP) or simple nonlinear program (NLP). If the problem contains both continuous and discrete variables, the problem is mixed-integer linear program (MILP) or mixed-integer nonlinear program (MINLP). Thus, there are four general types of optimization problems.

The solution to an LP or MILP can be found easily compared a nonlinear problem. For a linear problem, the constraints are linear in nature. Hence, the feasibility region created by constraints is convex and the global optimum lies at one of the extremities of feasibility region created by constraints. A feasible region is convex if, for any two points present within the region, the line segment joining those two points lies entirely within the feasibility region (Morris and Stark, 2015). Some examples of convex region are shown in Figure 4.1. For a nonlinear problem, the feasibility region could be convex or non-convex. MINLPs occur in many areas such as plant designs, scheduling, process synthesis and molecular design. The MINLP in this work is non-convex in nature. Some examples of non-convex region are shown in Figure 4.2.

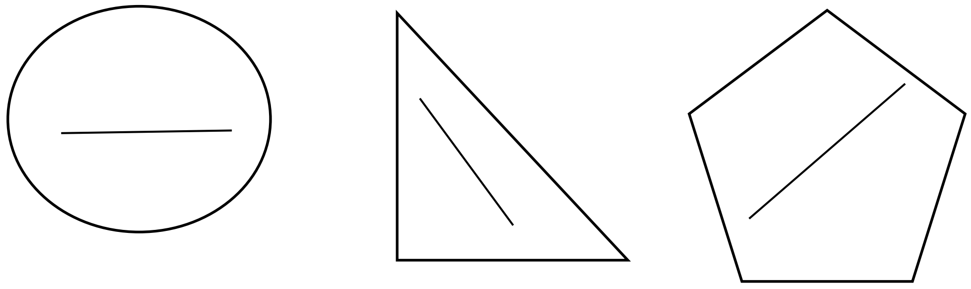


Figure 4.1: Examples of convex regions

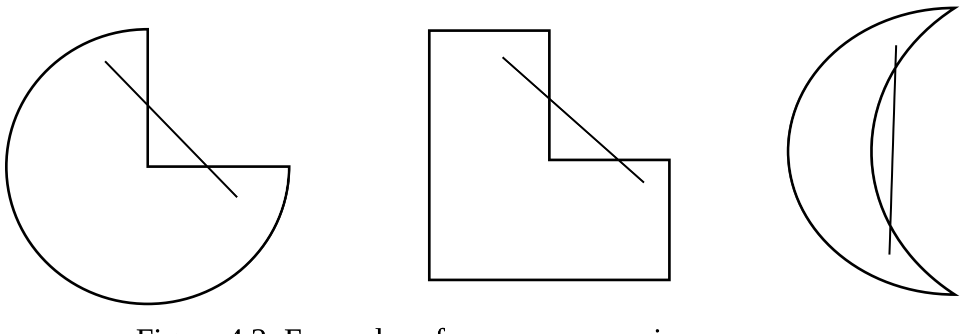


Figure 4.2: Examples of non-convex regions

The optimization problem formulated in this chapter is nonlinear, and it contains both continuous and discrete variables. The nonlinearity arises due to the MoRSE descriptor terms in the QSPR equations. Even though the QSPRs are linear in nature, the calculation of MoRSE descriptors, shown in Equation 4.2, adds the element of nonlinearity to the problem. The problem contains both continuous and discrete variables. For example, the MIC and Log *P* property values are continuous variables, while on the other hand, the existence of a particular group in a molecule is a binary variable. Thus, the formulation of design problem results in an MINLP.

$$I(s) = \sum_{i=2}^{N}\sum_{j=1}^{i-1} A_i A_j \frac{\sin(sr_{ij})}{sr_{ij}} \tag{4.2}$$

It is a good idea to determine the size of an optimization problem to understand how large the search space is. Usually one can calculate the total number of feasible solutions for a given optimization problem and have a fair idea about the scope of the problem. For a molecular design problem, the number of feasible solutions is going to be the total number of structures that can be designed by using groups that are included in the design. The groups used in this work are listed in Table 4.1 and Table 4.2.

The number of feasible solutions to the molecular design are calculated as follows:

Total number of groups having valency one = 10

Total number of groups having valency two = 2

Number of positions where replacement can occur with valency one groups = 4

Number of positions where replacement can occur with valency two groups = 4

Total number of possible structures = 10×10×10×10×2×2×2×2 =160,000

Thus, the total number of feasible solutions/ structures are 160,000. The goal is to find a structure which will have MIC and Log *P* values as close as possible to the target values.

Table 4.1: Groups having valency one used in the molecular design

No.	Groups with valency one
1	-F
2	$-NH_2$
3	$-CH_3$
4	-OH
5	$-CH_2-OH$
6	$-CH_2-NH_2$
7	$-CH_2-CH_2-OH$
8	$-CH_2-CH_2-NH_2$
9	$-CH_2-CH_2-CH_2-OH$
10	$-CH_2-CH_2-CH_2-NH_2$

Table 4.2: Groups having valency two used in the molecular design

No.	Groups with valency two
1	-O-
2	$-CH_2-$

4.4 Deterministic vs stochastic methods

Solutions to any optimization problem can be sought by implementing a deterministic method, or a stochastic method. Deterministic methods guarantee the determination of a global optimum by constantly decreasing the difference between upper and lower bounds until they finally meet. Hence, with the same initial solution, a deterministic algorithm will always take the same route to give the same final solution. Stochastic methods, on the other hand, employ random search techniques which are easy to implement, robust and can be effectively parallelized (Mendivii *et* *al.*, 1999). Even though stochastic methods do not guarantee the global optimum, they are very effective in finding near-optimal solutions. Deterministic methods encounter problems while solving large design problems, as they must calculate upper and lower bounds and the bounds must meet to get a globally optimal solution. In case of a non-convex problem, multiple local optima are present. If the algorithm gets stuck in a local optimum, it's difficult to escape that optimum in a deterministic method. However, as the solutions are generated randomly in stochastic algorithms, they offer a great advantage of escaping local optima and achieving globally optimal or near-optimal solution. It is very important to note that as the QSPRs have limited accuracy, there is no guarantee that the global optimum provided by a deterministic method would be better than the near optimal solution found by a stochastic method. A stochastic method run multiple times gives a list of different near-optimal solutions. The user can select a criterion like cost or ease of synthesis to rank these solutions.

4.5 Summary of antibiotic design optimization problem

The antibiotic design problem formulated in this chapter can be stated as follows:

Objective function:

$$Min\ S = w_{MIC}\left|\frac{P_{MIC(target)} - P_{MIC(predicted)}}{P_{MIC(target)}}\right| + w_{Log\ P}\left|\frac{P_{Log\ P(target)} - P_{Log\ P(predicted)}}{P_{Log\ P(target)}}\right| \tag{4.3}$$

Subject to:

$$\begin{aligned} -\log(MIC) = {} & -0.002 * Mor01u + 0.09 * Mor02u - 0.082 * Mor03u - 0.127 * Mor04u \\ & - 0.634 * Mor05u - 0.413 * Mor06u + 0.19 * Mor07u - 0.581 * Mor08u \\ & - 0.237 * Mor09u - 2.509 \end{aligned} \tag{4.4}$$

$$\log(\mathrm{P}) = 0.022 * Mor01u + 0.011 * Mor02u + 0.624 * Mor03u - 0.41 * Mor04u + 2.312 * Mor05u - 2.227 * Mor06u + 0.497 * Mor07u - 0.465 * Mor08u + 0.589 * Mor09u - 11.375 \tag{4.5}$$

$$I(s) = \sum_{i=2}^{N} \sum_{j=1}^{i-1} A_i A_j \frac{\sin(sr_{ij})}{sr_{ij}} \tag{4.6}$$

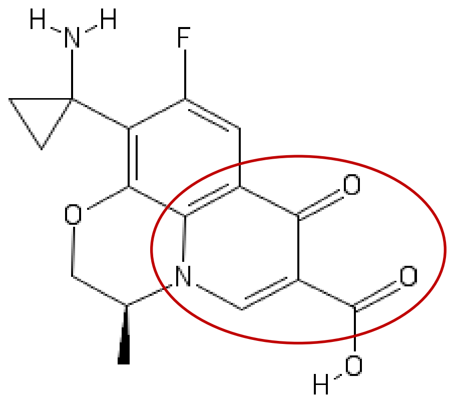


Figure 4.3: Pazufloxacin molecule

The objective function stated as Equation 4.3 aims to minimize the difference between target and predicted property values while subjected to a set of constraints. A few of these constraints are explicitly present as equations in the problem, which are Equations 4.4, 4.5 and 4.6. Equations 4.4 and 4.5 are the QSPRs for calculating MIC and Log *P* respectively, while Equation 4.6 is used for calculating MoRSE descriptors. The remaining constraints are present implicitly. They are structural feasibility and connectivity constraints along with a constraint that keeps the pharmacophore structure intact. The circled portion shown in Figure 4.3 is the pharmacophore.

The solutions to the optimization problem formulated in this chapter are antibiotic candidate molecules possessing desired MIC and Log *P* values. In this work, these solutions are found by using the ABC algorithm, which will be discussed in the next Chapter.

Chapter 5

Artificial Bee Colony Algorithm

Many engineering optimization problems are nonlinear and constrained in nature. The optimization problem formulated in Chapter Four for antibiotic molecular design is nonlinear and finding solutions to it is computationally expensive as there are more than 100,000 feasible solutions. As mentioned previously in Chapter Four, nonlinear optimization problems are difficult to solve compared to the linear ones. Especially if a nonlinear problem is non-convex in nature, it becomes even more difficult to find solutions for it as there could be numerous local optima where an algorithm may get stuck. This issue does not arise when the problem is convex as there is only one local optimum. Numerous algorithms have been reported in the past to solve such problems including genetic algorithm, particle swarm optimization, differential evolution algorithm. All these algorithms are commonly used for finding near optimal solutions in a reasonable time (Karaboga and Basturk, 2008). Karaboga and Akay (2009) compared the performance of the Artificial Bee Colony (ABC) algorithm with genetic algorithms, particle swarm optimization, a differential evolution algorithm and an evolutionary strategy algorithm on a large set of unconstrained functions and found that ABC is better or comparable in performance with respect to these algorithms.

