M.Sc. Engg. Thesis

Meta-heuristic Algorithms and Experimental Analysis for the Unique Restriction Site Placement Problem

by

Rashedul Hasan

MASTER OF SCIENCE IN INFORMATION AND COMMUNICATION TECHNOLOGY

Institute of Information and Communication Technology (IICT)

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY (BUET)
Dhaka 1000

June 2013

The thesis titled **"**Meta-heuristic Algorithms and Experimental Analysis for the Unique Restriction Site Placement Problem**"** submitted by Rashedul Hasan, Roll No. M10073106P, Session: October 2007, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Information and Communication Technology on 26th June 2013.

BOARD OF EXAMINERS

1. _____

Dr. M. Sohel Rahman
Associate Professor
Department of CSE, BUET, Dhaka.
(Supervisor)

Chairman

2. _____

Director
IICT, BUET, Dhaka.

Member
(Ex-officio)

3. _____

Dr. Md. Liakot Ali
Professor
IICT, BUET, Dhaka.

Member

4. _____

Dr. Mohammad Shah Alam
Assistant Professor
IICT, BUET, Dhaka.

Member

5. _____

Dr. Mohammad Nurul Huda
Professor
Department of CSE
United International University (UIU), Dhaka.

Member
(External)

Declaration of Authorship

This is to certify that the work presented in this thesis entitled "Meta-heuristic Algorithms and Experimental Analysis for the Unique Restriction Site Placement Problem" is the outcome of the investigation carried out by us under the supervision of Dr. M. Sohel Rahman, Associate Professor, Department of Computer Science and Engineering (BUET), Dhaka. It is also declared that neither this thesis nor any part thereof has been submitted or being currently submitted anywhere else for the award of any degree or diploma.

_____

(Author)
Rashedul Hasan
Roll No.: M10073106P
Institute of Information and Communication Technology (BUET),
Dhaka-1000

ACKNOWLEDGMENT

I would like to express my deep and sincere gratitude to my supervisor Dr. M. Sohel Rahman, Associate Professor, Computer Science and Engineering (CSE), Bangladesh University of Engineering and Technology (BUET), Dhaka. I owe to him for his constant supervision, encouragement, personal guidance during the progress of my thesis. His in-depth knowledge in Meta-heuristics and Bio-informatics computing and his logical way of thinking have been helpful for the successful completion of this work. I am grateful to him for his cooperation.

I would like to thank the members of my thesis examination committee for their patience in understanding my work. I warmly thank professors Dr. Md. Saiful Islam, Dr. Md. Laikot Ali, Dr. Mohammad Shah Alam and Dr. Mohammad Nurul Huda for their valuable suggestions. Specially, I would like to thank Professor Dr. S. M. Lutful Kabir for his special guidance. I want to give thanks to Dr. Siblee Sadeque Shakil for helping me from his rich biological background. I am thankful to all my teachers, colleagues and friends for their supports during the period of my thesis.

Finally, I owe my thanks to my parents who have encouraged me continuously from the beginning of my thesis work. I would like to convey my thanks to my wife, my little sisters and relatives. Without their encouragement it would have been impossible for me to finish this work.

Above all, I am grateful to Almighty Allah who gave me the strength to complete this work successfully.

CONTENTS

BOARD OF EXAMINERS ii

DECLARATION OF AUTHORSHIP iii

ACKNOWLEDGMENT iv

LIST OF TABLES viii

LIST OF FIGURES xii

ABSTRACT xiv

1. Introduction 1
   - 1.1 Definition of Bioinformatics 1
   - 1.2 History and Prospect of Bioinformatics 2
   - 1.3 Synthetic Biology 4
     - 1.3.1 Definition 4
     - 1.3.2 Computer Science and Synthetic Biology 5
   - 1.4 Literature survey 5
     - 1.4.1 Problem Statement 5
     - 1.4.2 Previous Results on URSPP 6
     - 1.4.3 Related Works on Gene Synthesis 8
   - 1.5 Objectives of the Thesis 10
   - 1.6 The Contribution of this Thesis 10
   - 1.7 Organization of Thesis 12

2. Background 13
   - 2.1 Synthetic Biology 13
     - 2.1.1 Viral Genome Synthesis 13
     - 2.1.2 Refactoring: Genome Vs. Software 14
     - 2.1.3 Restriction Enzymes and Restriction Sites 14
     - 2.1.4 Protein and Amino Acid 15
     - 2.1.5 Genetic Code: Codon 15
     - 2.1.6 Subcloning 16
   - 2.2 Meta-heuristics 17

2.2.1 Definition of Meta-heuristics . . . 17
2.2.2 Single State Methods . . . 18
2.2.2.1 Hill Climbing . . . 19
2.2.2.2 Steepest Ascent Hill Climbing . . . 19
2.2.3 Population Methods . . . 19
2.2.3.1 Genetic Algorithms . . . 20
2.2.4 Selection Procedure . . . 21
2.2.5 Multiobjective Optimization . . . 22
2.2.5.1 Non-Dominated Sorting Genetic Algorithm . . . 22
2.3 Summary . . . 24
3. URSPP in Synthetic Genomes by Elitists Meta-heuristics . . . 25
3.1 Motivation behind application of Eliitist Meta-heuristics . . . 25
3.2 Motivation behind change of an objective . . . 25
3.3 Proposed Methodology . . . 27
3.3.1 Candidate Solution Representation . . . 27
3.3.2 Elitist Meta-heuristics . . . 28
3.3.3 Breeding Operators . . . 30
3.3.4 Standard Deviation Instead of Maximum Gap . . . 31
3.3.5 Quality Assessment . . . 32
3.4 Algorithms . . . 32
3.4.1 Hill Climbing . . . 32
3.4.2 Steepest Ascend Hill Climbing . . . 33
3.4.3 Genetic Algorithm . . . 33
3.4.4 Non-Dominated Sorting Genetic Algorithm . . . 33
3.5 Overall Elitist Approach . . . 34
3.6 Other Multiobjective Optimazation Technique . . . 36
3.6.1 Weighted Sum Model . . . 36
3.6.2 Goal Programming . . . 37
3.7 Summary . . . 38
4. Experimental Results . . . 39
4.1 Statistical Test . . . 39
4.1.1 The T Test . . . 39
4.1.2 Paired or Unpaired Test . . . 40

4.1.3 Concept of Null Hypothesis 40
4.1.4 P Value 41
4.1.5 Confidence Interval (C.I.) 41
4.2 Simulation Results 42
4.2.1 Experimental set up and Representation 42
4.2.2 Results Summary and Analysis 45
4.3 Summary 89
5. Software 90
5.1 Programming Environment 90
5.2 Prerequisite Files 90
5.3 Description of the Software 94
5.4 Summary 98
6. Conclusion 99
6.1 Summary of Contribution 100
6.2 Future Research 101
REFERENCES 102

LIST OF TABLES

2.1 Amino Acid Descriptions 16

3.1 Two Different Solutions - 1 26

3.2 Two Different Solutions - 2 26

3.3 One Candidate Solution 28

4.1 List of Viral Sequences 43

4.2 Objective set 1 43

4.3 Objective Set 2 43

4.4 Algorithms Names 44

4.5 Comparison of NSGA-(HC) for Polio virus Considering Objective Set 1 . . 46

4.6 Comparison of NSGA-(SAHC) for Polio virus Considering Objective Set 1 . 46

4.7 Comparison of NSGA-(GA) for Polio virus Considering Objective Set 1 . . 47

4.8 Comparison of NSGA-(HC) for Polio virus Considering Objective Set 2 . . 47

4.9 Comparison of NSGA-(SAHC) for Polio virus Considering Objective Set 2 . 48

4.10 Comparison of NSGA-(GA) for Polio virus Considering Objective Set 2 . . 48

4.11 Comparison of NSGA-(HC) for Rubella virus Considering Objective Set 1 . 49

4.12 Comparison of NSGA-(SAHC) for Rubella virus Considering Objective Set 1 50

4.13 Comparison of NSGA-(GA) for Rubella virus Considering Objective Set 1 . 50

4.14 Comparison of NSGA-(HC) for Rubella virus Considering Objective Set 2 . 51

4.15 Comparison of NSGA-(SAHC) for Rubella virus Considering Objective Set 2 51

4.16 Comparison of NSGA-(GA) for Rubella virus Considering Objective Set 2 . 52

4.17 Comparison of NSGA-(HC) for Papilloma virus Considering Objective Set 1 53

4.18 Comparison of NSGA-(SAHC) for Papilloma virus Considering Objective Set 1 53

4.19 Comparison of NSGA-(GA) for Papilloma virus Considering Objective Set 1 54

4.20 Comparison of NSGA-(HC) for Papilloma virus Considering Objective Set 2 54
4.21 Comparison of NSGA-(SAHC) for Papilloma virus Considering Objective Set 2 55
4.22 Comparison of NSGA-(GA) for Papilloma virus Considering Objective Set 2 55
4.23 Comparison of NSGA-(HC) for Mosaic virus Considering Objective Set 1 . 56
4.24 Comparison of NSGA-(SAHC) for Mosaic virus Considering Objective Set 1 56
4.25 Comparison of NSGA-(GA) for Mosaic virus Considering Objective Set 1 . 57
4.26 Comparison of NSGA-(HC) for Mosaic virus Considering Objective Set 2 . 57
4.27 Comparison of NSGA-(SAHC) for Mosaic virus Considering Objective Set 2 58
4.28 Comparison of NSGA-(GA) for Mosaic virus Considering Objective Set 2 . 58
4.29 Comparison of NSGA-(HC) for HIV virus Considering Objective Set 1 . . . 59
4.30 Comparison of NSGA-(SAHC) for HIV virus Considering Objective Set 1 . 59
4.31 Comparison of NSGA-(GA) for HIV virus Considering Objective Set 1 . . . 60
4.32 Comparison of NSGA-(HC) for HIV virus Considering Objective Set 2 . . . 60
4.33 Comparison of NSGA-(SAHC) for HIV virus Considering Objective Set 2 . 61
4.34 Comparison of NSGA-(GA) for HIV virus Considering Objective Set 2 . . . 61
4.35 Comparison of NSGA-(HC) for Hepatitis C virus Considering Objective Set 1 62
4.36 Comparison of NSGA-(SAHC) for Hepatitis C virus Considering Objective Set 1 62
4.37 Comparison of NSGA-(GA) for Hepatitis C virus Considering Objective Set 1 63
4.38 Comparison of NSGA-(HC) for Hepatitis C virus Considering Objective Set 2 64
4.39 Comparison of NSGA-(SAHC) for Hepatitis C virus Considering Objective Set 2 64
4.40 Comparison of NSGA-(GA) for Hepatitis C virus Considering Objective Set 2 65
4.41 Comparison of NSGA-(HC) for Bronchitis virus Considering Objective Set 1 66
4.42 Comparison of NSGA-(SAHC) for Bronchitis virus Considering Objective Set 1 67
4.43 Comparison of NSGA-(GA) for Bronchitis virus Considering Objective Set 1 67

4.44 Comparison of NSGA-(HC) for Bronchitis virus Considering Objective Set 2 68

4.45 Comparison of NSGA-(SAHC) for Bronchitis virus Considering Objective Set 2 68

4.46 Comparison of NSGA-(GA) for Bronchitis virus Considering Objective Set 2 69

4.47 Comparison of NSGA-(HC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 1 69

4.48 Comparison of NSGA-(SAHC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 1 70

4.49 Comparison of NSGA-(GA) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 1 70

4.50 Comparison of NSGA-(HC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 2 71

4.51 Comparison of NSGA-(SAHC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 2 71

4.52 Comparison of NSGA-(GA) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 2 72

4.53 Comparison of NSGA-(HC) for Shrimp White Spot Syndrome virus Considering Objective Set 1 72

4.54 Comparison of NSGA-(SAHC) for Shrimp White Spot Syndrome virus Considering Objective Set 1 73

4.55 Comparison of NSGA-(GA) for Shrimp White Spot Syndrome virus Considering Objective Set 1 73

4.56 Comparison of NSGA-(HC) for Shrimp White Spot Syndrome virus Considering Objective Set 2 74

4.57 Comparison of NSGA-(SAHC) for Shrimp White Spot Syndrome virus Considering Objective Set 2 74

4.58 Comparison of NSGA-(GA) for Shrimp White Spot Syndrome virus Considering Objective Set 2 75

4.59 Comparison of Weighted Sum Methods for Polio Virus Considering Objective Set 1 75

4.60 Comparison of Goal Programming for Polio Virus Considering Objective Set 2 76

4.61 Results of proposed Algorithms for Polio Virus 78
4.62 Results of proposed Algorithms for Rubella Virus 79
4.63 Results of proposed Algorithms for Papilloma Virus 79
4.64 Results of proposed Algorithms for Mosaic Virus 80
4.65 Results of proposed Algorithms for HIV Virus 80
4.66 Results of proposed Algorithms for Hepatitis C Virus 81
4.67 Results of proposed Algorithms for Bronchitis Virus 81
4.68 Results of proposed Algorithms for Glossina Pallidipes Salivary Gland Hypertrophy Virus 82
4.69 Results of proposed Algorithms for Shrimp White Spot Syndrome Virus . . 82
5.1 Algorithms incorporate in software tool 95

LIST OF FIGURES

2.1 Insertion of a restriction site. 17

3.1 Parts of the Cutting Sequences. 27

4.1 95% Confidence Interval. 42

4.2 Comparison Considering Objective Set 1 for HIV Virus in terms of f_1 (higher is preferred) 78

4.3 Comparison Considering Objective Set 1 for HIV Virus in terms of f_2 (lower is preferred) 83

4.4 Comparison Considering Objective Set 1 for HIV Virus in terms of f_3 (lower is preferred) 83

4.5 Comparison Considering Objective Set 2 for HIV Virus in terms of f_1 (higher is preferred) 84

4.6 Comparison Considering Objective Set 2 for HIV Virus in terms of f_2 (lower is preferred) 84

4.7 Comparison Considering Objective Set 2 for HIV Virus in terms of f_4 (lower is preferred) 85

4.8 Comparison Considering Objective Set 1 for Shrimp White Spot Syndrome Virus in terms of f_1 (higher is preferred) 85

4.9 Comparison Considering Objective Set 1 for Shrimp White Spot Syndrome Virus in terms of f_2 (lower is preferred) 86

4.10 Comparison Considering Objective Set 1 for Shrimp White Spot Syndrome Virus in terms of f_3 (lower is preferred) 86

4.11 Comparison Considering Objective Set 2 for Shrimp White Spot Syndrome Virus in terms of f_1 (higher is preferred) 87

4.12 Comparison Considering Objective Set 2 for Shrimp White Spot Syndrome Virus in terms of f_2 (lower is preferred) 87

4.13 Comparison Considering Objective Set 2 for Shrimp White Spot Syndrome Virus in terms of f_4 (lower is preferred) 88

5.1 RE.txt file. 91

5.2 sequence.txt file. 92

5.3 locked.txt file. 92
5.4 genes.txt file. 93
5.5 Software Tool. 94
5.6 Approaches in Software 95
5.7 Tuning Parameters. 95
5.8 Objective Sets. 96
5.9 Result 96
5.10 output.txt file. 97
5.11 View Sequence 97

ABSTRACT

Genome synthesis has become an important tool in many fields of recombinant DNA technology including vaccine development, gene therapy and molecular engineering. There had been a number of researches that investigated the function of genes in a sequence and how to synthesize genome sequence according to user specification. The purpose of this study is to find a genome sequence which provides maximum amount of flexibility and independence to biologists to run experiment with the sequence. *Restriction enzymes* are reagents widely used by molecular biologists for genetic manipulation and analysis. The decision version of *Unique Restriction Site Placement Problem (URSPP)* has been proved to be NP complete. Since URSPP is a *multi-objective optimization problem* and multi-objective optimization often shows good performance through *elitist approach*, our target is to introduce elitism in different *meta-heuristic strategies* to solve URSPP. Additionally, we change one of the objectives of URSPP to make the problem biologically more interesting and meaningful. We carry out extensive experiments considering different clinically important viruses as well as large viral sequences considering existing meta-heuristic algorithms in the literature as well as our newly developed algorithms. Our experimental results show that our approaches give better results than traditional meta-heuristics. As a result, we provide a software tool according to our approach which would be helpful for genome synthesis from biological viewpoint.