The ABC algorithm is based on the simple idea of the foraging behavior of honeybees, hence its implementation is also easy and straightforward. Among the algorithms that have been reported based on honeybee swarm behavior, the ABC algorithm is the most widely studied and has found many applications in real-world problems (Karaboga *et al.*, 2014). ABC has been found to be quite successful at producing good results with low computational cost and in solving single objective optimization problems. A few examples where ABC has been used for solving MINLPs are as follows. Ayan and Kilic (2012) used ABC to minimize active power loss in power systems. Kaur *et al*. (2014) solved a non-convex, nonlinear problem for minimization of losses in distribution systems. Ajorlou *et al*. (2011) generated an optimal sequence of jobs for minimizing the completion time. Also, the ABC algorithm can handle unconstrained and constrained problems efficiently including combinatorial problems (Karaboga and Basturk, 2007). Therefore, in this work, the ABC algorithm is used for solving the molecular design problem formulated in previous chapter.

Swarm intelligence is a research branch that models the behavior of self-organizing agents. Bonabeau has defined swarm intelligence as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies" (Bonabeau *et. al*., 1999). The term 'swarm' is used in general sense to represent a group of individuals interacting/ working together. One of the typical examples of swarm system is bees swarming around their hive. A flock of birds can be thought of as a swarm whose individual agents are birds. Similarly, an ant colony is swarm of ants, an immune system is a swarm of cells and molecules and a crowd is swarm of people.

5.1 Approach of ABC algorithm

An idea for numerical optimization based on the behavior of honeybee swarm was reported by Karaboga (2005). He simulated the idea of real bees working around their hive into an algorithm and called it 'artificial bee colony algorithm'. As per the model described by Karaboga, the colony of artificial bees consists of three types of bees: employed bees, onlooker bees and scout bees. All the bees work in an organized fashion to constantly find a better food source. A bee going to the food source visited by itself previously is called an 'employed bee'. A bee that makes decision to choose a food source is named an ‘onlooker bee’. A bee carrying out random search is called a ‘scout bee’. The half of the colony consists of the employed bees and the remaining half includes the onlookers. The employed bee whose food source is exhausted by the employed and onlooker bees becomes a scout. The exchange of information about quality of food source is necessary for the bees to perform effectively. This exchange occurs through a dance performed by bees called ‘waggle dance’ and the area around hive where this dance takes place is called ‘dancing area’. The main steps of the algorithm are as follows:

1. Initialize
2. Repeat:
    i. Send the employed bees to the food sources in the memory;
    ii. Send the onlooker bees to the food sources in the memory;
    iii. Send the scouts to the search area for discovering new food sources;
    iv. Store the best solution found in this cycle.
3. Until (requirements are met).

Each cycle goes through three main stages: sending the employed bees onto the food sources where they measure the nectar amount of food source; selecting of the food sources by the onlooker bees based on the information shared by employed bees and determining the nectar amount of those food sources; determining the scout bees and then sending them onto possible food sources.

In the initialization stage, bees randomly select food sources, store their positions in the memory and determine their nectar amounts. Then these bees return to the hive and share information about the nectar of food sources with the bees waiting in the dance area around the hive. At the second stage, the employed bees go to the food source area visited in the previous cycle as that food source exists in their memory, and now choose new food sources by means of visual inspection in the neighborhood of the present one. At the third stage, the onlookers choose food source areas depending on the nectar information collected from the employed bees by their waggle dance. The probability of selection of a food source by an onlooker increases as the nectar amount of food source increases. Therefore, the dance of employed bees is important in terms of exchanging information as it recruits the onlookers for the food source areas with higher nectar amount. The onlookers then arrive at the selected area and choose a new food source by gathering visual information in the neighborhood of the one in the memory. Visual information is based on the comparison of food source positions. When a food source is completely exhausted by the bees, it is abandoned, and a new food source is randomly chosen by a scout bee to replace with the abandoned one. In the model developed by Karaboga, in each cycle at most one scout goes to search for a new food source and the number of employed and onlooker bees are kept equal.

5.2 Application to molecular design

The bees in artificial bee colony work together for finding better food sources in every new cycle. The analogy for that approach is used in this work to find solution to the molecular design problem. The working of ABC algorithm for solving molecular design problem is shown in Fig 5.1. In ABC approach, the bees store the solutions as location of food source in their memory and the quality of solutions is associated with the nectar amount. In our model, the bees are feasible solutions which represent different molecular structures. The quality of solution is evaluated by calculating their property values. The employed and onlooker bees in ABC find new solutions around the previous solutions. Similarly, the employed and onlooker bees in our model create new solutions by changing one group at a time in their previous structures.

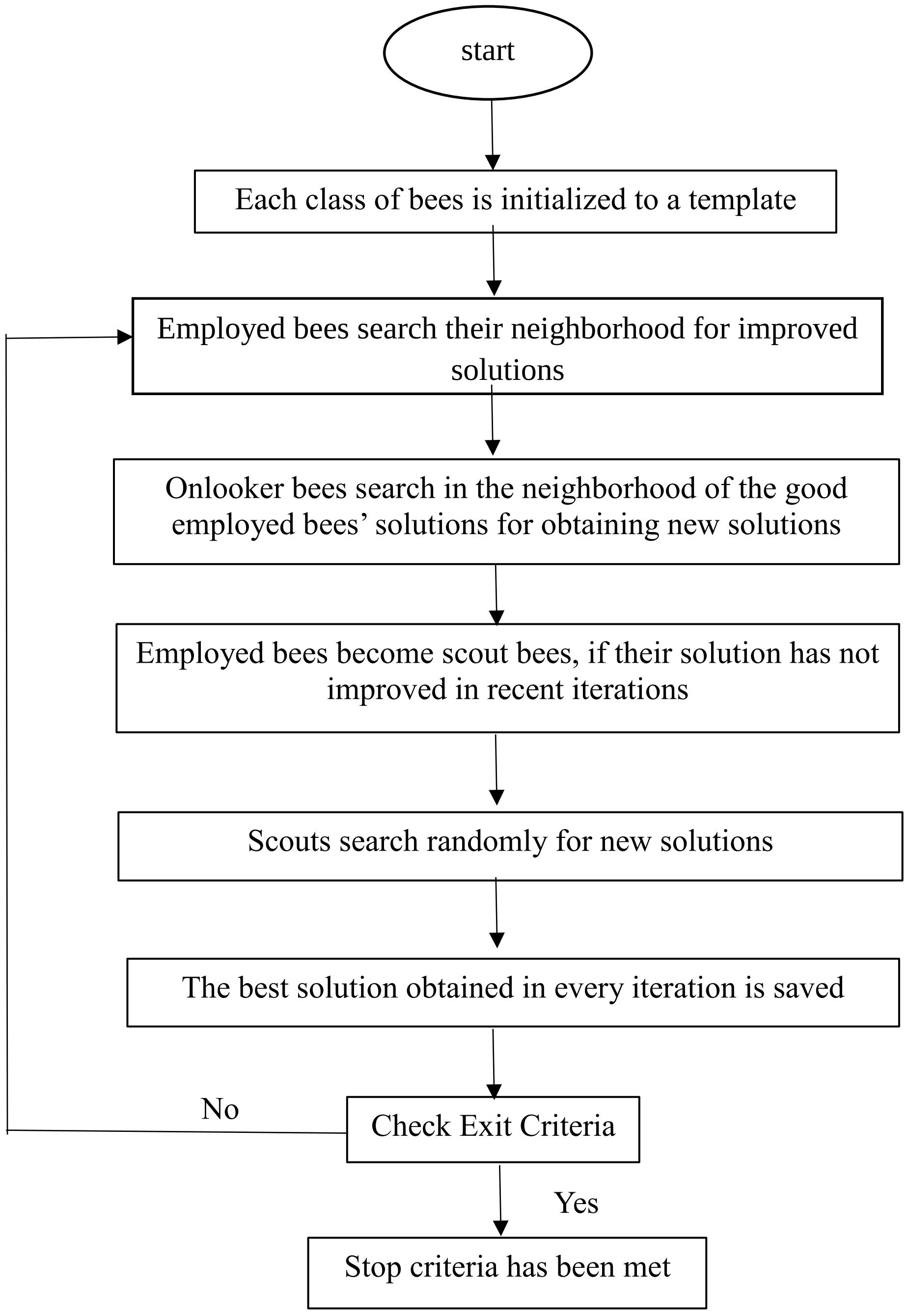

start
Each class of bees is initialized to a template
Employed bees search their neighborhood for improved solutions
Onlooker bees search in the neighborhood of the good employed bees' solutions for obtaining new solutions
Employed bees become scout bees, if their solution has not improved in recent iterations
Scouts search randomly for new solutions
The best solution obtained in every iteration is saved
No
Check Exit Criteria
Yes
Stop criteria has been met

Figure 5.1: Artificial Bee Colony algorithm flowchart for CAMD

At the initialization, all the solutions are adjusted to the structure of template molecule. A template molecule is a base molecule to which changes will be made in the program to find the one that matches target property value. Every cycle goes through three phases: the employed phase, the onlooker phase and the scout phase. In the employed phase, the employed bees make a random change to their template structure by changing only one group. If the new property value is closer to the target property value than previous solution, it abandons the previous solution and stores the new one. This method of selection of solution is called the ‘greedy selection’ method. It makes sure the new solution is not worse than the previous one. At the end of this phase, all the employed bees’ solutions are then assessed in terms of their property values. The solutions are ranked from the best to worst depending on how close they are to the target property value. The onlooker bees then target only a first few of these solutions. The onlookers will randomly pick any of the good solutions and make changes to them by changing one group at a time. The onlookers also follow ‘greedy selection’ criteria and keep the better solution. These solutions are again assessed with respect to their property values and ranked from best to worst. The best solution is saved as the best solution found in that cycle. Finally, in the scout phase, if a solution is not improved within a certain number of iterations, it is abandoned and replaced with a new solution. The algorithm repeats itself through these phases until a near-optimal solution is obtained.