CHAPTER 1
Introduction

It is undeniable that, among the branches of science, biology has played a key role during this century. That role is likely to acquire further importance in the years to come. The huge volume of biological data makes it impossible to analyze them manually. As a result, computers have become indispensable to biological research. This chapter introduces the reader with the *bioinformatics* research and provides an overview of what we have done in our thesis. In particular, we give an overview of bioinformatics in general and of a popular branch thereof namely, *synthetic biology* in specific. It is the latter sub discipline of bioinformatics, which the subject of this thesis belongs to.

1.1 Definition of Bioinformatics

Paulien Hogeweg, a Dutch theoretical biologist and complex systems researcher, with her colleague, Ben Hesper, coined the term Bioinformatics in 1978 [1, 2] as the study of informatic processes in biotic systems. Now-a-days, it has become a very popular "buzz" word in science. As submitted to the Oxford English Dictionary, Bioinformatics is conceptualising biology in terms of molecules (in the sense of Physical Chemistry) and applying "informatics techniques" (derived from disciplines such as Applied Mathematics, Computer Science and Statistics) to understand and organise the information associated with these molecules, on a large scale. In short, Bioinformatics is a management information system for Molecular Biology and has many practical applications.

Neil C. Jones and Pavel Pevzner in their book 'Introduction to Bioinformatics Algorithms' [3], defined bioinformatics as follows:

> "Bioinformatics is a computational science. Bioinformatics merges the two fields, and adds a healthy dose of statistics, combinatorics, and other branches of mathematics."

In the book 'Bioinformatics Introduction' [4], Mark Gerstein referred to Bioinformatics as a combination of biological data and computer calculations. Bioinformatics is a

grammatical contraction of "*Biological Informatics*" and is therefore related to the computer science disciplines of information science or information technology [5]. Bioinformatics has grown from a need to manage and analyze several forms of biological data: 1) DNA sequences; 2) Protein sequences; 3) Protein/RNA structures; 4) Gene expression; The primary goal of Bioinformatics is to increase the understanding of biological processes. What sets it apart from other approaches, however, is its focus on developing and applying computationally intensive techniques to achieve this goal. For example, pattern recognition, data mining, machine learning algorithms, and visualization. Major research efforts in the field include sequence alignment, gene finding, genome assembly, drug design, drug discovery, protein structure alignment, protein structure prediction, prediction of gene expression and protein protein interactions, genome-wide association studies, and the modeling of evolution [6].

1.2 History and Prospect of Bioinformatics

In 1968, the beginning of bioinformatics can be traced back to Margaret Dayhoff's collection of protein sequences known as the Atlas of Protein Sequence and Structure [7]. The application of a sequence similarity searching program to the identification of the origins of a viral gene was one of the early significant experiments in bioinformatics [8]. In this study, scientists used one of the first sequence similarity searching computer programs (called FASTP), to determine that the contents of v-sis, a cancer causing viral sequence, were most similar to the well-characterized cellular PDGF gene. This surprising result provided important mechanistic insights for biologists working on how this viral sequence causes cancer [9]. Since this first initial application of computers to biology, the field of bioinformatics has exploded. The growth of bioinformatics is parallel to the development of DNA sequencing technology. In the same way that the development of the microscope in the late 1600s revolutionized biological sciences by allowing Anton Van Leeuwenhoek to look at cells for the first time, DNA sequencing technology has revolutionized the field of bioinformatics. The rapid growth of bioinformatics can be illustrated by the growth of DNA sequences contained in the public repository of nucleotide sequences called GenBank.

Genome sequencing projects have become the flagships of many bioinformatics

initiatives. In 1988, the Human Genome organization (HUGO) was founded. The first complete genome map was published of bacteria *Haemophilus Influenza* [10]. In 1990, the Human Genome Project was started. By 1991, a total of 1879 human genes had been mapped. In France, in 1993, Genethon, a human genome research center produced a physical map of the human genome. Three years later , Genethon published the final version of the human genetic map. This concluded the end of the first phase of the Human Genome Project [11].

Bioinformatics was fuelled by the need to create huge databases (such as Genbank, EMBL) to store and compare the DNA sequence data erupting from the human genome and other genome sequencing projects. It enables researchers to analyze the terabytes of data being produced by the Human Genome Project. Gene sequence databases and related analysis tools all help scientists to determine whether and how a particular molecule is directly involved in a disease process. That in turn, helps them find new and better drug targets. Bioinformatics can be thought of as a central hub that unites several disciplines and methodologies - molecular biology; information technology/information management; applications/databases; computational resources; CADD (Computer Aided Drug Design); and Genomics/Proteomics/x-omics .

Computer-Aided Drug Design (CADD) is a specialized discipline that uses computational methods to simulate drug-receptor interactions. CADD methods are heavily dependent on bioinformatics tools, applications and databases. As such, there is considerable overlap in CADD research and bioinformatics [12].

The same way developments in microscopy foreshadowed discoveries in cell biology, new discoveries in information technology and molecular biology are foreshadowing discoveries in bioinformatics. In fact, an important part of the field of bioinformatics is the development of new technology that enables the science of bioinformatics to proceed at a very fast pace. On the computer side, the Internet, new software developments, new algorithms, and the development of computer cluster technology have enabled bioinformatics to make great leaps in terms of the amount of data which can be efficiently analyzed. On the laboratory side, new technologies and methods such as DNA sequencing, serial analysis of gene expression (SAGE), microarrays, and new mass spectrometry chemistries have developed at an equally blistering pace enabling scientists to produce data for analyses at

an incredible rate. Bioinformatics provides both the platform technologies that enable scientists to deal with the large amounts of data produced through genomics and proteomics initiatives as well as the approach to interpret these data. In many ways, bioinformatics provides the tools for applying scientific method to large-scale data and should be seen as a scientific approach for asking many new and different types of biological questions.

1.3 Synthetic Biology

In this thesis, we particularly focus on a special branch of bioinformatics known as Synthetic Biology. This section gives a brief introduction to this branch.

1.3.1 Definition

Synthetic biology is an emerging and exciting field in bioinformatics and genetic engineering. For several decades, genetic modification technologies have been used to move genes from one species and splice them into the DNA of another. Here the ultimate goal is to create transgenic plants/animals/microorganisms with new and improved characteristics.

Report of a NEST[1] high-level expert group of European Commission defined Synthetic Biology as follows [13],

> "Synthetic Biology is the *Engineering of Biology*. It focuses on the synthesis of complex, biologically based (or inspired) systems which display functions that do not exist in nature. This engineering perspective may be applied at all levels of the hierarchy of biological structures (from individual molecules to whole cells, tissues and organisms). In essence, synthetic biology will enable the design of 'biological systems' in a rational and systematic way."

The Royal Society of United Kingdom defined Synthetic Biology as follows [14],

> "Synthetic biology is an emerging area of research that can broadly be described as the design and construction of novel artificial biological pathways, organisms or devices, or the redesign of existing natural biological systems."

[1] New and Energing Science and Technology (NEST) is a research activity under the European Communitys 6th Framework Programme

1.3.2 Computer Science and Synthetic Biology

In simple words, Synthetic Biology is nothing but putting Engineering into Biology. Tim Gardner and Jim Collins developed an engineered genetic toggle switch[2] which is a good example of how engineering principles are driving the boat of synthetic biology [15]. Researchers are now trying to adapt concepts developed in area of programming language development and software engineering for synthetic biology applications. For example, a recent paper in PLoS Computational Biology shows how methods used by computer scientists to develop programming languages can be applied to DNA sequences [16]. Authors in [16] report an attribute grammar based formalism to model the structure-function relationships in synthetic DNA sequences. An attribute grammar is constructed as an extension of a context-free grammar and in computer science it is commonly used to translate the text of a program source code or the syntax tree directly into the computational operations or machine level instructions.

Another example of the study and research under the hood of synthetic biology is *redesigning Bacteria*. Bacteria are the simplest known objects from the natural world that are capable of replicating when provided with only simpler components. Still, bacteria are far from simple. Bacteria also provide the basic environment in which synthetic biological systems exist and act; in some sense, they are like the power supply and chassis of a computer. By redesigning/refactoring a simple living system we hope to learn how to better couple (and decouple) our designed systems from their host environment.

1.4 Literature survey

The goal of this chapter is to present the problem definition and discuss the related literature. Here, in our discussion we include previous works on the original problem we tackle in this thesis as well as some other works related to gene synthesis.

1.4.1 Problem Statement

The problem of our interest originates from the laboratory of viral genome synthesis. The original problem takes as input a viral plasmid sequence and a set of restriction

[2]Genetic toggle switch is a synthetic, bistable gene regulatory network in an organism, e.g., Escherichia coli. The toggle is flipped between stable states using transient chemical or thermal induction and exhibits a nearly ideal switching threshold.

enzymes. The goal then is to find a new plasmid sequence containing unique recognition sites for the given set of restriction enzymes such that (1) the number of unique sites inserted is maximum, (2) amount of sequence editing required is as less as possible and (3) the placement of the sites are as even as possible. This problem is referred to as the Unique Restriction Site Placement Problem (URSPP) in the literature [17]. From combinatorial viewpoint the same problem can be stated as follows. Given a text and a set of patterns, find a new text with maximum number of patterns inserted, ensuring minimum editing of letters and that the maximum distance between two consecutive inserted patterns (considering all insertions) is minimum. This is the optimization version of URSPP. The decision version of the same problem, as defined in [17], asks whether it is possible to insert all patterns from a given set with gaps between patterns being at most K for a given integer, K.

While editing the sequence, some practical restrictions need to be considered. To retain the original functionality of the genome, its amino acid sequence must remain unaltered. Thats why, synonymous codons are used to introduce a site to and/or delete a site from the sequence through the change of one or more bases. During such deletions and insertions no accidental occurrence of any of the restriction sites should be introduced in the sequence. The difficulty of the problem, among others, arises from the task of keeping track of exponential number of possibilities in the order of considering restriction enzymes for insertion and/or deletion. We must also decide on which occurrence of a restriction site to keep so that the site serves as a unique one. This also adds to the difficulty.

Another restriction is that a site in the so called *locked region* can not be deleted, nor a site can be inserted there. Locked region is the part of a genome without which the virus dies/can never function as expected, though the actual function of such a region is still a mystery to the biologists.

1.4.2 Previous Results on URSPP

URSPP is a real life problem. This has originated from the laboratory while conducting viral genome synthesis. The computational version of the problem has come into into surface through the work of [17]. The decision version of URSPP problem has been

proved to be NP complete [17]. The optimization version cannot even be approximated within a factor of 3/2 unless P=NP [17]. The best known result for the optimization version is a 2-approximation algorithm [17]. Also a Dynamic Programming (DP) algorithm with an exponential running time of $O(n^2 2^r)$ and some heuristic algorithms have been proposed in [17]. Here, n is the number of events[3] and r is the number of unused restriction enzymes.

The authors in [17] provided two types of approximate implementations of the DP algorithm because of its exponential running time and memory requirement. In both of those the DP algorithm is run in turns considering consecutive blocks of enzymes one after another. In particular, at first the optimal placement is sought for the first block (set) of enzymes and then for the following block, and so on. However, in one implementation the order of considering the enzymes gives the highest possible insertion points (forward implementation) whereas the other gives the lowest possible insertion points (backward implementation).

The other heuristic versions are implemented first by eliminating all but one restriction site for each enzyme that appear in the genome initially and then inserting new restriction sites for enzymes which do not exist in the initial genome. To insert the enzymes they considered a greedy approach and the maximal bipartite matching technique. The greedy heuristic insertion selects unused restriction enzymes with least number of potential insertion points. The other approach, to insert enzymes, applies Hungarian algorithm [18] of Weighted Bipartite Matching on the weighted bipartite graph $G = (X \cup Y; E)$, where X is a set of unused restriction enzymes and Y is a list of ideal places where enzymes should be inserted. The weight of an edge $e \in E$ depends on the distance between an ideal place, $y \in Y$ and the location where $x \in X$ can be inserted.

Recently, some popular local search techniques like hill climbing and variations thereof have been applied to solve URSPP [19]. In [19], the representation of candidate solution and quality assessment function were proposed. For assessing fitness of a solution the objectives were converted to a single one by weighted sum. Simple Hill Climbing Algorithm had been used where candidate solutions were iteratively tested in the region of the current candidate and the new ones were adopted if they were better. Steepest As-

[3] An event corresponds to either a location where a restriction enzyme is currently cutting or a place where an unused restriction enzyme can be inserted [17].

cent Hill Climbing had also been used where it sampled all around the original candidate solutions and then picked the best one. Additionaly, Steepest Ascent Hill Climbing with Replacement which is allowed more exploration had been applied to solve URSPP. Experiments were conducted by some small viruses and significant optimization had been achieved comparing with the works of [17].