5.3 Characteristics of the ABC algorithm

The control parameters in the ABC algorithm are the number of employed bees, the number of onlooker bees, the number of predetermined cycles after which the scouts will randomly choose new solution and the maximum number of cycles. In addition to these parameters, one more control parameter is considered in this work: the number of good employed bees. Employed bees’ solutions are ranked based on their closeness to the target property values, from best to worst. The first few solutions are selected, based on which onlooker bees choose their new solutions. The employed bees corresponding to these chosen solutions are named good employed bees. The progress of the bee colony can be measured in terms of how rapidly it discovers good food sources. Similarly, the progress of algorithm is dependent on how fast it generates good solutions. Good solutions for this work will be the ones that are closer to the target value. For a robust algorithm, the exploration and exploitation of good solutions should occur simultaneously. In ABC, the employed and onlooker bees perform task of exploiting while scouts do the exploring. The exploring done by scouts helps in achieving global optimum and the exploitation by employed and onlooker bees is good for local optimization.

Even though the applicability of ABC has been explored in numerous fields, there are not many cases where it has been used in molecular design. This is the novel work which has employed ABC approach for molecular design. Solutions to the antibiotic design MINLP, which correspond to near-optimal antibiotic molecular candidates are found by using ABC algorithm and are discussed in the next Chapter.

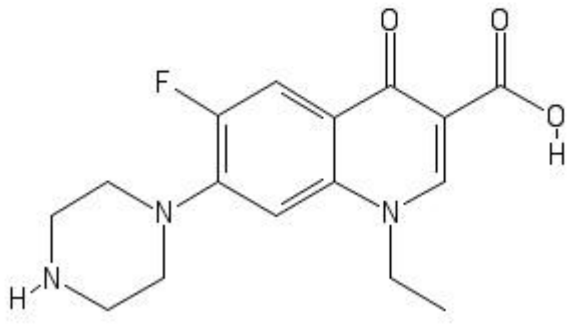
Chapter 6

Results and Discussion

The solution to the antibiotic design problem formulated in Chapter Four is found by using the ABC algorithm described in Chapter Five. A code has been written in the programming language Python to implement the ABC algorithm. The goal is to find antibiotic candidate molecules that match the target property values set for MIC and Log *P*. There are two methods which can be followed to find these candidate molecules. The first is to create new molecules by putting together different atoms/ groups to form a stable structure and checking property values of these molecules until a molecule is found which matches target property values. The second one involves making changes to an existing molecule at different positions and checking their property values until a molecule is found which matches target property values. The later approach, which is called ‘templating’ is applied in this work. The base molecule which will be changed at different positions is called the ‘template’. The data structure used for representation of the molecules, the Python packages used for this work, and the results for specific example cases are discussed below.

6.1 Data structure used for storing molecules and procedure for generating new solutions

There are different data structures which can be used for storing molecular structure numerically. One of them is a PAM, which has been discussed previously. In this work, representation of molecules is accomplished using the SMILES notation for that molecule. The SMILES notation is a string of characters that represents the connections between atoms within a molecule and is unique for each molecule. Hydrogen atoms are usually excluded from the SMILES string. An example is shown in Figure 6.1.



C(=C(C(C(C2C(O)=O)=O)=C1)N(C=2)CC)C(N(CCN3)CC3)=C1F

Figure 6.1: Norfloxacin molecule and its SMILES notation

As mentioned previously, a template molecule is the base molecule which will undergo changes to generate new solutions. A few groups are selected on the template molecule where changes could be made. Every time a new solution is to be generated, one of these groups is picked randomly and replaced with a new group. Thus, a new solution is generated. All the groups considered in this work and the total number of possible structures have already been discussed in Chapter Four. Pazufloxacin is used as the template molecule in this work. The groups where changes can be made are circled in Figure 6.2.

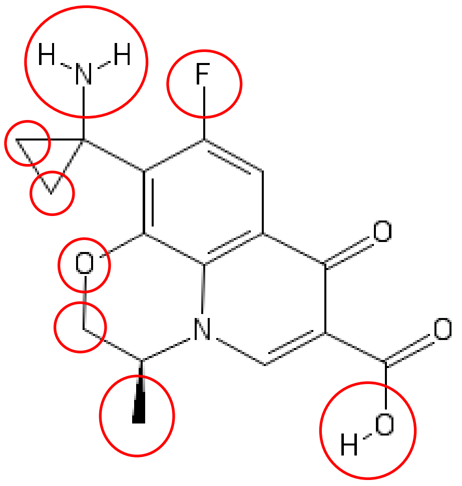


Figure 6.2: Pazufloxacin molecule

In the Python code written to implement the ABC algorithm, this change is made by changing the corresponding position in the SMILES notation for that group. The template molecule SMILES notation is saved for each solution before the iterations begin. When a certain group is to be replaced with a new one, the old group position in the SMILES string is replaced with the new character/characters corresponding to the new group. For example, if -F is replaced with -NH_2 in pazufloxacin, F for fluorine atom from pazufloxacin SMILES notation will be changed to an N for amine group.

Pazufloxacin SMILES:

C(C(=C(C1OC3)N(C2)[C@@H](C)3)C(C(C(O)=O)=2)=O)=C(C=1C(C4)(C4)N)F

SMILES after replacing -F with -NH_2:

C(C(=C(C1OC3)N(C2)[C@@H](C)3)C(C(C(O)=O)=2)=O)=C(C=1C(C4)(C4)N)N

New molecules generated by making changes in the template molecule will again be changed in the next iteration by replacing one group at a time. The process of generating new solutions continues until maximum number of iterations are completed. The goal of following this process is to come across a molecule that will match or will come very close to the target property value.

6.2 Python packages used in this work

This work employs 3D MoRSE descriptors, which depend on the interatomic distances within a molecule. As mentioned above, the representation of molecules is done with the SMILES format, which is a 2D format for representing molecules. In order to use 3D descriptors, this 2D information must be augmented with 3D structural information. Therefore, a package called ‘Open Babel’ was installed in the Python environment for converting 2D connectivity information from SMILES into 3D coordinates of atoms in the molecule. Open Babel is a chemistry toolbox designed for interconverting chemical data from one format to another (O′Boyle *et al.*, 2011). It accepts a variety of molecular file formats and can be used in different programming languages. Open Babel has a C++ library and can be easily called from C++. Additionally, it can also be used in Python, Perl, Ruby, CSharp and Java via language bindings. For example, for using Open Babel in Python, Python bindings are needed. For Python, a module named 'Pybel' is provided which makes it easier to access features of the bindings (O′Boyle *et al*., 2008). Hence, Pybel was also installed into the Python environment. Pybel has functions and classes which make it easier to access the Open Babel libraries from Python, especially for accessing the attributes of atoms and molecules.

To use Open Babel and Pybel, they need to be imported into a Python code. One of the ways in which a molecule is created in Pybel is by reading a string. This string is provided in the form of SMILES notation of that molecule. The function pybel.readstring reads molecule from the given string. For adding hydrogen atoms to molecule, the addh() function is used. The 3D coordinates are computed using make3D() function. By default, Open Babel uses a 50-step procedure for geometry optimization of molecule using the MMFF94 forcefield. This forcefield was developed by Merck and it provides good accuracy for a range of organic and drug-like molecules (Halgren, 1996). Once 3D coordinates of a molecule are obtained, the interatomic distances for all atomic pairs can be easily calculated.

6.3 Validation of code

It is necessary to validate the code to check if it is performing correctly. As the problem size of the antibiotic design problem is very big, a small sized problem is considered for validating the code. This will help in obtaining output in a shorter time. A small sized problem is formulated by decreasing the number of positions where changes could occur, and the number of groups used for making these changes. The template molecule used is pazufloxacin molecule. The groups which can be replaced for this validation example are circled in Figure 6.3.

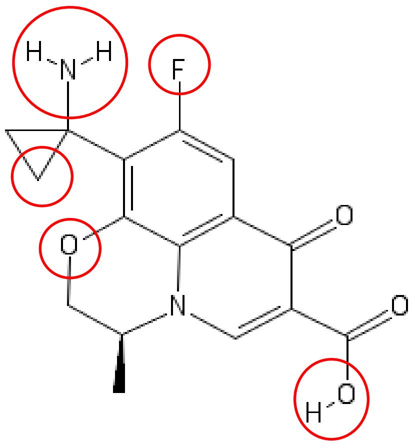


Figure 6.3: Template molecule

The number of feasible solutions to this smaller problem are calculated as follows:

Single valency groups used: -NH_2, -F, -OH, -CH_3, -CH_2-NH_2, -CH_2-CH_2-OH

Double valency groups used: -CH_2-, -O-

Total number of groups having valency one = 6

Total number of groups having valency two = 2

Number of positions where replacement can occur with valency one groups = 3

Number of positions where replacement can occur with valency two groups = 2

Total number of possible structures = 6×6×6×2×2 = 864

Thus, the total number of feasible solutions/ structures are 864.

The accuracy of the code is verified by following a simple procedure. A molecule is created by making changes at some positions on the template. This molecule is shown in Figure 6.4 and the corresponding changes are circled. The -log (MIC) of this structure is calculated by using Equation 4.4. This -log (MIC) value is now used as the target property value in the code. The Python code will try to find a molecule which has -log (MIC) very close or equal to the target value set. If the output is the molecule shown in Figure 6.4, it can be confirmed that the code is running without any problems.

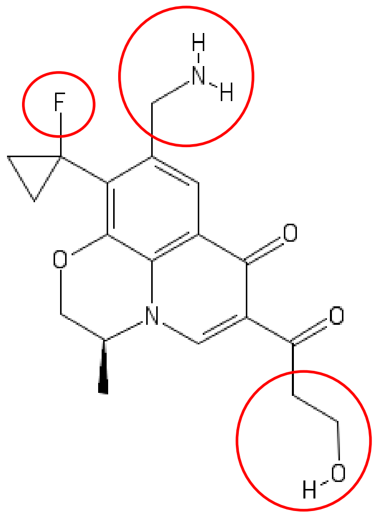


Figure 6.4: Molecule used for validating Python code

The -log (MIC) of structure shown in Figure 6.4 is found to be 4.901 after calculating with equation 4.4. The code is run with 4.901 as the target property value and ABC control parameters given below. ABC control parameters have been discussed previously in Chapter Five.

Maximum number of iterations = 30

Number of employed bees = 6

Number of good employed bees = 4

Number of onlooker bees = 10

Number of scout bees = 1

The molecule shown in Figure 6.4 was obtained at the twenty-second iteration in one of the first few runs of the code. Thus, the validation of the code was completed.

6.4 Selection of target property values

As the goal of this work is to develop antibiotic leads which will show activity against *S. aureus*, the target value for -log (MIC) was chosen based on the data used for developing MIC QSPR. This data contains fluoroquinolones showing antibacterial activity against *S. aureus*. The target value was chosen close to the lowest value in the data, as lower MIC values mean highly potent antibiotics. The lowest MIC has -log (MIC) equal to 7.4 and the highest MIC has -log (MIC) equal to 3.2 in the data. Hence, the target value selected for -log (MIC) is 6.

Log *P* value is an indicator of hydrophobicity of the molecule. A value above 0 indicates that the molecule is hydrophobic, whereas Log *P* value less than 0 means the molecule is hydrophilic. Some studies have found a relation between molecular hydrophobicity and the concentration of fluoroquinolone accumulated inside bacteria *S. aureus* (Bazile *et al*.,1992, McCaffrey *et al*.,1992, Yoshida *et al*., 1990). Bazile *et al.* (1992) reported that a bulky and hydrophobic quinolone having Log *P* value 1.05, was accumulated in higher concentration in *S. aureus*. Nikaido and Thanassi (1993) suggested that resistant strains which efflux fluoroquinolones actively from their cells can be effectively treated, if higher rate of influx is present. Piddock *et al.* (2001) conducted a study which showed that there is a direct relationship between the fluoroquinolone accumulation and molecular hydrophobicity. Hence, it seems a hydrophobic fluoroquinolone candidate would be favorable for treating *S. aureus*. It should also be considered that the antibiotic candidate should not be too hydrophobic, such that it becomes difficult for the molecule to pass through hydrophilic barriers. Therefore, 1.5 was chosen as the target Log *P* value.

6.5 Molecular structures obtained as results

Results are obtained for three different cases. The first case gave the MIC value a 100% weight and Log *P* a 0% weight, the second case gave the MIC value a 0% weight and Log *P* a 100%

weight and, the third case sets both the properties to a 100% weight. Cases one and two would give the molecules having property values equal to or very close to the individual target values and the third case would try to find a molecule which comes very close to both the target values.

For ABC algorithm, different control parameters are used which have been discussed in Chapter Five. Different combinations of control parameters were used to obtain the results. This was done to study the effect of individual parameters and their combination on quality of results.

The objective function for antibiotic design, defined in Chapter Four is stated below.

$$Min\ S = w_{MIC}\left|\frac{P_{MIC(target)} - P_{MIC(predicted)}}{P_{MIC(target)}}\right| + w_{Log\ P}\left|\frac{P_{Log\ P(target)} - P_{Log\ P(predicted)}}{P_{Log\ P(target)}}\right|$$

6.5.1 Case one

In this case, w_{MIC} = 1 and $w_{Log\ P}$= 0, so the objective function can be written as below.

Objective function:

$$Min\ S = \left|\frac{6 - P_{MIC(predicted)}}{6}\right|$$

The variation in control parameters along with the absolute difference between target and predicted -log (MIC) is shown in Table 6.1 for a few runs. For each run, the computational time was one minute per iteration.

The best result obtained is shown in the fourth observation. Comparing parameters from fourth observation with the rest, it is observed that higher number of good employed bees help in getting a better result. The graph showing the decrease in absolute difference between target and predicted -log (MIC) corresponding to the best result is given in Figure 6.5. The molecular structure corresponding to the best result is given in Figure 6.6. This structure has -log (MIC) equal to 6.000029, which is closest to the target value among all the results obtained for case one. It should be noted that a molecular structure that corresponds to -log (MIC) equal to 6 may not exist.

Therefore, a near optimal solution with the least difference between target and predicted property values can be considered as the best solution. Actual MIC for best result is 9.99×10^{-7} mol/L. The groups that are different from the template molecule are circled in Figure 6.6.

Table 6.1: Control parameters and results for case one

Observation number	**Maximum number of iterations**	**Number of employed bees**	**Number of good employed bees**	**Number of onlooker bees**	**Number of scout bees**	**Absolute difference between target and predicted -log (MIC)**
1	20	50	5	20	2 after every 10 iterations	0.0004
2	40	50	5	20	2 after every 10 iterations	0.0006
3	20	20	5	50	2 after every 10 iterations	0.001
4	20	50	10	20	2 after every 10 iterations	0.00003
5	20	50	5	20	2 after every 5 iterations	0.0003

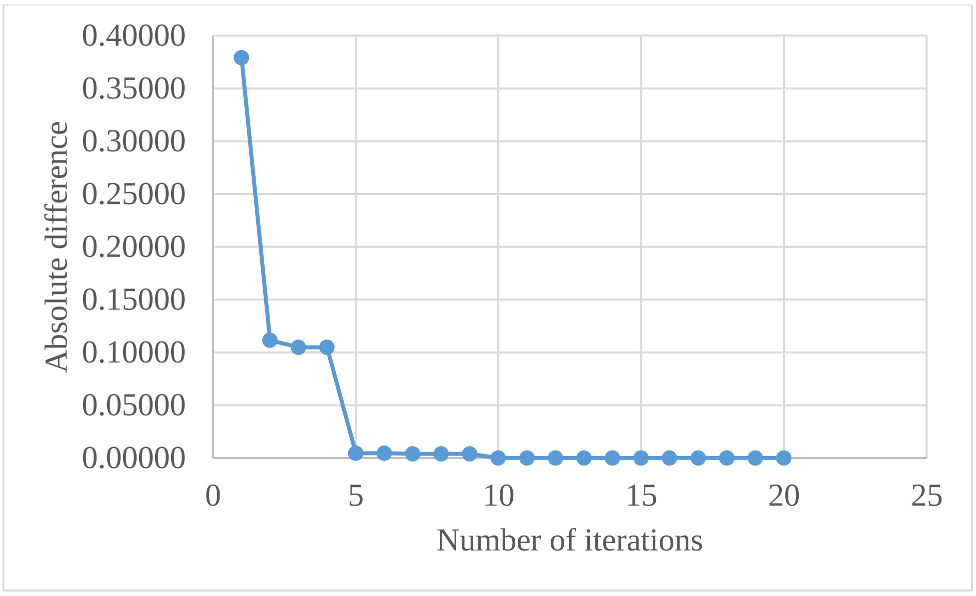


Figure 6.5: Change in absolute difference of target and predicted -log (MIC) with number of iterations

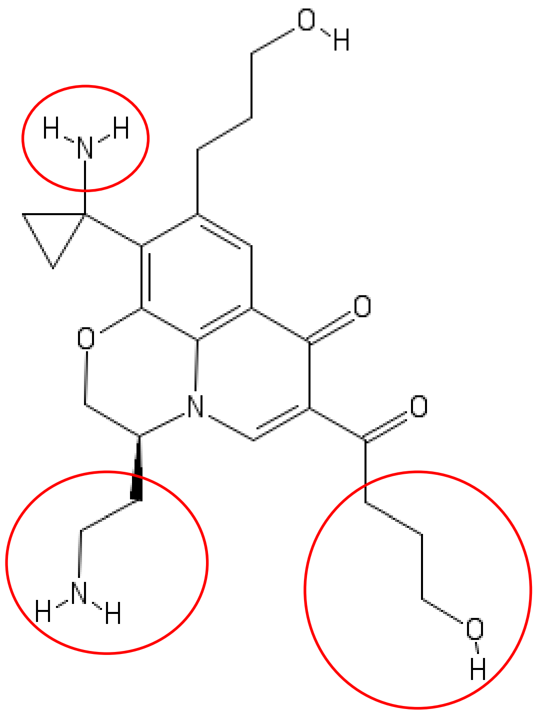


Figure 6.6: Best structure obtained for case one

6.5.2 Case two

In this case, w_{MIC} = 0 and $w_{Log\ P}$= 1, so the objective function can be written as below.

Objective function:

$$Min\ S = \left|\frac{1.5 - P_{Log\ P(predicted)}}{1.5}\right|$$

The variation in control parameters along with the absolute difference between target and predicted Log *P* is shown in Table 6.2 for a few runs. For each run, the computational time was one minute per iteration.

The best result obtained is found in the first and fourth observations. Comparing parameters from first and fourth observations with the rest, it is observed that higher number of good employed bees help in getting a better result. The graph showing the decrease in absolute difference between target and predicted Log *P* for one of these observations is given in Figure 6.7. The molecular structure corresponding to first and fourth observation is the same and is given in Figure 6.6. This structure has Log *P* equal to 1.4968, which is closest to the target value among all the results obtained for case two. The groups that are different from the template molecule are circled in Figure 6.8.