Very recently, in [20], Genetic Algorithms were applied to solve URSPP. The genetic algorithms considered in [20] were basic Genetic Algorithm and Non-Dominated Sorting Genetic Algorithm. Among these, basic GA was hybridized by a popular local improver, namely, hill-climbing. The versions used for Tournament Selection were, Multiobjective Lexicographic Tournament Selection (mlts), Multiobjective Majority Tournament Selection (mmts) and Multiobjective Ratio Tournament Selection (mrts). For breeding of new candidate solutions, both recombination/crossover and mutation were used. For recombination, standard two point crossover was applied. For mutation, problem specific point mutation was applied. For statistical significance test, along with the mean values and standard deviations of the objective value differences, the confidence interval (C.I.) around the mean value using a 95% confidence level and P Values are provided in [20]. It had been shown that Non-Dominated Sorting Genetic Algorithm (NSGA) was less costlier (in terms of base changes) than local search techniques keeping a very good maximum gap between the restriction sites. Notably, In [19, 20], small viruses like polio and alpha phase were experimented.

1.4.3 Related Works on Gene Synthesis

Although URSPP was introduced within some recent years, works on Gene Synthesis is not new in the literature. So, here we give a brief literature review on some of these works. Designing synthetic genes by hand is a time-consuming and error-prone process. In the past, a gene synthesis company used to take the requirements from the researchers to design genes using its own proprietary programs. Now-a-days these syntheses are relatively inexpensive commercialized services. It has become the most cost-effective, time and resource-saving method for obtaining nearly any desired DNA construct, outperforming conventional molecular biology techniques in many aspects from time and economization to expression performance, stability, and quality.

Blue Heron [21], OriGene [22], GeneArt [23] are some of the leading commercial vendors for Gene Synthesis. These vendors continue to innovate with technologies to meet the demands of the researchers. Also, a large number of tools and algorithms are being designed to provide a platform for synthetic gene design according to the user requirements.

Since 1999, Blue Heron has delivered tens of millions of base pairs of perfectly accurate genes to thousands of customers worldwide. Blue Heron can deliver to the customers one gene or one thousand according to the order from them. The options for such delivery ranges from the simplest sequence to comprehensive codon substitutions creating variants across hundreds of regions. Now-a-days Blue Heron is a part of OriGene.

OriGene Technologies was founded as a research tool company focused on the creation of the largest commercial collection of full-length human cDNAs[4] in a standard expression vector. OriGene Technologies uses high-throughput, genome wide approach to develop products for pharmaceutical, biotechnology, and academic research.

GeneJAX [24] is a JavaScript web application CAD tool for genome refactoring. It is used for the parts extraction and visualization stages of the genome re-designing/refactoring process [25]. At this point we note that, unlike the introduction of a single restriction site, the problem we study in this thesis tries to introduce as many restriction sites as possible as part of the redesign process of a whole genome sequence. GeneJax has been inspried by google maps. It can map a local region from a large data set and parts of the sequence can also be manipulated (e.g. created, deleted, renamed, exported).

SiteFind is a free web-based software tool. It enables the user to introduce a novel restriction site into the mutation primers without changing the peptide nucleotide sequence. These sites can be used as a marker for successful mutation [26]. SiteFind uses a novel "moving window" algorithm to reduce the number of possible sequences to be searched to a manageable level. The user enters a nucleotide sequence, specifies what amino acid residues should be changed in the mutation, and SiteFind generates a list of possible restriction sites and what nucleotides must be changed to introduce that site. However, SiteFind considers the much more restricted problem of introducing a single restriction site into a short (< 400 bps) sequence specifically to serve as a marker for

[4]Complementary DNA.

successful mutagenesis[5].

Refactored genomes offer a promising technological advance to better understand the structures and functions of DNA sequences [27]. Redesign of bacteriophage genomes is a recent approach that has been deployed to the T7 [25] and M13 phage (please refer to [27] and references therein).

To solve URSPP, we have applied elitism in meta-heuristic algorithms. In addition, we have used clinically important and large viruses to conduct experiments and to compare our algorithms with previous solutions in the literature.

1.5 Objectives of the Thesis

The objectives of the thesis are as follows.

1) To devise new algorithms based on different meta-heuristic and elitist approaches.

2) To carry out extensive experiments considering different viruses considering existing algorithms as well as newly developed algorithms. Especially, we have considered clinically important viruses as well as large viruses.

3) To compare the performance of the newly designed meta-heuristic algorithms with other traditional metaheuristics in the literature.

4) To provide a cost effective software tool which will help in the field of biology.

1.6 The Contribution of this Thesis

The main target of this thesis is to aid in drug designing, in particular vaccine designing. To design an effective vaccine, we need to analyze functionality of genes and their parts contained in a viral genome sequence. To better understand what a specific part in the sequence does, what is does not, it is a biologically popular way to keep that specific part in the sequence and temporarily remove/abrupt/make de-functional others using some chemicals (enzymes). Synthesizing viral sequences in such a way so that specific part of the sequence is responsive to some enzymes and others are non-responsive, is a computationally hard problem. The contributions of this thesis are as follows:

[5]Mutagenesis is a process by which the genetic information of an organism is changed in a stable manner, resulting in a mutation.

In this thesis, we apply some *elitist meta-heuristic techniques* to find synthesized genome sequences. To achieve this, we first map the problem into a *multi-objective optimization problem*. We have proposed 3 different algorithms to find these genome sequences. All of our proposed meta-heuristic algorithms work in two steps. Firstly, our algorithms generate different population sets based on different objectives. We have proposed three different meta-heuristic approaches/algorithms to generate these populations. Secondly, combine these populations and apply multi-objective meta-heuristic approach on those combined populations. Besides these elitists meta-heuristic algorithms, we have also used some other multi objective optimization techniques and conducted some experiments with these algorithms.

We have introduced a new objective of the problem which is more meaningful and significant from the biological viewpoint. To examine a genome sequence, biologists want to cut the sequence to equal sized subsequences. Equal sized subsequences are more helpful for biologists to examine. Say, a sequence is cut into some big and some small subsequences. Obviously, big subsequences are very hard to examine by biologists. On the other hand, the little sequences are not interesting for biologists. For this, we have introduced a new objective which can be more interesting for biologists.

We have carried out extensive experiments with different viral genome sequences considering our newly developed algorithms as well as existing meta-heuristic algorithms. So far, the existing algorithms were experimented with the small viral sequences in the literature. To analyze the performance of those algorithms, we have conducted our experiments with some clinically important viral sequences. Since the lengths of these clinically important viral sequences are not large enough, we have extended our experiments with some large viral sequences.

We have compared the performance of our newly designed and developed algorithms with other traditional meta-heuristics in the literature. We have conducted suitable statistical tests for every comparison.

We have provided an open source java software tool for biologists. We believe that this tool will help biologists a lot for examining the viral plasmid sequences and will contribute for gene therapy and vaccine development. To the best of our knowledge, the synthesized genome designed by this tool would be most attractive to the biologists

because of cost and other criteria.

1.7 Organization of Thesis

A brief overview of the organization of the rest of this thesis is presented in this section. In chapter 2, we present some relevant concepts of Bioinformatics notions, basic Meta-heuristics algorithms and some Multi objective optimization techniques which are used through out the thesis. A fundamental challenge in Bioinformatics is transforming a biological problem into a computational problem. Before such conversion, it is quite necessary to understand the biological context so that they can smartly be applied in algorithmic context. Meta-heuristic Algorithms show good result for a variety of NP hard problems. Basic meta-heuristic algorithms are very important to understand the hybrid meta-heuristic techniques.

In chapter 3, we present our efforts to construct new genome sequences applying different elitist meta-heuristic techniques. This chapter first presents our motivation behind using elitism. We also present our motivation to introduce new objective of the problem. Then all the required notions to apply elitism are introduced at length. Finally, we formally present our algorithms in this chapter.

In chapter 4, we present insightful discussion based on the comparisons done with state of the art meta-heuristic algorithms on the same problem. We have designed and conducted extensive experiments to analyze the performance of our algorithms. This chapter is dedicated to present the experimental results. At the end of this chapter, we have provided summary results along with insightful discussions about the outcome followed by a comparative analysis with other related meta-heuristic results in the literature.

In chapter 5, we present a brief description of our open source software tool. This chapter also describes how to run the software with a genome sequence as well as the format of the required input files.

In chapter 6, we conclude our thesis and put our results into the context of the state of the art of the literature. Some future research directions are also identified and briefly discussed in this chapter.

CHAPTER 2
Background

In this chapter, we discuss some terminology, concepts and relevant notations and meta-heuristics background that are used throughout the thesis. The terms discussed here, belong to both biological and algorithmic studies. The discussion in this chapter will help the readers to grasp the significance and importance of our study and will aid in properly comprehending the meaning of studied problem which are discussed throughout the subsequent chapters.

2.1 Synthetic Biology

A brief description of synthetic biology has been given in Chapter 1. Since our focus is on this particular branch of bioinformatics, here, we discuss about it more elaborately. The new buzzword 'Synthetic Biology' entered the vocabulary of the scientific community only a few years ago [28, 29]. Synthetic biology refers to both: (a) the design and fabrication of biological components and systems that do not already exist in the natural world and (b) the re-design and fabrication of existing biological systems. Synthetic biologists approach the creation of new biological systems from different perspectives, focusing on finding how life works (the origin of life) or how to use it to benefit society.

The advance of synthetic biology relies on several key enabling technologies provided at ever increasing speed and lower cost. DNA sequencing, fabrication of genes, modeling how synthetic genes behave, and precisely measuring gene behavior are essential tools in synthetic biology. Its popularity has grown as a result of increasing developments within DNA synthesis technologies. Now it is more affordable to synthesize a gene as opposed to cloning it. Also, genome databases can be used as a template for creating viruses at minimal cost.

2.1.1 Viral Genome Synthesis

Gene synthesis has become an important tool in many fields of recombinant DNA technology including vaccine development, gene therapy and molecular engineering. The

synthesis of nucleic acid sequences is often more economical than classical cloning and mutagenesis procedures. Gene synthesis is the process of synthesizing a gene without the need for initial template DNA samples. Genome synthesis technique is largely used for virus vaccine development. For example, in 2002 Eckard Wimmer's group at SUNY Stony Brook succeeded in synthesizing poliovirus from its chemical code, producing the world's first synthetic virus. Scientists first converted poliovirus's published RNA sequence, 7741 bases long, into a DNA sequence, as DNA was easier to synthesize. Short fragments of this DNA sequence were obtained by mail-order, and assembled. The complete viral genome was then assembled by a gene synthesis company. This whole painstaking process took two years. Nineteen markers were incorporated into the synthesized DNA, so that it could be distinguished from natural polio viruses.

2.1.2 Refactoring: Genome Vs. Software

Refactoring [30] is a software engineering term for redesigning a program to improve its internal structure for better ease of maintenance while leaving its external behavior unchanged. Genome synthesis technology enables us to refactor biological organisms: we seek to restructure the genome of an organism into a sequence which is functionally equivalent (i.e., behaves the same in its natural environment) while being easier to manipulate [25, 27].

2.1.3 Restriction Enzymes and Restriction Sites

Restriction enzymes are key reagents for a variety of applications including genomic mapping and DNA sequencing. For diagnosing/synthesizing DNA sequence restriction enzymes offer nonparallel opportunities. These special enzymes recognize and cut specific nucleotide sequences in DNA molecules. For example, the *Escherichia coli* produces *EcoRI enzyme* that cuts DNA wherever it finds the sequence 5'GAATTC3'/3'CTTAAG5'. The pattern, GAATTC/CTTAAG, is called the *restriction enzyme recognition site/restriction site*. Unique restriction enzyme cuts the DNA at exactly one place. Due to their unambiguous recognition property, unique restriction enzymes are more interesting, useful and appealing to the scientists. A sequence containing unique restriction sites at regular intervals is easier to manipulate in the laboratory. That is why, in many cases, before ex-

perimenting the original sequence obtained from a living organism, such unique sites are artificially inserted and/or deleted in the DNA sequence keeping it functionally equivalent to the original one. Often, inserting (deleting) a recognition site for a restriction enzyme is referred to as inserting (deleting) a restriction enzyme.

2.1.4 Protein and Amino Acid

Proteins are one of the building blocks of the body and Amino Acids serve as the building blocks of proteins, which are basically, linear chains of Amino Acids. Amino Acids can be linked together in varying sequences to form a vast varieties of proteins. A total of 20 different kinds of amino acids form proteins. These 20 amino acids are encoded by the universal genetic code. To restructure the genome of an organism into a functionally equivalent sequence, the Amino Acid sequence of it must be preserved. The redundancy of the genetic code (to be described in the following section) plays an important role in preserving the amino acid sequences or proteins even after the insertion of a new restriction site at a certain place of the genome or after the removal of any restriction site from it.

2.1.5 Genetic Code: Codon

Each amino acid is encoded by a series of three adjacent bases, called Codon. Codons specify which amino acid will be added next during protein synthesis. With some exceptions, a three-nucleotide codon in a nucleic acid sequence specifies a single amino acid. The genetic code has redundancy but no ambiguity (see the Table 2.1). For example, although codons CAA and CAG both specify Glutamine (redundancy), neither of them specifies any other amino acid (no ambiguity). There are three amino acids encoded by six different codons: serine, leucine, and arginine. Only two amino acids are specified by a single codon. One of these is the amino-acid methionine, specified by the codon ATG, which also specifies the start of translation; the other is tryptophan, specified by the codon TGG.

Degeneracy results because there are more codons than encodable amino acids. There are only 20 different kinds of amino acids found in the proteins of living organisms whereas 64 ($4^3 = 64$) possible codons are available. Multiple codons representing the

Table 2.1: Amino Acid Descriptions

Amino Acid	SLC	DNA codons
Isoleucine	I	ATT, ATC, ATA
Leucine	L	CTT, CTC, CTA, CTG, TTA, TTG
Valine	V	GTT, GTC, GTA, GTG
Phenylalanine	F	TTT, TTC
Methionine	M	ATG
Cysteine	C	TGT, TGC
Alanine	A	GCT, GCC, GCA, GCG
Glycine	G	GGT, GGC, GGA, GGG
Proline	P	CCT, CCC, CCA, CCG
Threonine	T	ACT, ACC, ACA, ACG
Serine	S	TCT, TCC, TCA, TCG, AGT, AGC
Tyrosine	Y	TAT, TAC
Tryptophan	W	TGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAT, AAC
Histidine	H	CAT, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAT, GAC
Lysine	K	AAA, AAG
Arginine	R	CGT, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	TAA, TAG, TGA

same amino acid are called *Synonymous codons*. These properties of the genetic code make it more fault-tolerant for point mutations. For successful insertion and deletion of restriction sites, synonymous codons must be used while placing one codon in lieu of another. Figure 2.1 shows an example of this concept. GAG and GAA both are the code for the aminoacid glutamic acid. Here a single nucleotide change introduces the EcoRI restriction site, without modifying the amino acid sequence.