Table 6.2: Control parameters and results for case two

Observation number	**Maximum number of iterations**	**Number of employed bees**	**Number of good employed bees**	**Number of onlooker bees**	**Number of scout bees**	**Absolute difference between target and predicted Log *P***
1	20	50	5	20	2 after every 10 iterations	0.0032
2	40	50	5	20	2 after every 10 iterations	0.007
3	20	20	5	50	2 after every 10 iterations	0.034
4	20	50	10	20	2 after every 10 iterations	0.0032
5	20	50	5	20	2 after every 5 iterations	0.031

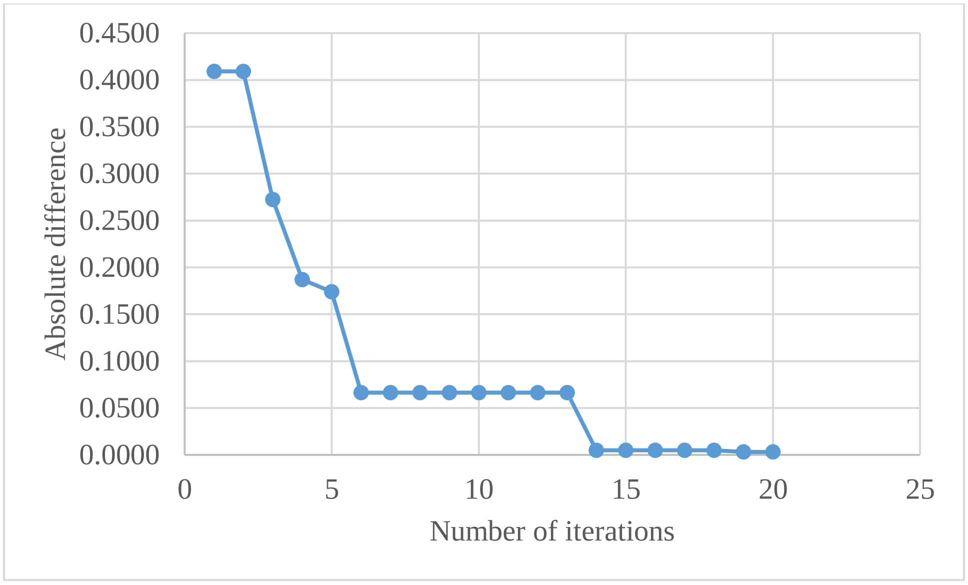


Figure 6.7: Change in absolute difference of target and predicted Log *P* with number of iterations

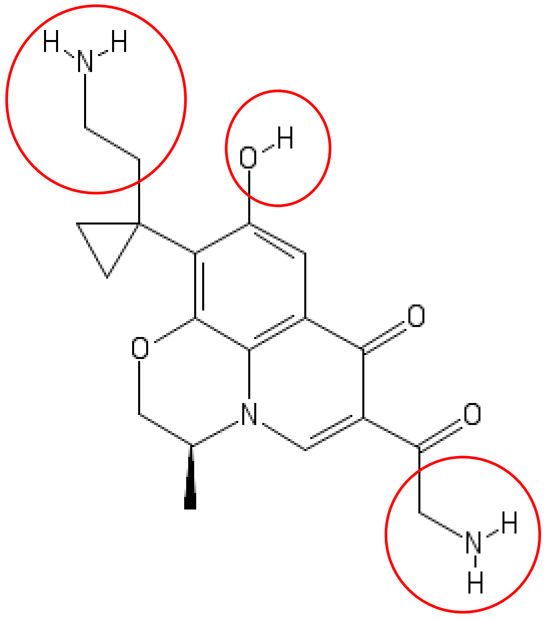


Figure 6.8: Best structure obtained for case two

6.5.3 Case three

In this case, w_{MIC} = 1 and $w_{Log\ P}$= 1, so the objective function can be written as below.

Objective function:

$$Min\ S = \left|\frac{6 - P_{MIC(predicted)}}{6}\right| + \left|\frac{1.5 - P_{Log\ P(predicted)}}{1.5}\right|$$

The variation in control parameters along with the sum of absolute difference between target and predicted properties is shown in Table 6.3 for a few runs. For each run, the computational time was one minute per iteration.

The best result obtained is given in first observation. Comparing parameters from first observation with the rest, it is observed that no parameter particularly affects the final result in this case. The graph showing the decrease in the sum of absolute difference between target and predicted properties is given in Figure 6.9. It can be observed that the final sum of differences between target and predicted properties is quite higher compared to the previous cases. The reason behind this is that there may not be a molecule which possesses -log (MIC) equal to 6 and Log *P* equal to 1.5. So, whenever more than one property is optimized, trade-offs occur between the properties to find optimal solution. The molecular structure corresponding to the best result is given in Figure 6.10. This structure has -log (MIC) equal to 5.9695 and Log *P* equal to 1.9979, which is closest to the target value among all the results obtained for case three. The groups that are different from the template molecule are circled in Figure 6.10.

Table 6.3: Control parameters and results for case three

Observation number	**Maximum number of iterations**	**Number of employed bees**	**Number of good employed bees**	**Number of onlooker bees**	**Number of scout bees**	**Sum of absolute difference between target and predicted properties**
1	20	50	5	20	2 after every 10 iterations	0.528
2	40	50	5	20	2 after every 10 iterations	0.87
3	20	20	5	50	2 after every 10 iterations	0.616
4	20	50	10	20	2 after every 10 iterations	0.583
5	20	50	5	20	2 after every 5 iterations	0.87

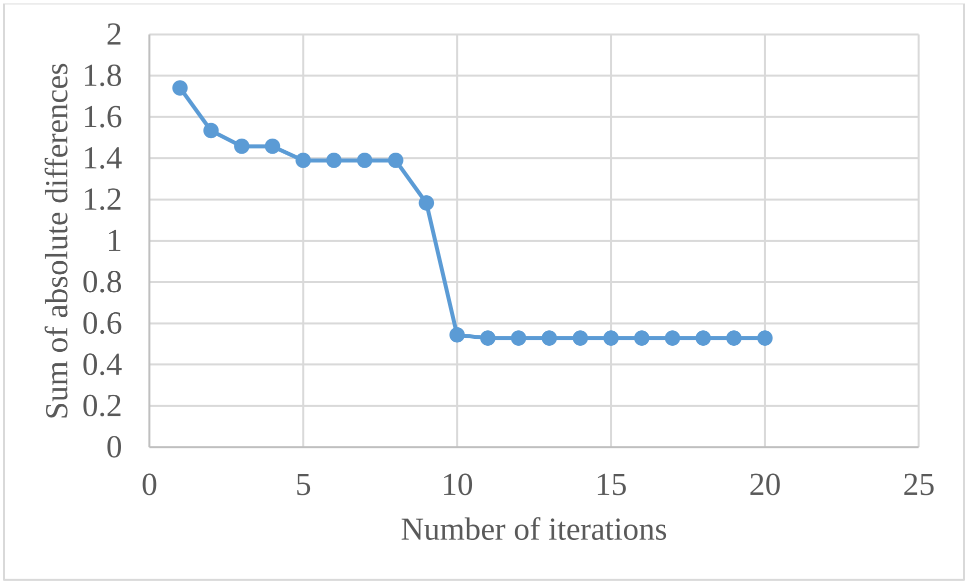


Figure 6.9: Change in the sum of absolute difference of target and predicted properties with number of iterations

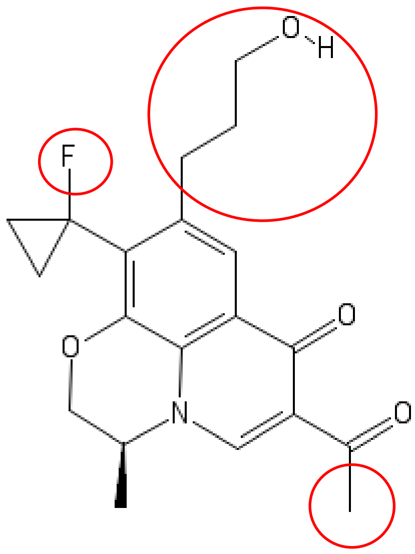


Figure 6.10: Best structure obtained for case three

6.6 Significance of control parameters

The goal of varying parameters in each case was to observe how each parameter or combination of different parameters affects the results and if they can be adjusted to obtain near optimal results faster i.e. decrease the total number of evaluations. After studying cases one, two and three, it can be observed that no single parameter particularly helps in finding a near optimal solution faster or defines the quality of result. The combination of control parameters, however, does affect the result to some extent.

Overall, it was observed that there should be enough number of employed bees for given problem size. This is important because a larger number means that there will larger number of initial solutions to begin with. First few solutions which are closer to the target property value are chosen as good employed bee solutions. The onlooker bees then locally search around these good solutions. Therefore, even if there are lesser onlooker bees than employed bees, there is still better chance of finding near optimal solutions. The number of good employed bees chosen should not be so few that they will continue to search around only few good solutions, and may get stuck in local optima. The number of good employed bees chosen should also include moderately good employed bees' solutions so that exploration is wider. Scout bees do not affect the progress of algorithm in any way, but they replace the worst employed bees' solutions after a few iterations. This is done hoping that the replacement will result in making the future solutions better in further iterations. Two scout bees were used for every run. One scout bee was set to the best onlooker bee solution found in that iteration, to increase chances for finding near optimal solution faster. The other scout bee was set as the template molecule to increase the chances of escaping local optima. The number of iterations does not affect the output in any way after the algorithm converges to a near optimal solution. There should be enough number of iterations to reach this near optimal point.

Any combination of control parameters that satisfies the above criteria should be able to find near optimal solutions. It should be noted that the same combination of control parameters may give different results for different runs. This happens because the ABC algorithm is inherently stochastic in nature, so we cannot control the generation of new solutions. The new solutions would however be discarded, if they are not better than previous ones.

An efficient algorithm should be able to find the optimal or near optimal results with the least number of evaluations. The number of solutions evaluated before the best result was obtained were 700, 1330 and 700 for cases one, two and three respectively. All these numbers are much smaller compared to all the feasible solutions, 160,000. This is a good indicator that ABC is an efficient algorithm to find near optimal solutions. The only drawback of current Python code is a larger computational time. This could be arising because of the lengthy procedure of 3D coordinates generation by Open Babel.

Chapter 7

Conclusions and Recommendations

In this work, combinatorial optimization approach has been used in the area of molecular design. The previous use of optimization techniques in designing solvents, polymers, pharmaceutical products has been extended to the design of novel antibiotic molecules. The property values of molecules were related to their structures by using molecular descriptors. The properties considered in this work, MIC and Log *P* values, were linearly correlated with molecular structures using two types of molecular descriptors, connectivity indices and 3D MoRSE descriptors. Zeroth, first and second order connectivity indices and unweighted 3D MoRSE descriptors were used for development of correlations. The correlations showing better quality in terms of correlation coefficient and predictive capability were employed in the molecular design problem. These structure-property correlations, structural feasibility and connectivity constraints along with an objective function were formulated as an MINLP antibiotic design problem. This MINLP design problem was solved using the ABC algorithm to find near optimal solutions.

7.1 Conclusions

A computer-aided molecular design (CAMD) approach has been successfully used for the design of novel antibiotic candidates in this work. The goal was to match the MIC and Log *P* values of resulting molecules as closely as possible to the target values. Among molecular connectivity indices and 3D MoRSE descriptors, the correlations with 3D MoRSE descriptors showed better results, when compared on the basis of correlation coefficient and predictive ability.

Therefore, these structure-property correlations were employed for prediction of properties in the molecular design, for ultimately generating novel antibiotic molecules.

An improvement over previous work in the field of CAMD is the use of 3D molecular descriptors. Most of previous CAMD work has been performed using group contribution methods and topological indices like connectivity indices. For designing compounds based on physicochemical properties, 2D molecular descriptors like connectivity indices contribute enough information from chemical structure to form a good correlation with property values. This work involves prediction of biological activity of a compound, which is dependent on the interaction of a molecule with biological target. Therefore, molecular descriptors that contain information about 3D structure of molecule should be used. 3D MoRSE descriptors depend on the distances between all atomic pairs present in the molecule, and thus account for 3D characteristics of molecule.

The ABC algorithm derived from the foraging behavior of honeybees was implemented to find solutions to the antibiotic design problem. A small problem was solved using this algorithm for verifying the Python code and the results indicated the convergence to optimal solution in very few evaluations. The same code was used for solving bigger problem, specifically three different cases and results provided numerous near-optimal solutions i.e. molecular structures having property values closer to target property values. The solution to a larger problem was also obtained in a smaller number of evaluations compared to the problem size. Thus, it was shown that a large MINLP formulated for antibiotic design can be solved efficiently with a stochastic algorithm like ABC to generate near optimal solutions.

New antibiotic leads, which have been designed in this work, may not necessarily end up being new choices for treating *S. aureus* infections. It should be noted that a lot of leads are structurally very similar to final drug that gets approved. So, shortlisting leads is a very important stage in the process of drug discovery. The drug leads go through a rigorous process of preclinical and clinical trials before any of them is approved for official use. The purpose of finding leads is to narrow down a large database for a few potential candidates. In this work, antibiotic leads are shortlisted from large molecular set containing 160,000 molecules based on target values selected for two properties. So, this work provides a methodology for other molecular design problems where new candidates can be found based on the relevant property values.

7.2 Recommendations

The stepwise approach for synthesis of the novel antibiotic molecules has not been determined in this work. In future work, the antibiotic candidates provided in Chapter Six could be synthesized and tested for their MIC and Log *P* values.

More data can be collected for fluoroquinolones showing antibacterial activity against *S. aureus* for MIC and Log *P*. This data can be included in the current data to make even better QSPRs. Even though current QSPRs with 3D MoRSE descriptors show good correlation coefficient and predictability, other 3D descriptors can be tested to see if they generate better mathematical models for property prediction.

The molecular design problem can be made larger by adding more basic groups and properties. This allows user to generate variety of chemical structures suited for different purposes. Current molecular design is done using combinatorial technique where groups are replaced. New aspects can be added to the molecular design which allow ring formation or building of new groups on current groups in addition to replacing them.

For large non-convex MINLPs, it is difficult to find near optimal solutions in reasonable amount of time. This is also the drawback of the current Python code. It took approximately a minute to compute solutions for 70 molecules. It can be understood why computational time is large in this case. There is an optimization problem that needs to be solved for every molecule that is generated. This problem is finding a stable, minimum energy conformation of this newly generated molecule so that its 3D coordinates can be determined. This has been done using Open Babel and Pybel within a Python environment. This process makes the computational time to find near optimal solutions larger compared to the same sized molecular design problem that doesn’t need 3D coordinates. There are two ways which could solve this problem. First is, adding a custom function to find a low energy conformation of the molecule in a simple way from 2D structure so that there is no need to use Open Babel and Pybel. The second one is use of parallel computing. An algorithm can be parallelized either by performing some steps of an algorithm in parallel while maintaining the same flow as original algorithm, or by fitting the algorithm into some parallel computing structure. The ABC algorithm which has been used in this work has a series of iterations that needs to be performed in order to reach the final solution. Every iteration goes through employed bees’ and onlooker bees’ phase where evaluation of solutions is performed. For example, let us consider an ABC system that has 10 maximum number of iterations, 5 employed bees and 5 onlooker bees. Hence, every iteration has 10 total evaluations. Here, parallelization can be done at the employed phase and onlooker phase. In normal ABC algorithm, 5 employed bees’ solutions will be evaluated one after the other and the next step will be evaluation of 5 onlooker bees’ solutions one after the other. In a parallelized ABC algorithm, 5 employed bees’ solutions can be evaluated parallelly and the onlooker phase which is dependent on employed bees’ solutions can also be parallelized. Thus, for one iteration, the computational time can be reduced by 5 times at the employed phase and at the onlooker phase. For the antibiotic design problem, the employed bees and onlooker bees are much larger in number. Therefore, parallelization would end up reducing the computational time significantly.

References

Achenie, Luke EK, and Manish Sinha. "Interval global optimization in solvent design." *Reliable computing* 9, no. 5 (2003): 317-338.

Achenie, Luke, Venkat Venkatasubramanian, and Rafiqul Gani, eds. *Computer aided molecular design: theory and practice*. Vol. 12. Elsevier, (2002).

Ajorlou, Saeede, Issac Shams, and Mirbahador G. Aryanezhad. "Optimization of a multiproduct conwip-based manufacturing system using artificial bee colony approach." In *Proceedings of the international multiconference of engineers and computer scientists (IMECS 2011)*. 2011.

Aldred, Katie J., Robert J. Kerns, and Neil Osheroff. "Mechanism of quinolone action and resistance." *Biochemistry* 53, no. 10 (2014): 1565-1574.

Andrews, Jennifer M. "Determination of minimum inhibitory concentrations." *Journal of antimicrobial chemotherapy* 48, no. suppl_1 (2001): 5-16.

Ayan, Kürşat, and Ulaş Kılıç. "Artificial bee colony algorithm solution for optimal reactive power flow." *Applied soft computing* 12, no. 5 (2012): 1477-1482.

Basak, Subhash C., V. R. Magnuson, G. J. Niemi, R. R. Regal, and G. D. Veith. "Topological indices: their nature, mutual relatedness, and applications." *Mathematical modelling* 8 (1987): 300-305.

Bath, Peter A., Andrew R. Poirrette, Peter Willett, and Frank H. Allen. "The extent of the relationship between the graph-theoretical and the geometrical shape coefficients of chemical compounds." *Journal of chemical information and computer sciences* 35, no. 4 (1995): 714-716.

Bazile, S., N. Moreau, D. Bouzard, and M. Essiz. "Relationships among antibacterial activity, inhibition of DNA gyrase, and intracellular accumulation of 11 fluoroquinolones." *Antimicrobial agents and chemotherapy* 36, no. 12 (1992): 2622-2627.

Bermejo, Marival, Virginia Merino, Teresa M. Garrigues, Jose M. Pla Delfina, Antonio Mulet, Patrick Vizet, Gérard Trouiller, and Christiane Mercier. "Validation of a biophysical drug absorption model by the PATQSAR system." *Journal of pharmaceutical sciences* 88, no. 4 (1999): 398-405.

Bicerano, Jozef. *Prediction of polymer properties*. cRc Press, 2002.

Bonabeau, Eric, Directeur de Recherches Du Fnrs Marco, Marco Dorigo, Guy Théraulaz, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. No. 1. Oxford university press, 1999.

Camarda, Kyle V., and Costas D. Maranas. "Optimization in polymer design using connectivity indices." *Industrial & engineering chemistry research* 38, no. 5 (1999): 1884-1892.

Camarda, Kyle V., and Patrick Sunderesan. "An optimization approach to the design of value-added soybean oil products." *Industrial & engineering chemistry research* 44, no. 12 (2005): 4361-4367.

Chavali, Sunitha, Bao Lin, David C. Miller, and Kyle V. Camarda. "Environmentally-benign transition metal catalyst design using optimization techniques." *Computers & chemical engineering* 28, no. 5 (2004): 605-611.

Churi, Nachiket, and Luke EK Achenie. "Novel mathematical programming model for computer aided molecular design." *Industrial & engineering chemistry research* 35, no. 10 (1996): 3788-3794.

Dang, Poonam, and A. K. Madan. "Structure-activity study on anticonvulsant (thio) hydantoins using molecular connectivity indices." *Journal of chemical information and computer sciences* 34, no. 5 (1994): 1162-1166.

Devinyak, Oleg, Dmytro Havrylyuk, and Roman Lesyk. "3D-MoRSE descriptors explained." *Journal of Molecular Graphics and Modelling* 54 (2014): 194-203.