2.1.6 Subcloning

In molecular biology, subcloning is a technique used to move a particular gene of interest from a parent vector to a destination vector in order to further study its functionality. Restriction enzymes are used to excise the gene of interest (the insert) from the parent. Simultaneously, the same restriction enzymes are used to digest (cut) the destination. The insert and the destination vector are then mixed together with certain ratio.

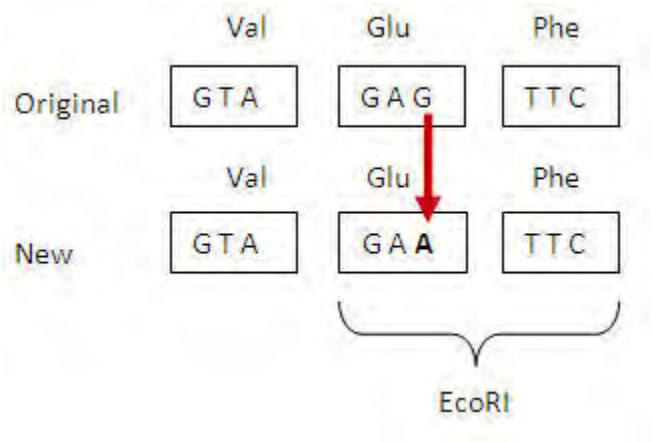

Figure 2.1: Insertion of a restriction site.

After letting the reaction mixture sit for a set amount of time at a specific temperature, the insert should become successfully incorporated into the destination plasmid.

2.2 Meta-heuristics

Before describing elitist meta-heuristics techniques to solve URSPP, here we present basic definitions of meta-heuristics and some search techniques. Meta-heuristics is a widely acknowledged tool in addressing problems in numerous and diverse fields. Our discussion on meta-heuristics in this chapter is confined within the techniques that have been applied in the literature to solve the problem under consideration.

2.2.1 Definition of Meta-heuristics

In computer science and mathematical optimization, a meta-heuristic is a procedure designed to find a good solution to a difficult optimization problem [31]. Meta-heuristics make few assumptions about the optimization problem being solved, and so they are usable for a variety of problems [32]. Meta-heuristics are used for combinatorial optimization in which an optimal solution is sought over a discrete search-space.

The term meta-heuristic, first introduced in [33], derives from the composition of two Greek words. 'Heuristic' derives from the verb heuriskein (euriskein) which means "to find", "art of discovering new", while the suffix meta means "beyond, in an upper level". Before this term was widely adopted, meta-heuristics were often called modern heuristics [34]. Below we give some recent definitions of meta-heuristics [35, 36],

> "A meta-heuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method."

> "A meta-heuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space. Learning strategies are used to structure information in order to find efficiently near-optimal solutions."

The properties that characterize most meta-heuristics are as follows : 1) Meta-heuristics are strategies that guide the search process. 2) The goal is to efficiently explore the search space in order to find near optimal solutions. 3) Techniques which constitute meta-heuristic algorithms range from simple local search procedures to complex learning processes. 4) Meta-heuristic algorithms are approximate algorithms and usually are non-deterministic. 5) Meta-heuristics are not problem-specific.

Typically, meta-heuristics are approximation algorithms and hence they cannot always produce provably optimal solutions. But they do have the potential to produce good solutions in short amount of time (if used appropriately). Meta-heuristics are different from exact optimization algorithms in that they do not guarantee the optimality of the obtained solutions. On the other hand, unlike approximation algorithms, nor does it define how close are the obtained solutions from the optimal ones.

2.2.2 Single State Methods

These methods are often called local search techniques. In combinatorial optimization, Single State Methods have been proved to be very useful. It has been shown to be very effective for a large number of combinatorial problems. These techniques are based on the iterative exploration of a solution space: at each iteration, the algorithm steps from one solution to one of its neighbors, i.e., solutions that are (in some sense) close to the former.

2.2.2.1 Hill Climbing

Hill Climbing is the simplest local search technique. This technique is related to gradient ascent, but it doesnt require to know the strength of the gradient or even its direction. New candidate solutions are iteratively tested in the region of the current candidate and the new ones are adopted if they are better. This enables to climb up the hill until the local optima is reached. The basic structure of Hill climbing is outlined in Algorithm 1.

Algorithm 1 HILL CLIMBING

```
1: S ← some initial candidate solution
2: repeat
3:    R ← Tweak(Copy(S))
4:    if Quality(R) > Quality(S) then
5:       S ← R
6:    end if
7: until S is the ideal solution or we have run out of time
8: return S
```

2.2.2.2 Steepest Ascent Hill Climbing

Steepest Ascent Hill Climbing is little more aggressive than simple Hill Climbing algorithm. It creates n "tweaks" to a candidate solution all at one time, and then adopts the best one. By sampling all around the original candidate solution and then picking the best, Steepest Ascend Hill Climbing essentially samples the gradient and marches straight up it. Algorithm 2 illustrates Steepest Ascent Hill Climbing techniques.

2.2.3 Population Methods

Population-based methods differ from the previous methods in that they keep around a sample of candidate solutions rather than a single candidate solution. Each of the solutions is involved in tweaking and quality assessment, but what prevents this from being just a parallel hill-climber is that candidate solutions affect how other candidates will hill-climb in the quality function. This could happen either by good solutions causing poor solutions to be rejected and new ones created, or by causing them to be Tweaked in the direction of the better solutions.

Evolutionary Algorithms (EA) are popular approaches to solving single and multi objective optimization problems [37, 38]. These are search methods that take inspira-

Algorithm 2 STEEPEST ASCENT HILL CLIMBING

```
1: n ← number of tweaks desired to sample the gradient
2: S ← some initial candidate solution
3: repeat
4:    R ← Tweak(Copy(S))
5:    for n1 times do
6:       W ← Tweak(Copy(S))
7:       if Quality(W) > Quality(R) then
8:          R ← W
9:       end if
10:   end for
11:   if Quality(R) > Quality(S) then
12:      S ← R
13:   end if
14: until S is the ideal solution or we have run out of time
15: return S
```

tions from natural phenomenon of selection and survival of the fittest in the biological world. EAs are well suited for a wide range of combinatorial and continuous problems, though the different variations are tailored towards specific domains. The variant, namely, Genetic algorithms are well suited for optimizing combinatorial problems.

2.2.3.1 Genetic Algorithms

The Genetic Algorithm (GA) was invented by John Holland at the University of Michigan in the 1970s [39]. It is similar to a (μ, λ) Evolution Strategy [38] in many respects. It iterates through fitness assessment, selection and breeding, and population reassembly. The primary difference is in how selection and breeding take place whereas Evolution Strategies select the parents and then creates the children. The Genetic Algorithm little-by-little selects a few parents and generates children until enough children have been created. To breed, the Genetic Algorithm begins with an empty population of children. It then selects two parents from the original population, copies them, crosses them over with one another, and mutates the results. This forms two children, which then will be added to the child population. This algorithm repeats this process until the child population is entirely filled. Algorithm 3 provides the pseudo code for Genetic Algorithm.

Algorithm 3 GENETIC ALGORITHM

```
1: popsize ← desired population size {Make it even.}
2: P ← {}
3: for popsize times do
4:    P ← P∪ new random individual
5: end for
6: Best ← null
7: repeat
8:    for each individual P_i ε P do
9:       AssessFitness(P_i)
10:      if Best = null or Fitness(P_i) > Fitness(Best) then
11:         Best ← P_i
12:      end if
13:   end for
14:   Q ← {}
15:   for popsize/2 times do
16:      Parent P_a ← SelectWithReplacement(P)
17:      Parent P_b ← SelectWithReplacement(P)
18:      Children C_a, C_b ← Crossover(Copy(P_a), Copy(P_b))
19:      Q ← Q∪ Mutate(C_a), Mutate(C_b)
20:   end for
21:   P ← Q
22: until Best is the ideal solution or we have run out of time
23: return Best
```

2.2.4 Selection Procedure

In the meta-heuristics algorithms the selection procedure plays a significant role. The selection procedure decides which candidate solutions to retain and which to reject as it wanders through the space of possible solutions to the problem [38]. We use Tournament Selection as Selection Procedure to solve our problem. ; Tournament Selection is a non-parametric selection algorithm which is both simple and robust (tunable). It throws away the notion that fitness values mean anything other than bigger is better, and just considers their rank ordering. The algorithm returns the fittest individual of some t individuals picked at random, with replacement, from the population. Algorithm 4 provides the pseudo code for Tournament Selection.

Algorithm 4 TOURNAMENT SELECTION

1: $P \leftarrow$ Population
2: $t \leftarrow$ tournament size, $t \geq 1$
3: $Best \leftarrow$ individual picked at random from P with replacement
4: **for** $i = 2$ to t **do**
5: $Next \leftarrow$ individual picked at random from P with replacement
6: **if** Fitness($Next$) $>$ Fitness($Best$) **then**
7: $Best \leftarrow Next$
8: **end if**
9: **end for**
10: **return** $Best$

2.2.5 Multiobjective Optimization

Multiobjective optimization (also known as multiobjective programming, vector optimization, multicriteria optimization, multiattribute optimization or Pareto optimization) is an area of multiple criteria decision making, that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Multiobjective optimization has been applied in many fields of science, including engineering, economics and logistics where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Minimizing weight while maximizing the strength of a particular component, and maximizing performance whilst minimizing fuel consumption and emission of pollutants of a vehicle are examples of multiobjective optimization problems involving two and three objectives, respectively.

The process of optimizing systematically and simultaneously a collection of objective functions is called multiobjective optimization. There are many multi objective optimization techniques i.e., Steady State Genetic Algorithms, Goal Programming, Preemptive Methods, Weighted Sum Methods etc. Among these algorithms, Non-Dominated Sorting Genetic Algorithm (NSGA) gives significant better results for multiobjective optimization.

2.2.5.1 Non-Dominated Sorting Genetic Algorithm

Before going into details of Non-Dominated Sorting Genetic Algorithm, we need some definitions. Individual A Pareto dominates Individual B if A is at least as good as B in every objective and better than B in at least one objective. Computing pareto

domination and binary tournament selection based on pareto do;mination is illustrated in Algorithm 5.

Algorithm 5 PARETO DOMINATION

```
1: A ← individual A
2: B ← individual B
3: O ← {O_1, ....., O_n} Objectives to assess with
4: a ← false
5: j ← random number picked uniformly from 1 to n
6: for each objective O_j ϵ O do
7:    if ObjectiveValue(O_j, A) > ObjectiveValue(O_j, B) then
8:       a ← true
9:    else if ObjectiveValue(O_j, A) < ObjectiveValue(O_j, B) then
10:      return false
11:   end if
12: end for
13: return a
```

The set of individuals that can not pareto dominate each other lies in the same pareto front and individual on same pareto front has the same rank. The lower the rank the better the candidate solution is. So the best candidate solutions in a population have pareto front rank 1.

Non-Dominated Sorting was first proposed by Srinivas and Deb in [40]. In this sorting approach, the population P is first partitioned into ranks, with each rank (a group of individuals) stored in the vector F. Then, a rank number is assigned to an individual (perhaps the individual gets it written internally somewhere). So the fitness of i_{th} individual is, $fitness(i) = \frac{1}{1+ParetoFrontRank(i)}$

While selecting in NSGA, along with the fitness defined above, another measure, namely, sparsity is used. It is assumed that, if the individuals in the population are being spread more evenly across the front, the population can serve the purpose of searching a good solution in a better way. The sparsity of an individual employs the following notion: an individual is in a more sparse region if the closest individuals on either side of it in its Pareto Front Rank are not too close to it. Individuals at the far ends of the Pareto Front Rank are assigned an infinite sparsity. As a selection procedure, the tournament selection is used where based on Pareto Front Rank individuals are selected first. If there is a tie, it is broken by using sparsity. These helps to get the individuals which are not

only close to the true Pareto front, but also nicely spread out along it. Non-Dominated Sorting Lexicographic Tournament Selection with Sparsity is shown in Algorithm 6.

Algorithm 6 NON-DOMINATED SORTING LEXICOGRAPHIC TOURNAMENT SELECTION WITH SPARSITY

```
1: P ← Population with Pareto Front Ranks assigned
2: Best ← individual picked at random from P with replacement
3: t ← tournament size, t ≥ 1
4: for i = 2 to t do
5:    Next ← individual picked at random from P with replacement
6:    if ParetoFrontRank(Next) < ParetoFrontRank(Best) then
7:       Best ← Next
8:    else if ParetoFrontRank(Next) = ParetoFrontRank(Best) then
9:       if Sparsity(Next) > Sparsity(Best) then
10:         Best ← Next
11:      end if
12:   end if
13: end for
14: return Best
```

Non-Dominated Sorting Genetic Algorithm keeps around all the best known individuals so far, in a sort of $(\mu + \lambda)$ or elitist notion. The general idea is to hold in P an archive of the best n individuals discovered so far. A new population Q is generated from P by breeding operation. Then every individual in P and Q competes to stay in the archive.

2.3 Summary

In summary, in this chapter we have discussed Synthetic Biology and various meta-heuristics techniques which are needed for the following chapters. Here, we discuss about some biological keywords which will be used throughout this thesis. We will present how we apply meta-heuristics techniques in the problem under consideration in the following chapters. These meta-heuristic techniques will be used also to apply elitism to solve the underlying problem. Therefore, the discussion of meta-heuristics of this chapter aids the reader to understand the meaning of terms and notations used there.

CHAPTER 3
URSPP in Synthetic Genomes by Elitists Meta-heuristics

In this chapter, we present our algorithms. In particular, the application of various search techniques using an elitist approach to obtain the desired genome sequence are illustrated in this chapter. At first we briefly present our motivation to apply the elitist meta-heuristics. Then we elaborately discuss all the aspects of our algorithms: how we represent our solution, problem specific elitist approach, how to asses the fitness of a solution and the algorithmic steps. In addition, we discuss about how and why we introduce a new modified objective for URSPP.

3.1 Motivation behind application of Eliitist Meta-heuristics

Over the last decades, research on solving multi-objective problems using meta-heuristic techniques and algorithms has attracted a lot of attention from the scientific community. A majority of these algorithms use either Pareto dominance [38] or weighting method for fitness assignment. Unique Restriction Site Placement Problem (URSPP) is a multi-objective optimization problem. Multi-objective optimization often shows good performance through elitist approach. This thesis applies Genetic algorithms/ Local search techniques to generate populations based on different objectives and uses pareto dominance with those populations.

A disadvantage of heuristic methods is that they either generate only a very limited number of different solutions, or stop at poor quality local optima, which is the case for iterative improvement methods. In this thesis, to solve the URSPP, elitist meta-heuristics have been proposed with a goal to overcome these problems.

3.2 Motivation behind change of an objective

We have changed an objective of URSPP to make it more useful and meaningful for biologists. In the literature so far, the maximum distance between two consecutive inserted enzymes has been considered as one of the objectives of URSPP. However, as it turns out this objective is not always meaningful in the context of biology as discussed

below. A genome sequence with maximum enzymes in a concentrated area[6] does not aid biologists in examining the complete sequence. This is because there may exist many big areas without any restriction enzymes and hence those areas would remain completely unexplored. Hence, it seems more meaningful to formulate an objective that allows biologists to explore more areas of the genome. In this regard, we introduce the standard deviation of gaps for inserted enzymes as an objective instead of maximum gaps of consecutive inserted enzymes.

Table 3.1: Two Different Solutions - 1

	p_1	p_2	p_3	p_4	p_5	p_6
$Solution_1$	16	104	123	209	227	317
$Solution_2$	50	97	202	256	311	354

For example, we have a sequence of 400 base pairs. Table 3.1 shows two different solutions with the position of 6 inserted enzymes for each solution. P_i denotes the position of ith insertion enzyme.

Table 3.2: Two Different Solutions - 2

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	Max Gap	STD
$Solution_1$	16	88	19	86	18	90	83	90	37
$Solution_2$	50	47	105	54	55	43	46	105	21.54

Table 3.2 is formed from Table 3.1. In Table 3.2, g_1,g_2,....,g_7 denotes the gaps between two consecutive enzymes. *Max Gap* and *STD* are denoted as *Maximum Gap* and *Standard Deviation* respectively. For $Solution_1$, maximum gap is 90 and standard deviation is 37 where as, for $Solution_2$ maximum gap is 105 and standard deviation is 21.54. If we prefer the objective of "Maximum Gap", then we have to take $solution_1$ which has many big areas and investigate with those big areas would be cumbersome for biologists. But if we prefer the objective of "Standard Deviation", then we have to take $Solution_2$. $Solution_2$ has only one big area. Other areas are near about equal sized and these equal sized areas are more preferable for biologists to examine. Figure 3.1 illustrates the cutting parts of two solutions.

[6]area is a subsequence of a genome sequence

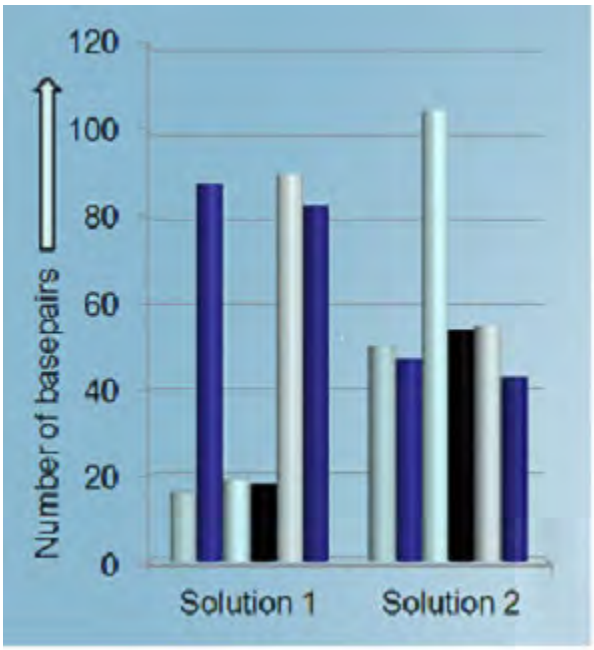

Figure 3.1: Parts of the Cutting Sequences.

3.3 Proposed Methodology

Before starting the main algorithm, we preprocess the given set of restriction enzymes to construct the Restriction Map [17]. Restriction Map is a data structure used to keep track of the list of restriction enzymes, each with its name and its recognition site. The map is built using a dictionary-matching algorithm, namely the Aho-Corasick algorithm [41] to efficiently find all occurrences of a finite set of patterns P in a given text. When we get the sequence, further processing is done as follows. A simple $O(nm)$ time algorithm is implemented to find all the possible places where a given restriction enzyme can be inserted, where n is the length of the sequence and m is the length of the recognition site. Now, for each restriction enzyme, all the occurrences from the Restriction Map and all the possible insertion points are listed. This list is used as the potential insertion list while applying breeding operators on candidate solutions as will be described in the following sections.

3.3.1 Candidate Solution Representation

For representation, we have followed the strategy of [19]. Each candidate solution is represented by a direct encoding of a fixed sized double vector as shown in Table

3.3 [19]. In this representation, one sub-vector, namely $\mathcal{B}$, is boolean denoting the absence/presence of restriction enzymes and the other, namely $\mathcal{L}$, keeps track of the possible locations (taken from potential insertion list of that enzyme) to insert the enzymes. Each position of the whole vector is dedicated for a particular restriction enzyme which is referred to as a gene and g_i denotes the ith gene of the vector. We follow a left to right order while traversing a vector which means that the enzymes are inserted from left to right. The size, N, of each candidate solution is equal to the number of restriction enzymes that are going to be inserted. According to the sample solution shown in Table 3.3, restriction enzyme, RE_1 is present and its location is l_1. Note carefully that RE_2 in Table 3.3 is absent and hence the location l_2 is currently insignificant. However, if the corresponding presence bit is turned on (e.g., due to mutation), l_2 will be RE_2's location.

Table 3.3: One Candidate Solution

Meaning of notation	RE_1,	RE_2,	...	RE_i,	...
Gene for RE's	g_1	g_2	...	g_i	...
Presence or Absence, $\mathcal{B}$	1,	0,	...	1,	...
Possible location, $\mathcal{L}$	l_1,	l_2,	...	l_i,	...

Here, the vector size, N is less than or equal to the cardinality of the given enzyme set, S. If the site in the locked region occurs once for a restriction enzyme, it is taken as inserted. If however, for any restriction enzyme more than one sites reside in the locked region, then it is assumed that the enzyme 'can never be inserted'. Rest of the restriction enzymes are used to construct candidate solutions and hence $N \leq |S|$. However, such different dealing of the sites in the locked region follows from the fact that we can not alter any base in these region. Notably, the same strategy has been adopted in [17, 19, 20].

3.3.2 Elitist Meta-heuristics

Our solutions use different meta-heuristic techniques i.e., Hill Climbing, Steepest Ascent Hill Climbing, Genetic Algorithm and Non-dominated Sorting Genetic Algorithm (NSGA). These algorithms were also used in [19] and [20] but we introduce elitism and propose hybridization.

As in [19], we use iterative algorithms like hill climbing which starts with a random individual solutions of URSPP. Then it tries to find a better solution by incrementally

changing a single element. In simple hill climbing, the first closer node is chosen, whereas in steepest ascent hill climbing all successors are compared and the closest to the solution is chosen. As in [20], we also use basic Genetic Algorithm and Non-Dominated Sorting Genetic Algorithm. Among these, basic GA improves its solution by a popular local improver. Basic GA little-by-little selects a few parents and generates children until enough children have been created. To breed, we select two parents from the original population, copy them, cross them over with one another, and mutate the results. This forms two children, which we then add to the child population. We repeat this process until the child population is entirely filled. For the uniqueness of inserted enzymes, we always apply a validate operation after a crossover or mutation.

Algorithm 7 Elitist Meta-heuristic Approach for URSPP

1: $n \leftarrow 3$ (Number of objectives)
2: $P \leftarrow \{\}$
3: **for** T times **do**
4: $P \leftarrow P \cup$ new random individual
5: **end for**
6: **for** $k = 1 \rightarrow n$ **do**
7: **for** T times **do**
8: $P \leftarrow P \cup$ new individual generated by Hill Climbing/Steepest Ascent Hill Climbing/Genetic Algorithm based on objective k
9: **end for**
10: **end for**
11: Apply NSGA with P

In our approach, we incorporate the elitist notion to NSGA as follows. The idea is to provide NSGA with an initial population that exhibits elitism in some sense. We use one of the 3 objectives to generate fit population based on the particular objective using some of the methods used in [19, 20]. In particular, we use Hill Climbing/Steepest Ascend Hill Climbing/Genetic Algorithm considering a particular objective to get some quality solutions with respect to that objective. We do this for each objective and then add their solutions to form the initial population for NSGA. Algorithm 7 provides the high level pseudo code for our approach.

For Hill Climbing/Steepest Ascent Hill Climbing/Genetic algorithm in Lines 8 of Algorithm 7, we consider only a single objective. In Line 3 and 7, T is a preselected number. We generate T number of populations for every objective. We refer to these

populations as *elite populations*, which are found after the execution of Line 8 of Algorithm 7. In Algorithm 7, these elite populations are used as initial population of NSGA. In Contrast, random populations were used in [20]. To include an element of diversity, initially we had a plan to blend these elite population with some randomly generated population. But through preliminary experiments it was found that adding randomness does not provide significant better results.

We have proposed 3 different approaches to solve URSPP.

Approach 1: Generate populations with Hill Climbing considering each objective separately and apply NSGA with those elite population sets.

Approach 2: Generate populations with Steepest Ascent Hill Climbing considering each objective separately and apply NSGA with those elite population sets.

Approach 3: Generate populations with Genetic Algorithms considering each objective separately and apply NSGA with those elite population sets.

3.3.3 Breeding Operators

Algorithm 8 MUTATION

1: $G < g_1, g_2, ..., g_N >$ vector to be mutated
2: $r \leftarrow$ random integer picked uniformly from 1 to N
3: $\alpha \leftarrow$ probability of bit flip of *presence*
4: $p \leftarrow$ pick a random value from 0.0 to 1 inclusive
5: **if** $p < \alpha$ **then**
6: flip the presence bit, b_r
7: **else**
8: pick another random location from the list of potential insertion points
9: update the location, l_r with newly chosen location
10: **end if**

We have followed the strategy from [20] for breeding operation. While applying the Genetic Algorithm and the NSGA, for breeding of new candidate solutions we use both recombination/crossover. For recombination we apply the standard two point crossover technique as follows. Say, N is the size of candidate solution vector. In two point crossover, we pick two numbers i and j, where $1 \leq i < j \leq N$ and swap the genes between them. From biology we know that, properties, specially degeneracy, of the genetic code make a genome sequence more fault-tolerant for point mutations. Hence, for

mutation we apply a variation of the point mutation. Our mutation operator is problem specific and is described in Algorithm 8.

3.3.4 Standard Deviation Instead of Maximum Gap

Recall that, our goal is to place unique restriction sites for as many restriction enzymes as possible allowing minimum number of base changes and to minimize the maximum gap between the consecutive sites. So, to determine the quality of a candidate solution, we have to consider three criteria: (1) number of unique sites, (2) number of base changes from the original genome sequence and (3) the maximum gap between the consecutive sites. We formally define, the following objectives for these criteria:

f_1 = number of inserted restriction enzymes

f_2 = number of positions where the original sequence differs from the synthesized sequence

f_3 = the maximum distance between two consecutive restriction sites.

Clearly, Our aim is to have higher f_1 and lower f_2 and f_3. Recall that, The objective f_3 (i.e., *Maximum Distance between two consecutive restriction sites*) is not always meaningful in the context of biology as discussed below. A genome sequence with maximum enzymes in a concentrated area does not aid biologists in examining the complete sequence. This is because there may exist many big areas without any restriction enzymes and hence those areas would remain completely unexplored. Hence, it seems more meaningful to formulate an objective that allows biologists to explore more areas of the genome. In this regard, we introduce the standard deviation of gaps for inserted enzymes as an objective instead of maximum gaps of consecutive inserted enzymes. We denote the resulting new objective as f_4 as follows:

f_4= the standard deviation of gaps of two consecutive restriction enzymes.

Our experiments and analyses are done with two different programs with respect to the two different set of objectives. In particular, we have done our experiments with objective f_1,f_2 and f_3 and then f_1,f_2 and f_4 in two separate programs.

3.3.5 Quality Assessment

Fitness or quality of solution is higher when f_1 is higher and f_2 and f_3 (or f_4) are lower. A naive way to assess fitness could be to define the quality of a solution as a weighted sum of how well it meets various objectives. This approach is used to locally find a solution of better quality.

When we apply algorithms for single objective, we have concentrated only with the specific objective. For example, when our algorithms are generating population for objective f_1, it tries to maximize f_1 while generating population. Our algorithms do not even think about the other objectives while working with one objective. This is also true for generating populations for f_2 and f_3 (or f_4). Note carefully that, we always try to minimize f_2 and f_3 (or f_4) while we are working with them.

The other kind of fitness assessment we apply is the Pareto domination while using NSGA. Recall that, Individual A pareto dominates Individual B if A is at least as good as B in every objective and better than B in at least one objective. All individuals who can not pareto dominate each other forms pareto front of same rank. Pareto front of lowest rank gives the best individuals. So the fitness of i_{th} individual is, $fitness(i) = \frac{1}{1+ParetoFrontRank(i)}$.

3.4 Algorithms

URSPP has more than one goal to achieve. Recall that, we have used 3 different approaches to generate population sets. The approaches are hill climbing, Steepest Ascend Hill Climbing, Genetic Algorithm. After generating population set, we apply NSGA for multiobjective optimization using those population sets. Besides these algorithms, we have tried different multiobjective techniques.

3.4.1 Hill Climbing

Hill Climbing is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

3.4.2 Steepest Ascend Hill Climbing

In simple hill climbing, the first closer node is chosen, whereas in steepest ascent hill climbing all successors are compared and the closest to the solution is chosen. Steepest ascent hill climbing is similar to best-first search, which tries all possible extensions of the current path instead of only one.