Draper, Norman R., and Harry Smith. "Advanced regression analysis." (1966).

Duvedi, Amit P., and Luke EK Achenie. "Designing environmentally safe refrigerants using mathematical programming." *Chemical Engineering Science* 51, no. 15 (1996): 3727-3739.

Efron, Bradley. "Estimating the error rate of a prediction rule: improvement on cross-validation." *Journal of the American statistical association* 78, no. 382 (1983): 316-331.

Eslick, John C., Q. Ye, J. Park, Elizabeth M. Topp, P. Spencer, and Kyle V. Camarda. "A computational molecular design framework for crosslinked polymer networks." *Computers & chemical engineering* 33, no. 5 (2009): 954-963.

Estrada, Ernesto, Enrique Molina, and Iliana Perdomo-López. "Can 3D structural parameters be predicted from 2D (topological) molecular descriptors?." *Journal of chemical information and computer sciences* 41, no. 4 (2001): 1015-1021.

Estrada, Ernesto, Santiago Vilar, Eugenio Uriarte, and Yaquelin Gutierrez. "In silico studies toward the discovery of new anti-HIV nucleoside compounds with the use of TOPS-MODE and 2D/3D connectivity indices. 1. Pyrimidyl derivatives." *Journal of chemical information and computer sciences* 42, no. 5 (2002): 1194-1203.

Galvez, Jorge, Ramon Garcia-Domenech, Jesús Vicente de Julian-Ortiz, and Rosa Soler. "Topological approach to drug design." *Journal of chemical information and computer sciences* 35, no. 2 (1995): 272-284.

Gani, Rafiqul, Bjarne Nielsen, and Aage Fredenslund. "A group contribution approach to computer-aided molecular design." AIChE Journal 37, no. 9 (1991): 1318-1332.

Gasteiger, Johann, Jens Sadowski, Jan Schuur, Paul Selzer, Larissa Steinhauer, and Valentin Steinhauer. "Chemical information in 3D space." *Journal of Chemical Information and Computer Sciences* 36, no. 5 (1996): 1030-1037.

Gebreslassie, Berhane H., and Urmila M. Diwekar. "Efficient ant colony optimization for computer aided molecular design: case study solvent selection problem." *Computers & chemical engineering* 78 (2015): 1-9.

Halgren, Thomas A. "Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94." *Journal of computational chemistry* 17, no. 5-6 (1996): 490-519.

Kapetanovic, I. M. "Computer-aided drug discovery and development (CADDD): in silico-chemico-biological approach." *Chemico-biological interactions* 171, no. 2 (2008): 165-176.

Karaboga, Dervis, and Bahriye Akay. "A comparative study of artificial bee colony algorithm." *Applied mathematics and computation* 214, no. 1 (2009): 108-132.

Karaboga, Dervis, and Bahriye Basturk. "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems." In *International fuzzy systems association world congress*, pp. 789-798. Springer, Berlin, Heidelberg, 2007.

Karaboga, Dervis, and Bahriye Basturk. "On the performance of artificial bee colony (ABC) algorithm." *Applied soft computing* 8, no. 1 (2008): 687-697.

Karaboga, Dervis, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. "A comprehensive survey: artificial bee colony (ABC) algorithm and applications." *Artificial intelligence review* 42, no. 1 (2014): 21-57.

Karaboga, Dervis. *An idea based on honey bee swarm for numerical optimization*. Vol. 200. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

Karunanithi, Arunprakash T., Luke EK Achenie, and Rafiqul Gani. "A new decomposition-based computer-aided molecular/mixture design methodology for the design of optimal solvents and solvent mixtures." *Industrial & engineering chemistry research* 44, no. 13 (2005): 4785-4797.

Kaur, Sandeep, Ganesh Kumbhar, and Jaydev Sharma. "A MINLP technique for optimal placement of multiple DG units in distribution systems." *International journal of electrical power & energy systems* 63 (2014): 609-617.

Kier, Lemont B., Wallace J. Murray, Milan Randić, and Lowell H. Hall. "Molecular connectivity V: connectivity series concept applied to density." *Journal of pharmaceutical sciences* 65, no. 8 (1976): 1226-1230.

Kier, Lemont. *Molecular connectivity in chemistry and drug research*. Vol. 14. Elsevier, 2012.

Lowy, Franklin D. "Staphylococcus aureus infections." *New England journal of medicine* 339, no. 8 (1998): 520-532.

Mauri, Andrea, Viviana Consonni, Manuela Pavan, and Roberto Todeschini. "Dragon software: An easy approach to molecular descriptor calculations." *Match* 56, no. 2 (2006): 237-248.

McCaffrey, C., A. Bertasso, J. Pace, and N. H. Georgopapadakou. "Quinolone accumulation in Escherichia coli, Pseudomonas aeruginosa, and Staphylococcus aureus." *Antimicrobial agents and chemotherapy* 36, no. 8 (1992): 1601-1605.

McLeese, Samantha E., John C. Eslick, Nicholas J. Hoffmann, Aaron M. Scurto, and Kyle V. Camarda. "Design of ionic liquids via computational molecular design." *Computers & chemical engineering* 34, no. 9 (2010): 1476-1480.

Mendivii, Franklin, R. Shonkwiler, and M. C. Spruill. "Optimization by stochastic methods." (1999).

Morris, Carla C., and Robert M. Stark. Finite Mathematics: Models and Applications. John Wiley & Sons, 2015.

Nikaido, H., and D. G. Thanassi. "Penetration of lipophilic agents with multiple protonation sites into bacterial cells: tetracyclines and fluoroquinolones as examples." *Antimicrobial agents and chemotherapy* 37, no. 7 (1993): 1393.

O'Boyle, Noel M., Chris Morley, and Geoffrey R. Hutchison. "Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit." *Chemistry central journal* 2, no. 1 (2008): 5.

O'Boyle, Noel M., Michael Banck, Craig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. "Open Babel: An open chemical toolbox." *Journal of cheminformatics* 3, no. 1 (2011): 33.

Pérez, Miguel Angel Cabrera, Humberto González Dıaz, Carlos Fernández Teruel, José Ma Plá-Delfina, and Marival Bermejo Sanz. "A novel approach to determining physicochemical and absorption properties of 6-fluoroquinolone derivatives: experimental assessment." *European journal of pharmaceutics and biopharmaceutics* 53, no. 3 (2002): 317-325.

Petitjean, Michel. "Applications of the radius-diameter diagram to the classification of topological and geometrical shapes of chemical compounds." *Journal of chemical information and computer sciences* 32, no. 4 (1992): 331-337.

Piddock, Laura JV, Yu Fang Jin, and Deborah J. Griggs. "Effect of hydrophobicity and molecular mass on the accumulation of fluoroquinolones by Staphylococcus aureus." *Journal of antimicrobial chemotherapy* 47, no. 3 (2001): 261-270.

Raman, V. Shankar, and Costas D. Maranas. "Optimization in product design with properties correlated with topological indices." *Computers & chemical engineering* 22, no. 6 (1998): 747-763.

Randic, Milan. "Characterization of molecular branching." *Journal of the American Chemical Society* 97, no. 23 (1975): 6609-6615.

Sadowski, Jens, Johann Gasteiger, and Gerhard Klebe. "Comparison of automatic three-dimensional model builders using 639 X-ray structures." *Journal of chemical information and computer sciences* 34, no. 4 (1994): 1000-1008.

Schneider, Gisbert, and Uli Fechner. "Computer-based de novo design of drug-like molecules." *Nature reviews drug discovery* 4, no. 8 (2005): 649.

Schuur, Jan H., Paul Selzer, and Johann Gasteiger. "The coding of the three-dimensional structure of molecules by molecular transforms and its application to structure-spectra correlations and studies of biological activity." *Journal of chemical information and computer sciences* 36, no. 2 (1996): 334-344.

Siddhaye, Sachin, Kyle V. Camarda, Marylee Southard, and Elizabeth Topp. "Pharmaceutical product design using combinatorial optimization." *Computers & chemical engineering* 28, no. 3 (2004): 425-434.

Siddhaye, Sachin, Kyle V. Camarda, Elizabeth Topp, and Marylee Southard. "Design of novel pharmaceutical products via combinatorial optimization." *Computers & chemical engineering* 24, no. 2-7 (2000): 701-704.

Sinha, Manish, Luke EK Achenie, and Gennadi M. Ostrovsky. "Environmentally benign solvent design by global optimization." *Computers & chemical engineering* 23, no. 10 (1999): 1381-1394.

Soltzberg, Leonard J., and Charles L. Wilkins. "Molecular transforms: a potential tool for structure-activity studies." *Journal of the American Chemical Society* 99, no. 2 (1977): 439-443.

Team, R. Core. "R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2016." (2017): 860-864.

Terfloth, Lothar, and Johann Gasteiger. "10 Calculation of Structure Descriptors." *Chemoinformatics: Basic Concepts and Methods* (2018): 349.

Tetko, Igor V., Johann Gasteiger, Roberto Todeschini, Andrea Mauri, David Livingstone, Peter Ertl, Vladimir A. Palyulin et al. "Virtual computational chemistry laboratory–design and description." *Journal of computer-aided molecular design* 19, no. 6 (2005): 453-463.

Weininger, David. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules." *Journal of chemical information and computer sciences* 28, no. 1 (1988): 31-36.

Wierl, Ro. "Elektronenbeugung und Molekülbau." *Annalen der Physik* 400, no. 5 (1931): 521-564.

World Health Organization, Global priority list of antibiotic-resistant bacteria to guide research, discovery, and development of new antibiotics, Internal Report, (2017)

Yoshida, H., M. Bogaki, S. Nakamura, K. Ubukata, and M. Konno. "Nucleotide sequence and characterization of the Staphylococcus aureus norA gene, which confers resistance to quinolones." *Journal of bacteriology* 172, no. 12 (1990): 6942-6949.