3.4.3 Genetic Algorithm

The Genetic algorithms applied in URSPP is basic GA. Then it is hybridized by a popular local improver, namely, hill-climbing. Among these, basic GA little-by-little selects a few parents and generates children until enough children have been created. To breed, we begin with an empty population of children. We then select two parents from the original population, copy them, cross them over with one another, and mutate the results. This forms two children, which we then add to the child population. We repeat this process until the child population is entirely filled.

3.4.4 Non-Dominated Sorting Genetic Algorithm

The previous algorithms are used to generate populations based on a single objective. We use NSGA with those populations for multiobjective optimization. In NSGA we try to find which candidate solution pareto dominates others and thus extract only solutions consisting the pareto front. Since we return the lowest pareto front ranked solutions with sparsest individual, most of the local and global optima individuals can be found. Note that, when a potential insertion point for a recognition site of a given enzyme is picked we may accidentally create a recognition site for another enzyme. Similar situation may happen when an enzyme with multiple occurrences are turned on and all but the selected locations are deleted. To avoid this undesirable effect, each crossover and mutation is accompanied/followed by a validate operation. In the validation operation, if the creation of restriction site for so far inserted enzyme is detected, we choose next possible insertion point from potential insertion list. If no such possibility can be found, the presence bit of corresponding restriction enzyme is turned off. The other constraint, like avoiding modifying locked regions and maintaining the amino acid sequence of genes, are handled while constructing the potential insertion point list of enzymes.

3.5 Overall Elitist Approach

Algorithm 9 Get Pareto Front

```
1:  Input: Population set P
2:  front ← {}
3:  for each p ∈ P do
4:    if p has no rank then
5:      insert ← 1
6:      for i = 0 to sizeof(front) do
7:        f ← i_th element of front
8:        if f Pareto Dominates p then
9:          insert ← 0
10:         break
11:       else if p Pareto Dominates f then
12:         remove i_th individual from front
13:         continue
14:       end if
15:     end for
16:     if insert = 1 then
17:       add p to front
18:     end if
19:   end if
20: end for
21: return front
```

The detailed algorithms of our proposed approach are given below. Algorithm 9 gives the best Pareto Front from a population set. This Algorithm takes a polpulation set P as input. At first, it declares an empty population set $front$. Then it checks every individual of P. If any individual has not got rank, then this individual is compared with the others individuals of $front$ in every time for the elements of P. If the individual has quality to get into the front rank, it then be added to $front$. all the members of P which are not pareto dominated by any other individual of P will be in $front$ after the execution of Algorithm 9.

Algorithm 10 can assign ranks to all the individuals of input population. It takes a population set as input. At first, this algorithm finds the individuals which has fitness to get into front rank by calling Algorithm 9. It then add those fittest population as the first element of an population array, namely, $rankPop$. After that, it finds the individuals which have the fitness to get into the next rank. In this way, all the individuals of the input

Algorithm 10 Assign Front Rank

1: **Input: Population set** P
2: $rankPop \leftarrow \{\}$
3: $rank \leftarrow \mathbf{1}$
4: $size \leftarrow sizeof(P)$
5: **while** $size > 0$ **do**
6: $R \leftarrow$ **Get Pareto Front(**P**)**
7: **add** R **to** $rankPop$
8: **for each** $ri \in R$ **do**
9: **Rank(**ri**)** $\leftarrow rank$
10: $size \leftarrow size - 1$
11: **end for**
12: $rank \leftarrow rank + 1$
13: **end while**
14: **return** $rankPop$

populations can get their ranks.

Algorithm 11 describes our problem specific NSGA algorithm. This Algorithm takes the *elite population* as input. Then it assigns rank for every individual. Breeding Operation occurs for some preselected number of times. In every breeding, some new individuals are generated from P. After breeding in every time, our algorithm calculate the rank of the individuals. $bestfront$ gets the individuals of best front after every mutation. At the end, our Algorithm returns the best individual from the population set.

Algorithm 11 URSPP Specific NSGA

Input: Elite Population set P
$popsize \leftarrow$ **sizeof(**P**)**
$rankPop \leftarrow$ **Assign Front Rank(**P**)**
$bestfront \leftarrow$ **first element of** $rankPop$
for each $ri \in rankPop$ **do**
  assign sparsity for ri
end for
for some preselected times do
  $Q \leftarrow P \cup Breed(P)$
  $P \leftarrow \{\}$
  clear all rank of Q
  $rankPop \leftarrow$ **Assign Front Rank(**Q**)**
  $bestfront \leftarrow$ **first element of** $rankPop$
  for each $ri \in rankPop$ **do**
    assign sparsity for ri
    if $sizeof(P) + sizeof(ri) \geq popsize$ **then**
      add ($popsize - sizeof(P)$**) best elements of** ri **to** P
      break
    else
      $P \leftarrow P \cup ri$
    end if
  end for
end for
return best element of $bestfront$

3.6 Other Multiobjective Optimazation Technique

We have used other mutiobjective optimization techniques to solve URSPP in Primary Experiments. But we did not get any significant result using those. We are going to briefly discuss those techniques here.

3.6.1 Weighted Sum Model

The weighted sum model (WSM) is the simplest multi-criteria decision analysis (MCDA) / multi-criteria decision making method for evaluating a number of alternatives in terms of a number of decision criteria. It is very important to state here that it is applicable only when all the data are expressed in exactly the same unit. If this is not the case, then the final result is equivalent to "adding apples and oranges." In general, suppose that a given MCDA problem is defined on m alternatives and n decision criteria. Furthermore,

let us assume that all the criteria are benefit criteria, that is, the higher the values are, the better it is. Next suppose that w_j denotes the relative weight of importance of the criterion C_j and a_{ij} is the performance value of alternative A_i when it is evaluated in terms of criterion C_j. Then, the total (i.e., when all the criteria are considered simultaneously) importance of alternative A_i, denoted as $A_i^{WSM-score}$, is defined as follows:

$A_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij}$, for $i = 1, 2, 3, ..., m$.

For the maximization case, the best alternative is the one that yields the maximum total performance value

To solve URSPP, we have given a weight for every objective and try to solve URSPP. We have used weighted Sum to check the fitness of the hill climbing algorithm. We take a solution if it is better than the previous best. Then again tweak the solution and check weighted sum. But as we can not define a pure heuristic fuction based on those objectives using weight, we did not get a good result.

3.6.2 Goal Programming

Goal programming is a branch of multiobjective optimization, which in turn is a branch of multi-criteria decision analysis (MCDA), also known as multiple-criteria decision making (MCDM). This is an optimization programme. It can be thought of as an extension or generalization of linear programming to handle multiple, normally conflicting objective measures. Each of these measures is given a goal or target value to be achieved. Unwanted deviations from this set of target values are then minimized in an achievement function. This can be a vector or a weighted sum dependent on the goal programming variant used. As satisfaction of the target is deemed to satisfy the decision maker(s), an underlying satisfying philosophy is assumed. Goal programming is used to perform three types of analysis:

1. Determine the required resources to achieve a desired set of objectives.

2. Determine the degree of attainment of the goals with the available resources.

3. Providing the best satisfying solution under a varying amount of resources and priorities of the goals.

To solve URSPP, we set a goal as explained below. We first set goals for every objective. We set a goal to insert a preselected number of enzymes to be added by chang-

ing another preselected number of base pairs and standard deviation/maximum gap not exceeds another preselected number. For example, we have set a goal to insert not less than 80 inserted enzymes, not more than 400 base pair changes and maximum gap not exceeds 200 base pairs for some viral sequence and run our program. To insert this type of specific goals did not give us better experimental result primarily. Furthermore, goals are not static for every viral sequence. It is changed with the changing of number of base pairs of a viral sequence.

Primiry experimental results with Polio virus for these two multiobjective optimization techniques will be shown in the following chapter.

3.7 Summary

In this chapter we have presented our algorithms along with a clear description of how we represent candidate solution how to perform breeding and their assessment process. Here, we have also given a clear idea of the motivation of introducing new objective. We also present how we use elitism in meta-heuristic techniques to solve URSPP. In the next chapter, we will manifest the performance of our algorithms in compared to other heuristics which exist in literature.

CHAPTER 4
Experimental Results

We have conducted extensive experiments to analyze the performance of our algorithm and to compare it with the traditional meta-heuristic algorithms. This chapter presents our simulation results and related analysis. In addition to a simple figurative comparison, we also investigate the statistical significance of our results with respect to other results. We start this chapter with a brief discussion of different statistical tests performed here. Then we go to the simulations results.

4.1 Statistical Test

A statistical hypothesis test is a method of making decisions using data from a scientific study. In statistics, a result is called statistically significant if it has been predicted as unlikely to have occurred by chance alone, according to a pre-determined threshold probability, the significance level. The phrase "test of significance" was coined by statistician Ronald Fisher. These tests are used in determining what outcomes of a study would lead to a rejection of the null hypothesis for a pre-specified level of significance; this can help to decide whether the results contain enough information to cast doubt on conventional wisdom, given that conventional wisdom has been used to establish the null hypothesis.

4.1.1 The T Test

A t-test is any statistical hypothesis test in which the test statistic follows a Student's t distribution if the null hypothesis is supported. It can be used to determine if two sets of data are significantly different from each other, and is most commonly applied when the test statistic would follow a normal distribution if the value of a scaling term in the test statistics were known. When the scaling term is unknown and is replaced by an estimate based on the data, the test statistic (under certain conditions) follows a Student's t distribution.

4.1.2 Paired or Unpaired Test

Paired t-test is used when each data point in one group corresponds to a matching data point in the other group. Unpaired t-test is used whether or not the groups contain matching data-points. Whether you use a one- or two-tailed test depends on your testing hypothesis: One-tailed test is used where there is some basis (e.g., previous experimental observation) to predict the direction of the difference, e.g., expectation of a significant difference between the groups. Two-tailed test is used where there is no basis to assume that there may be a significant difference between the groups.

We should select a paired test when values in one group are more closely correlated with a specific value in the other group than with random values in the other group. It is only appropriate to select a paired test when the subjects were matched or paired before the data were collected.

4.1.3 Concept of Null Hypothesis

In statistical inference of observed data of a scientific experiment, the null hypothesis refers to a general or default position: that there is no relationship between two measured phenomena. Rejecting or disproving the null hypothesis, and thus concluding that there are grounds for believing that there is a relationship between two phenomena is a central task in the modern practice of science, and gives a precise sense in which a claim is capable of being proven false.

Hypothesis testing works by collecting data and measuring how likely the particular set of data is, assuming the null hypothesis is true. If the data-set is very unlikely, defined as being part of a class of sets of data that only rarely will be observed, the experimenter rejects the null hypothesis concluding that it (probably) is false. This class of data-sets is usually specified via a test statistic which is designed to measure the extent of apparent departure from the null hypothesis. The procedure works by assessing whether the observed departure measured by the test statistic is larger than a value defined so that the probability of occurrence of a more extreme value is small under the null hypothesis (usually in less than either 5% or 1% of similar data-sets in which the null hypothesis does hold).

The test statistic is compared with a lower critical value, and if it is less than this

limit, the null hypothesis is rejected. Thus, a statistical test requires a pair of hypotheses; namely,

H_0: a null hypothesis

H_a: an alternative hypothesis.

4.1.4 P Value

In statistical significance testing the P Value is the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. One often "rejects the null hypothesis" when the P Value is less than the predetermined significance level which is often 0.05 or 0.01, indicating that the observed result would be highly unlikely under the null hypothesis. Many common statistical tests, such as chi-squared tests or Student's t-test, produce test statistics which can be interpreted using P Values.

With many tests, we have to choose to calculate either a one- or two-sided P Value (same as one- or two-tailed P Value). As we know, the P Value is calculated for the null hypothesis that the two population means are equal, and any discrepancy between the two sample means is due to chance. If this null hypothesis is true, the one-sided P Value is the probability that two sample means would differ as much as was observed (or further) in the direction specified by the hypothesis just by chance, even though the means of the overall populations are actually equal. The two-sided P Value is twice the one-sided P Value.

4.1.5 Confidence Interval (C.I.)

In statistical estimation, a confidence interval (C.I.) is a kind of interval estimate which is used to indicate the reliability of an estimate. The selection of a confidence level for an interval determines the probability that the confidence interval produced will contain the true parameter value. Common choices for the confidence level C are 0.90, 0.95, and 0.99. These levels correspond to percentages of the area of the normal density curve. For example, a 95% confidence interval covers 95% of the normal curve. The probability of observing a value outside of this area is less than 0.05. Because the normal curve is symmetric, half of the area is in the left tail of the curve, and the other half of the

area is in the right tail of the curve. Figure 4.1 shows 95% Confidense Interval.

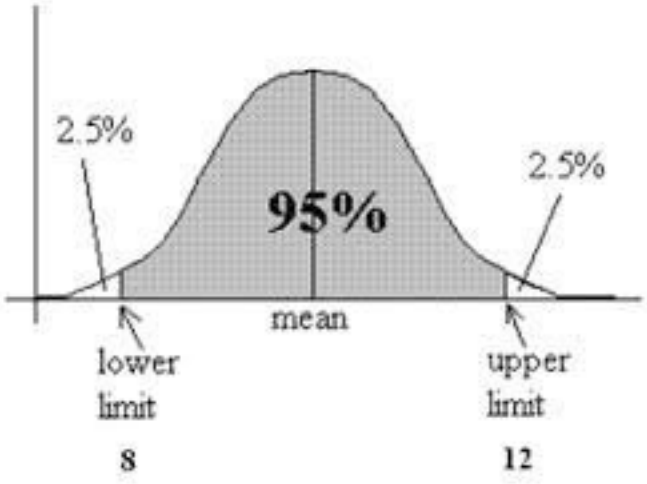

Figure 4.1: 95% Confidence Interval.

4.2 Simulation Results

4.2.1 Experimental set up and Representation

Our experiments are conducted on a computer having 2.5GHz Intel Core-i 5 processor with 4GB DDR3 Memory. The program is compiled on Java Eclipse SDK 4.2. We have run our experiments on some clinically important viral sequences obtained from the website of The European Bioinformatics Institute [42] and a set of 145 restriction enzymes from the REBase database [43]. Since these clinically important viruses are not very large (with respect to the size of base pairs), we also experiment with large viruses. Viral sequences that are extensively experimented with in this thesis are listed in Table 4.1.

Each algorithm with different types of approaches is compared with previous traditional meta-heuristics with respect to the three objectives by taking average of multiple runs (20 times). Since meta-heuristics solutions of [19, 20] were shown to have outperformed the result of the DP and greedy approach of [17], we limit our comparison with the works of [19, 20].

To analyze the results, a paired two sample t-tests has also been performed. The t-test assesses whether the means of two groups are statistically (significantly) different

Table 4.1: List of Viral Sequences

Virus	Base Pairs	Importance
Polio	7440	Clinically Important
Rubella	9761	Clinically Important
Papilloma	7313	Clinically Important
Mosaic	6366	Clinically Important
HIV	9157	Clinically Important
Hepatitis C	9533	Clinically Important
Bronchitis	27503	Clinically Important
Glossina Pallidipes Salivary Gland Hypertrophy	190032	Large Virus
Shrimp White Spot Syndrome	307287	Large Virus

from each other. To compare two paired values (such as in a before-after situation) where both observations are taken from the same or matched subjects, a paired t-test is applied. A small P Value indicates that the result is significant [44].

Recall that, we have introduced a new biologically meaningful objective of URSPP. We have done our experiments with two sets of objectives in two different programs. Table 4.2 and 4.3 show the two objective sets.

Table 4.2: Objective set 1

Objective Set 1
f_1 = number of inserted restriction enzymes f_2 = number of base pairs changes f_3 = maximum distance between two consecutive restriction sites.

Table 4.3: Objective Set 2

Objective Set 2
f_1 = number of inserted restriction enzymes f_2 = number of base pairs changes f_4= standard deviation of gaps of two consecutive restriction enzymes.

Following the style of [19, 20], we present our results in a condensed form in the Tables 4.5 to 4.58 shown below. In these tables, we use some abbreviated names whose meanings are given in the Table 4.4. Notably, the existing meta-heuristic algorithms from

[19, 20] have not been implemented by us, rather we have collected the software package and codes from the authors of [19, 20].

Table 4.4: Algorithms Names

Abbreviated Name	Explanation
Single State	Single-State Methods
Steady State	Steady-State Algorithms
GA Elit	Genetic Algorithms with Elitism
GA	Basic Genetic Algorithms
NSGA	Non-Dominated Sorting Genetic Algorithm
NSGA-(HC)	Hill Climbing considering each objective then NSGA
NSGA-(SAHC)	Steepest Ascent Hill Climbing considering each objective then NSGA
NSGA-(GA)	Genetic Algorithm considering each objective then NSGA

The convention followed to present the results is as follows. For each objective, the difference between the objective values obtained by the proposed algorithm and that obtained by an existing traditional meta-heuristic algorithm is computed. Then the mean and standard deviation (STD) of these differences are calculated and presented in the tables. For example, Table 4.7 shows the comparison of one of our approach (Hill climbing considering single objective to generate population and apply NSGA with those elite population) with the different meta-heuristics of [19, 20] on Polio virus considering Objective Set 1. Here, for Single State, the first row gives the difference of mean values e.g. Δf_1 = 1.333333 means our approach can insert 1.333333 more restriction sites than the Steady State Method, Δf_2 = -9.73333 denotes our approach causes 9.73333 base changes less than the Steady State Method and Δf_3 = -153.067 indicates our approach provides genome sequence with 153.067 less maximum gap than the Steady State Method.

Recall that, we want to insert larger number of enzymes with lower number of base changes having lower standard deviation (or maximum gap) of gaps. So, our aim is to have higher f_1 and lower f_2 and f_4 (or f_3). Therefore, in Δf_1 column, positive mean value is preferred which means that the proposed algorithm can insert larger number of restriction sites. On the other hand, negative mean values are preferable for Δf_2 and Δf_4 (or Δf_3). To elaborate, negative value in Δf_2 column denotes that the proposed algorithm costs lower in terms of base changes. Again, negative value in Δf_4 (or Δf_3) column denotes that our proposed algorithm can minimize the maximum gap between two

consecutive enzymes (or standard deviation between gaps of two consecutive enzymes). In the tables, along with the mean values and standard deviations of the objective value differences, we also present the confidence interval (C.I.) around the mean value using a 95% confidence level. Additionally, P Values are provided which indicates the statistical significance of our results.

4.2.2 Results Summary and Analysis

Note that, URSPP is an offline problem. The issue of performance is bigger than that of timing. Hence, we did not compare the algorithms from running time point of view.

The experimental results for Polio virus are summarized in Tables 4.5 through 4.10. Among these, Tables 4.5 to 4.7 presents comparison of our proposed algorithms considering Objective Set 1 with the meta-heuristics of [19, 20]. Tables 4.8 to 4.10 presents comparison of our proposed algorithms considering Objective Set 2 with the meta-heuristics of [19, 20]. In these tables we find that, our approaches have provided us with a better synthesized sequence in terms of number of inserted restriction sites and maximum distance between consecutive sites (or in terms of standard deviation of the gaps). For instance, in Table 4.6, against steady state, the mean difference of Δf_1 (mean=6.6, STD=7.048809, P Value=0.002751) is significantly greater than zero. A 95% confidence interval (C.I.=0.114126) around the mean of Δf_1 is 0.114126. Recall that, positive Δf_1, negative Δf_2 and negative Δf_4 (or Δf_3) are preferable.

Similarly, from values of Δf_2 we can say that, local search techniques find solution which is costly in terms of base changes. With respect to Δf_4 (or Δf_3) we also get better results though with lower significance. For example, in case of comparison with Meta-heuristics of [19, 20] the P Value is high in respect of Δf_2 which does not reject the null hypothesis that the mean difference is zero. But when comparison is made considering Δf_1 and Δf_4 (or Δf_3), the P Value is much lower which proves higher significance of mean difference.

From Table 4.11 to 4.16, comparative analyses of our proposed algorithm with the traditional meta-heuristics are shown for Rubella virus. All the mean values of Δf_1 of these tables are positive which indicates that our proposed algorithms can insert more

Table 4.5: Comparison of NSGA-(HC) for Polio virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	3.6	-0.46667	-142.667
	STD	6.208289	21.09796	114.2426
	C.I.	0.100517	0.341593	1.849682
	P Value	0.041378	0.932944	0.000264
Steady State	mean	10.13333	20.73333	-71.8667
	STD	6.738659	21.39582	100.5505
	C.I.	0.109104	0.346416	1.627996
	P Value	4.41E-05	0.00214	0.015097
GA Elit	mean	9.533333	16.46667	-117.133
	STD	5.853774	21.64607	107.5864
	C.I.	0.094777	0.350468	1.741912
	P Value	1.93E-05	0.010624	0.000862
GA	mean	11.33333	26.73333	-99.8667
	STD	6.608076	20.33809	67.07231
	C.I.	0.10699	0.32929	1.085956
	P Value	1.11E-05	0.000164	4.88E-05
NSGA	mean	8.4	24.53333	-102.467
	STD	9.560335	27.85387	122.0965
	C.I.	0.15479	0.450977	1.976842
	P Value	0.004288	0.004217	0.005809

Table 4.6: Comparison of NSGA-(SAHC) for Polio virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	6.6	4.666667	-165.667
	STD	7.048809	18.02644	107.6858
	C.I.	0.114126	0.291863	1.743522
	P Value	0.002751	0.333053	3.5E-05
Steady State	mean	13.13333	25.86667	-94.8667
	STD	7.308182	19.59908	92.25111
	C.I.	0.118325	0.317325	1.493621
	P Value	6.66E-06	0.000158	0.001361
GA Elit	mean	12.53333	21.6	-140.133
	STD	7.558029	17.39787	98.28811
	C.I.	0.122371	0.281686	1.591365
	P Value	1.59E-05	0.000278	7.52E-05
GA	mean	14.33333	31.86667	-122.867
	STD	7.027158	12.2758	59.22821
	C.I.	0.113775	0.198755	0.958953
	P Value	1.59E-06	8.74E-08	1.3E-06
NSGA	mean	11.4	29.66667	-125.467
	STD	6.684737	18.2978	112.6099
	C.I.	0.108231	0.296256	1.823246
	P Value	1.18E-05	2.02E-05	0.000712

Table 4.7: Comparison of NSGA-(GA) for Polio virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	1.333333	-9.73333	-153.067
	STD	8.303757	26.88459	122.1501
	C.I.	0.134445	0.435284	1.97771
	P Value	0.544013	0.182644	0.000256
Steady State	mean	7.866667	11.46667	-82.2667
	STD	10.13387	31.42989	102.3478
	C.I.	0.164076	0.508876	1.657094
	P Value	0.009429	0.179503	0.007631
GA Elit	mean	7.266667	7.2	-127.533
	STD	8.101734	24.7421	104.5008
	C.I.	0.131174	0.400595	1.691953
	P Value	0.003724	0.27867	0.000324
GA	mean	9.066667	17.46667	-110.267
	STD	9.79407	27.95115	72.12634
	C.I.	0.158574	0.452552	1.167785
	P Value	0.002984	0.029695	3.73E-05
NSGA	mean	6.133333	15.26667	-112.867
	STD	9.927355	25.76949	117.8546
	C.I.	0.160732	0.417229	1.908162
	P Value	0.031297	0.037743	0.002335

Table 4.8: Comparison of NSGA-(HC) for Polio virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	2.333333	-3.13333	-14.578
	STD	9.61893	28.74485	10.61146
	C.I.	0.155738	0.465403	0.171808
	P Value	0.363403	0.679309	0.000108
Steady State	mean	9.733333	19.26667	-15.7681
	STD	8.136923	26.73004	12.83846
	C.I.	0.131743	0.432781	0.207865
	P Value	0.000388	0.014417	0.000307
GA Elit	mean	7.933333	14.66667	-17.9199
	STD	8.622783	24.03767	11.23712
	C.I.	0.13961	0.38919	0.181938
	P Value	0.003117	0.033124	2.41E-05
GA	mean	7.666667	14.06667	-13.1814
	STD	8.68222	30.17441	10.82205
	C.I.	0.140572	0.488549	0.175218
	P Value	0.004145	0.092539	0.00033
NSGA	mean	7.2	14.2	-16.0582
	STD	8.081725	22.22033	11.62896
	C.I.	0.13085	0.359765	0.188282
	P Value	0.003901	0.026723	0.000103

Table 4.9: Comparison of NSGA-(SAHC) for Polio virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	0.6	-8.86667	-15.9229
	STD	9.13236	27.03402	12.70719
	C.I.	0.14786	0.437703	0.20574
	P Value	0.802842	0.224695	0.000256
Steady State	mean	8	13.53333	-17.1131
	STD	7.901175	27.21572	14.98934
	C.I.	0.127927	0.440645	0.24269
	P Value	0.001536	0.074672	0.00058
GA Elit	mean	6.2	8.933333	-19.2649
	STD	9.850308	25.14946	13.31954
	C.I.	0.159485	0.40719	0.215654
	P Value	0.028713	0.190519	6.53E-05
GA	mean	5.933333	8.333333	-14.5264
	STD	7.294486	18.67644	10.6705
	C.I.	0.118104	0.302387	0.172764
	P Value	0.007088	0.105951	0.000118
NSGA	mean	5.466667	8.466667	-17.4032
	STD	7.356889	21.54685	14.2027
	C.I.	0.119114	0.348861	0.229953
	P Value	0.012162	0.150307	0.000313

Table 4.10: Comparison of NSGA-(GA) for Polio virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	-3.53333	-19.3333	-12.4485
	STD	6.770384	24.10295	10.01424
	C.I.	0.109618	0.390247	0.162139
	P Value	0.062805	0.007731	0.000275
Steady State	mean	3.866667	3.066667	-13.6387
	STD	5.553206	24.06677	11.91784
	C.I.	0.089911	0.389661	0.19296
	P Value	0.017367	0.629305	0.000568
GA Elit	mean	2.066667	-1.53333	-15.7905
	STD	4.712698	19.93872	12.41623
	C.I.	0.076302	0.322824	0.201029
	P Value	0.111532	0.770198	0.000223
GA	mean	1.8	-2.13333	-11.052
	STD	4.988558	20.18439	8.000348
	C.I.	0.080769	0.326802	0.129532
	P Value	0.184025	0.688481	0.000102
NSGA	mean	1.333333	-2	-13.9288
	STD	4.186145	20.18486	12.42183
	C.I.	0.067777	0.326809	0.20112
	P Value	0.237666	0.706928	0.000675

restriction enzymes than the traditional meta-heuristics. The mean values of Δf_2 are all positive which indicate that our algorithms do not give better results in terms of base pair changes. Again, all values of Δf_4 (or Δf_3) are negative which indicate that our proposed algorithm can give better synthesized sequence in terms of 'standard deviation' or 'maximum gap'. The statistical P Value for Δf_1 is less than 0.01 for most of the values. Every P Values for Δf_4 (or Δf_3) except 3 values (out of 30) are less than 0.01. The statistical values reveal that our algorithms give better results than traditional meta-heuristics with respect to f_1 and f_4 (or f_3).