Appendix

Python code for implementing the ABC algorithm

A Python code has been written to solve the MINLP formulated in Chapter Four. The ABC algorithm is implemented in the python code to obtain results for MINLP, i.e. for obtaining molecular structures possessing target property values. Openbabel and Pybel are installed within the python environment to generate 3D coordinates of molecules. The code has three main sections: employed bees’ loop, onlooker bees’ loop and selection of scout bees after certain number of iterations. Every solution is assessed in terms of how close it is to the target property values. As the algorithm progresses, the difference between target and predicted properties decreases, which eventually leads to a near optimal solution. The code is given below.

```
import random

import math

import openbabel

import pybel

import numpy


# all the molecules will be represented in SMILES notation and will be saved in the form of a list
of all the characters

# for every group



smiles = 'C1=C(C(=C3C2=C1C(C(=CN2[C@@H](C)CO3)C(O)=O)=O)C4(CC4)N)F'
```

```
# template molecule smiles notation



switchable_groups = [30,32,33,38,49,50,53,55]

# corresponding index in smiles string


val_one = [55, 53, 30, 38]

val_two = [49, 50, 33, 32]



val_one_groups = ['F', 'N', 'C', 'O', 'CN', 'CO', 'CCN', 'CCO', 'CCCN', 'CCCO']

val_two_groups = ['C', 'O']


template_sequence = list(smiles)

print template_sequence

print '\n the number of characters are ', len(template_sequence)


from copy import deepcopy


'''ABC Settings'''


MaxIt = 20

# Maximum Number of Iterations
```

```
n_employed = 50

# number of employed bees


n_onlooker = 20

# number of onlooker bees




target_prop_mic = 6

#target property value for -log(MIC)


target_prop_logp = 1.5

#target property value for log(P)


'''data structures for storing employed and onlooker bees'''

smi_em =[[] for zz in range(n_employed+1)]


smi_on =[[] for zz in range(n_onlooker+1)]




'''data structures for storing best solutions'''

best_solution_smiles = [[] for zz in range(MaxIt)]

### stores best smiles sequence
```

```
best_solution_prop = [[] for zz in range(MaxIt)]

### stores best property value


best_solution_prop_diff = [[] for zz in range(MaxIt)]

### stores difference between target and best prop value




'''Create Initial Population'''

'''here all the employed and onlooker bees' initial solutions are the same molecule as initialised for
template_sequence '''

print('Initial Values\n')


for i in range(1,n_employed+1):

    smi_em[i] = deepcopy(template_sequence)


for i in range(1,n_onlooker+1):

    smi_on[i] = deepcopy(template_sequence)




def get_coordinates(smi_string_molecule):
    """
    use this function as get_coordinates("CCC")
    where CCC is the smi notation of the molecule you want to work on
    """


    mol = pybel.readstring("smi", smi_string_molecule)

    mol.addh()
    mol.make3D()

    foo = mol.write("sdf")
    foo
    bar = foo.split("\n")[4:][:mol.OBMol.NumAtoms()]

    coordinates = map(lambda line: line.split()[:3], bar)
    return coordinates, mol.OBMol.NumAtoms()

def property_value(smile_string):

    var = get_coordinates(smile_string) # var stores [(xyz cordinates), number of atoms]
    coords = var[0]
    Num_atoms = var[1]
```

```
nn=Num_atoms

## nn = number of total groups/atoms


t=0




B=[[0 for x in range(4)] for y in range(nn+1)]

# B is x y z co-ordinates matrix


dist=[[0 for x in range(nn+1)] for y in range(nn+1)]

### matrix for storing interatomic distances


for i in range(1,nn+1):

    for j in range(1,4):

        B[i][j]=float(coords[i-1][j-1]) # converts string of numbers into actual number




sin_sum=0


for i in range(1,nn+1):

    for j in range(i+1,nn+1):
        sum=0
        for h in range(1,4):
            sum=math.pow((B[i][h]-B[j][h]),2)+sum
        r=math.sqrt(sum)
        dist[i][j]=r

MORSE=[0 for x in range(11)]
#stores Mor01u to Mor10u

MORSE[1]=math.factorial(nn)/(2*math.factorial(nn-2))
# calculation of Mor01u

for s in range(1,10):
    sum_sum=0
    for i in range(1,nn+1):
        for j in range(i+1,nn+1):

            sum_sum = sum_sum + math.sin(s*dist[i][j])/(s*dist[i][j])
            sum_sum = round(sum_sum,6)

    MORSE[s+1]=sum_sum
    temp_2 = -0.002*MORSE[1]+0.09*MORSE[2]-0.082*MORSE[3]-0.127*MORSE[4]-
0.634*MORSE[5]-0.413*MORSE[6]+0.19*MORSE[7]-0.581*MORSE[8]-0.237*MORSE[9]-
2.509
    ##Structure - property relationship for -log(MIC)


    temp_logp = 0.022*MORSE[1]+0.011*MORSE[2]+0.624*MORSE[3]-
0.41*MORSE[4]+2.312*MORSE[5]-2.227*MORSE[6]+0.497*MORSE[7]-
0.465*MORSE[8]+0.589*MORSE[9]-11.375
    ##Structure - property relationship for -log(MIC)


    objective_fun = abs(temp_2-target_prop_mic)+abs(temp_logp-target_prop_logp)
    ##objective function



    return objective_fun


least_value =[]
for it in range(1,MaxIt+1):
    print '\n This is iteration=  ',it
    property_em = []
```

```
property_em_diff = [] ### difference with target property values

property_on = []

property_on_diff =[]


'''  ********    employed bees loop     *************'''


print '\n Employed Bees \n'

for e in range(1,n_employed+1):


    old_em = deepcopy(smi_em[e])

    l = ''.join(old_em)

    old_employed = deepcopy(l)

    old_property_em = property_value(old_employed)




    k = random.choice(switchable_groups)

    #selection of a position for replacing a group


    if k in val_one:


        u = random.choice(val_one_groups)

        # selection of a valency one group
    smi_em[e][k] = u

    # smile notation change with new group at corresponding smiles string position



if k in val_two:


    u = random.choice(val_two_groups)

    # selection of a valency two group


    smi_em[e][k] = u

    # smile notation change with new group at corresponding smiles string position



new_string_em = ''.join(smi_em[e])

### joining all the characters to form a string



R = deepcopy(new_string_em)


new_property_em = property_value(R)

## calculation of objective function
```

```
    ### greedy selection criteria   #####
    if new_property_em >= old_property_em:
        smi_em[e] = deepcopy(old_em)
        property_em.append(old_property_em)

    else:
        property_em.append(new_property_em)



old_property_values = deepcopy(property_em)
sorted_old_values = deepcopy(sorted(property_em))

print 'this is prop diff list ', property_em

index = []

######################### ranking bees in this iteration ##########################
index_two = []
for item in sorted_old_values:
    for ind, value in enumerate(old_property_values):
        if (item == value):
            #print("******************This is Working************")
            index_two.append(ind+1)

            break

        else:

            continue



################################################################################
###############

  index = index_two[0:5]

  ### these are best five employed bees


  current_best = index_two[0]

  ### this is best employed bee in the current iteration


  current_worst = index_two[-1]

  ### this is worst employed bee in the current iteration


  current_second_worst = index_two[-2]

   ### this is second worst employed bee in the current iteration


  best_employed_sequence = smi_em[current_best]

  best_employed_smiles = ''.join(smi_em[current_best])

  best_employed_prop = property_value(best_employed_smiles)
```

```
    '''This is the END of Employed Bees' Loop '''





''' ONLOOKER BEES LOOP Starts here'''


for on in range(1,n_onlooker+1):



    old_on = deepcopy(smi_on[on])
    ###previous onlooker bee sequence


    old_onlooker = ''.join(old_on)
    ###joining the characters to form SMILES string


    old_property_on = property_value(old_onlooker)
    ###finding the value of objective function for old bee


    onlook = random.choice(index)
```

```
##randomly selecting one bee out of good employed bees


now = ''.join(smi_em[onlook])

smi_on[on] = deepcopy(smi_em[onlook])




k = random.choice(switchable_groups)

###random selection of one group for replacement


if k in val_one:


    u = random.choice(val_one_groups)

    # selection of a valency one group


    smi_on[on][k] = u

    # smile notation change with new group at corresponding smiles string position


if k in val_two:


    u = random.choice(val_two_groups)

    # selection of a valency two group


    smi_on[on][k] = u
```

```
        # smile notation change with new group at corresponding smiles string position



    new_string_on = ''.join(smi_on[on])



    R = deepcopy(new_string_on)

    new_property_on = property_value(R)



    ##### greedy selection criteria   #######

    if new_property_on >= old_property_on:
        smi_on[on] = deepcopy(old_on)
        property_on.append(old_property_on)

    else:
        property_on.append(new_property_on)



least = min(property_on)
###closest property value to target value in this iteration
```

```
least_indexx = (property_on).index(least)

#least_indexx = (property_on_diff).index(least)


least_value.append(least)

#print("\n\n\n least  diff value and corresponding index is  \n\n\n", least, least_indexx)


best_onlooker_sequence = smi_on[least_indexx+1]

best_onlooker_smiles = ''.join(smi_on[least_indexx+1])

best_onlooker_prop = property_on[least_indexx]




### choosing best solution in this iteration ####

if best_employed_prop <= best_onlooker_prop:

   print '\n this is best soltution in this iteration ', best_employed_smiles

   print '\n this is the corresponding prop ', best_employed_prop


else:

   print '\n this is best soltution in this iteration ', best_onlooker_smiles

   print '\n this is the corresponding prop ', best_onlooker_prop


### choosing scout bees ######

if it%10 == 0:
```

```
smi_em[current_worst] = deepcopy(best_onlooker_sequence)
smi_em[current_second_worst] = deepcopy(template_sequence)


##these two are scout bees, worst two employed bees solutions will be changed after 10 iterations
```