Table 4.11: Comparison of NSGA-(HC) for Rubella virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	4.466667	56.33333	-145.467
	STD	9.956094	79.85314	115.5947
	C.I.	0.161197	1.292888	1.871572
	P Value	0.104225	0.0162	0.000246
Steady State	mean	13.8	71.8	-104.933
	STD	9.336564	100.6679	192.623
	C.I.	0.151167	1.629896	3.118725
	P Value	5.25E-05	0.01527	0.053352
GA Elit	mean	9.533333	36.66667	-103.733
	STD	8.52615	109.7924	118.8964
	C.I.	0.138045	1.77763	1.92503
	P Value	0.000692	0.216795	0.004496
GA	mean	13	92.93333	-162.6
	STD	10.14889	145.9953	100.9489
	C.I.	0.164319	2.363784	1.634446
	P Value	0.000209	0.027227	2.17E-05
NSGA	mean	11.93333	48.06667	-82.6667
	STD	11.33557	145.0879	76.69389
	C.I.	0.183532	2.349092	1.241737
	P Value	0.001131	0.220294	0.000936

From Table 4.17 to 4.22, comparative analyses of our proposed algorithm with the traditional meta-heuristics are shown for Papilloma virus. All the mean values of Δf_1 of these tables are positive which indicates that our proposed algorithms can insert more restriction enzymes than traditional meta-heuristics. The mean values of Δf_2 are positive for maximum values which indicate that our algorithms do not give better results in terms of base pair changes. Again, all values of Δf_4 (or Δf_3) are negative which indicate that our proposed algorithm can give better synthesized sequence in terms of 'standard deviation' or 'maximum gap'. The statistical P Value for Δf_1 is less than 0.01

Table 4.12: Comparison of NSGA-(SAHC) for Rubella virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	3.933333	74.13333	-159.733
	STD	10.49127	108.4942	112.0245
	C.I.	0.169862	1.756609	1.813769
	P Value	0.168532	0.019164	7.52E-05
Steady State	mean	13.26667	89.6	-119.2
	STD	8.827446	80.39794	196.47
	C.I.	0.142924	1.301709	3.18101
	P Value	4.44E-05	0.000711	0.033978
GA Elit	mean	9	54.46667	-118
	STD	10.16998	121.1993	105.4927
	C.I.	0.16466	1.962316	1.708014
	P Value	0.004084	0.103692	0.000689
GA	mean	12.46667	110.7333	-176.867
	STD	10.05603	135.6511	89.35632
	C.I.	0.162815	2.196302	1.446752
	P Value	0.000282	0.006931	2.24E-06
NSGA	mean	11.4	65.86667	-96.9333
	STD	11.50652	109.7366	62.90863
	C.I.	0.1863	1.776726	1.018542
	P Value	0.001813	0.035642	3.44E-05

Table 4.13: Comparison of NSGA-(GA) for Rubella virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	2.466667	33.66667	-170.267
	STD	11.02508	102.8818	119.7071
	C.I.	0.178505	1.665741	1.938156
	P Value	0.400817	0.225695	7.7E-05
Steady State	mean	11.8	49.13333	-129.733
	STD	9.259744	111.6556	200.4062
	C.I.	0.149923	1.807796	3.244741
	P Value	0.000219	0.110411	0.025116
GA Elit	mean	7.533333	14	-128.533
	STD	10.48718	106.2517	112.4226
	C.I.	0.169796	1.720303	1.820214
	P Value	0.014688	0.617782	0.000573
GA	mean	11	70.26667	-187.4
	STD	11.68638	130.9681	97.21758
	C.I.	0.189212	2.120481	1.574032
	P Value	0.002648	0.056595	3.03E-06
NSGA	mean	9.933333	25.4	-107.467
	STD	10.87899	103.225	63.71462
	C.I.	0.17614	1.671297	1.031592
	P Value	0.003289	0.356754	1.33E-05

Table 4.14: Comparison of NSGA-(HC) for Rubella virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	4.666667	-10.6667	-22.3479
	STD	7.227593	139.2878	16.67327
	C.I.	0.117021	2.255184	0.269954
	P Value	0.025433	0.771132	0.000137
Steady State	mean	10.6	40.66667	-20.7566
	STD	6.544354	120.0183	12.90138
	C.I.	0.105958	1.943194	0.208884
	P Value	2.05E-05	0.210524	2.2E-05
GA Elit	mean	10.26667	-2.73333	-20.1073
	STD	6.408328	104.762	10.71415
	C.I.	0.103756	1.696182	0.173471
	P Value	2.3E-05	0.920944	4.11E-06
GA	mean	9.8	53	-15.3043
	STD	7.437357	164.3255	11.85216
	C.I.	0.120417	2.660564	0.191896
	P Value	0.000161	0.232091	0.000194
NSGA	mean	11.4	51.33333	-21.6189
	STD	6.489552	129.1835	13.30933
	C.I.	0.105071	2.091587	0.215489
	P Value	8.55E-06	0.146097	1.98E-05

Table 4.15: Comparison of NSGA-(SAHC) for Rubella virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	1.4	-36.4667	-25.5017
	STD	7.028513	145.6703	16.49977
	C.I.	0.113797	2.358522	0.267145
	P Value	0.453263	0.348723	3.33E-05
Steady State	mean	7.333333	14.86667	-23.9104
	STD	4.995236	128.4222	13.24752
	C.I.	0.080877	2.079261	0.214488
	P Value	5.62E-05	0.66076	6.34E-06
GA Elit	mean	7	-28.5333	-23.2611
	STD	6.244998	127.2258	11.22538
	C.I.	0.101112	2.05989	0.181748
	P Value	0.000677	0.399708	1.32E-06
GA	mean	6.533333	27.2	-18.4581
	STD	6.937133	153.8107	11.21401
	C.I.	0.112318	2.490322	0.181564
	P Value	0.002638	0.504588	1.73E-05
NSGA	mean	8.133333	25.53333	-24.7728
	STD	5.998412	126.6722	13.23811
	C.I.	0.097119	2.050926	0.214336
	P Value	0.000123	0.447992	4.24E-06

Table 4.16: Comparison of NSGA-(GA) for Rubella virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	-1.6	-47.8667	-20.1766
	STD	10.04845	120.9893	17.18957
	C.I.	0.162693	1.958916	0.278313
	P Value	0.547338	0.14774	0.000457
Steady State	mean	4.333333	3.466667	-18.5853
	STD	8.901578	110.4264	14.1114
	C.I.	0.144124	1.787893	0.228475
	P Value	0.080302	0.904955	0.000161
GA Elit	mean	4	-39.9333	-17.936
	STD	9.118271	110.3226	12.14153
	C.I.	0.147632	1.786213	0.196581
	P Value	0.111422	0.182726	5.28E-05
GA	mean	3.533333	15.8	-13.133
	STD	10.81577	149.8176	12.23949
	C.I.	0.175116	2.42567	0.198167
	P Value	0.226439	0.689121	0.000971
NSGA	mean	5.133333	14.13333	-19.4477
	STD	7.633261	107.1753	14.50486
	C.I.	0.123589	1.735257	0.234846
	P Value	0.020792	0.617495	0.000136

for most of the values (26 out of 30). Every P Values for Δf_4 (or Δf_3) are less than 0.01. The statistical values reveal that our algorithms give better results than traditional meta-heuristics with respect to f_1 and f_4 (or f_3).

Table 4.17: Comparison of NSGA-(HC) for Papilloma virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	4.933333	10.13333	-186.933
	STD	5.812138	21.70868	97.84862
	C.I.	0.094103	0.351481	1.584249
	P Value	0.005396	0.092156	3.36E-06
Steady State	mean	11.6	27.46667	-134.267
	STD	6.577016	20.67734	140.221
	C.I.	0.106487	0.334783	2.270293
	P Value	8.18E-06	0.000149	0.002338
GA Elit	mean	8.933333	11.6	-97.1333
	STD	6.134989	22.66148	73.35225
	C.I.	0.099331	0.366908	1.187633
	P Value	6.1E-05	0.0674	0.000153
GA	mean	9.733333	10.33333	-93.8667
	STD	6.47486	20.11633	85.12165
	C.I.	0.104833	0.3257	1.378189
	P Value	4.43E-05	0.066552	0.000776
NSGA	mean	10.53333	18.2	-93.3333
	STD	8.634372	25.47043	85.49575
	C.I.	0.139798	0.412387	1.384246
	P Value	0.000326	0.015118	0.000843

Table 4.18: Comparison of NSGA-(SAHC) for Papilloma virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	4.266667	9.466667	-194.533
	STD	6.627504	22.87502	94.28139
	C.I.	0.107305	0.370365	1.526493
	P Value	0.025796	0.131293	1.39E-06
Steady State	mean	10.93333	26.8	-141.867
	STD	7.582561	24.62925	141.9053
	C.I.	0.122768	0.398768	2.297564
	P Value	6.73E-05	0.000866	0.001693
GA Elit	mean	8.266667	10.93333	-104.733
	STD	7.638873	16.57221	70.87561
	C.I.	0.12368	0.268318	1.147534
	P Value	0.000906	0.022887	5.27E-05
GA	mean	9.066667	9.666667	-101.467
	STD	7.860086	21.22555	86.1219
	C.I.	0.127261	0.343659	1.394384
	P Value	0.000531	0.099556	0.000443
NSGA	mean	9.866667	17.53333	-100.933
	STD	6.717426	19.03706	84.57496
	C.I.	0.108761	0.308226	1.369338
	P Value	5.6E-05	0.003094	0.000395

Table 4.19: Comparison of NSGA-(GA) for Papilloma virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	2.8	-3.33333	-172.067
	STD	5.143651	22.8619	111.8899
	C.I.	0.08328	0.370153	1.811589
	P Value	0.053505	0.581219	3.51E-05
Steady State	mean	9.466667	14	-119.4
	STD	6.566872	19.23167	145.3512
	C.I.	0.106323	0.311377	2.353355
	P Value	6.74E-05	0.013649	0.006661
GA Elit	mean	6.8	-1.86667	-82.2667
	STD	6.888914	19.40128	94.81898
	C.I.	0.111537	0.314123	1.535197
	P Value	0.001864	0.715002	0.004668
GA	mean	7.6	-3.13333	-79
	STD	7.189278	21.00635	110.0325
	C.I.	0.1164	0.34011	1.781516
	P Value	0.001094	0.572642	0.01473
NSGA	mean	8.4	4.733333	-78.4667
	STD	7.099296	21.6909	101.2718
	C.I.	0.114943	0.351194	1.639674
	P Value	0.000426	0.412237	0.009536

Table 4.20: Comparison of NSGA-(HC) for Papilloma virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	1.733333	-5.26667	-21.3026
	STD	6.005553	20.11562	10.77935
	C.I.	0.097235	0.325688	0.174526
	P Value	0.282456	0.327777	2.28E-06
Steady State	mean	7.933333	4.866667	-17.8847
	STD	7.896352	29.36438	9.228281
	C.I.	0.127848	0.475433	0.149413
	P Value	0.00163	0.531319	2.85E-06
GA Elit	mean	7.133333	0.2	-16.2953
	STD	6.801961	23.38253	10.68997
	C.I.	0.110129	0.378582	0.173079
	P Value	0.001166	0.974041	3.84E-05
GA	mean	9.333333	9.733333	-18.8759
	STD	6.757712	24.26775	9.382686
	C.I.	0.109413	0.392915	0.151913
	P Value	0.000103	0.142641	1.86E-06
NSGA	mean	8.933333	10.53333	-21.2564
	STD	8.075241	29.5535	11.56422
	C.I.	0.130745	0.478495	0.187234
	P Value	0.000756	0.189109	5.18E-06

Table 4.21: Comparison of NSGA-(SAHC) for Papilloma virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	1.666667	1.333333	-23.3089
	STD	6.298148	19.73998	13.39129
	C.I.	0.101972	0.319606	0.216816
	P Value	0.322794	0.797439	9.45E-06
Steady State	mean	7.866667	11.46667	-19.891
	STD	6.128	18.12601	7.98734
	C.I.	0.099217	0.293475	0.129322
	P Value	0.000205	0.028039	1.46E-07
GA Elit	mean	7.066667	6.8	-18.3015
	STD	6.850096	20.65084	12.46137
	C.I.	0.110909	0.334354	0.20176
	P Value	0.001328	0.222955	5.6E-05
GA	mean	9.266667	16.33333	-20.8822
	STD	7.497301	24.58997	12.63411
	C.I.	0.121387	0.398132	0.204557
	P Value	0.00029	0.022128	1.65E-05
NSGA	mean	8.866667	17.13333	-23.2627
	STD	5.853774	13.75223	8.112312
	C.I.	0.094777	0.22266	0.131345
	P Value	4.1E-05	0.00027	2.51E-08

Table 4.22: Comparison of NSGA-(GA) for Papilloma virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	1.533333	-4.2	-25.8555
	STD	5.527421	16.54518	12.83853
	C.I.	0.089493	0.26788	0.207866
	P Value	0.300819	0.342216	1.84E-06
Steady State	mean	7.733333	5.933333	-22.4376
	STD	5.175032	17.84643	7.560121
	C.I.	0.083788	0.288948	0.122405
	P Value	4.7E-05	0.218752	1.62E-08
GA Elit	mean	6.933333	1.266667	-20.8482
	STD	5.063407	19.45128	12.12645
	C.I.	0.081981	0.314932	0.196337
	P Value	0.000112	0.804545	1.08E-05
GA	mean	9.133333	10.8	-23.4288
	STD	4.405624	20.64392	11.63196
	C.I.	0.071331	0.334242	0.188331
	P Value	1.31E-06	0.06224	1.83E-06
NSGA	mean	8.733333	11.6	-25.8094
	STD	2.814926	12.24045	8.416577
	C.I.	0.045576	0.198183	0.136271
	P Value	9.2E-09	0.002521	1.07E-08

Table 4.23: Comparison of NSGA-(HC) for Mosaic virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	4.4	4.066667	-120.4
	STD	7.790104	17.16503	101.8099
	C.I.	0.126128	0.277916	1.648386
	P Value	0.046166	0.374381	0.000428
Steady State	mean	12.06667	25.66667	-77.2667
	STD	7.391759	19.33046	58.78954
	C.I.	0.119679	0.312976	0.951851
	P Value	1.88E-05	0.00015	0.000165
GA Elit	mean	11.06667	18.6	-82.2
	STD	8.572936	22.90602	87.14372
	C.I.	0.138803	0.370867	1.410928
	P Value	0.000195	0.007164	0.002608
GA	mean	13.6	22.86667	-60.1333
	STD	8.432929	21.40049	65.50558
	C.I.	0.136536	0.346491	1.060589
	P Value	2.14E-05	0.001004	0.003167
NSGA	mean	10.93333	24.13333	-76
	STD	8.362046	23.25285	67.2798
	C.I.	0.135388	0.376483	1.089315
	P Value	0.000173	0.001266	0.000635

Table 4.24: Comparison of NSGA-(SAHC) for Mosaic virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	4.533333	5.533333	-129.133
	STD	8.166715	19.6682	92.55953
	C.I.	0.132226	0.318444	1.498615
	P Value	0.049527	0.294289	9.3E-05
Steady State	mean	12.2	27.13333	-86
	STD	7.513797	24.02935	48.23455
	C.I.	0.121655	0.389055	0.780957
	P Value	1.99E-05	0.000637	7.26E-06
GA Elit	mean	11.2	20.06667	-90.9333
	STD	7.884886	24.04599	79.0167
	C.I.	0.127663	0.389324	1.279345
	P Value	7.8E-05	0.006024	0.000542
GA	mean	13.73333	24.33333	-68.8667
	STD	7.362712	21.47313	57.40068
	C.I.	0.119208	0.347668	0.929364
	P Value	4.4E-06	0.000618	0.000377
NSGA	mean	11.06667	25.6	-84.7333
	STD	9.931671	24.75537	56.50849
	C.I.	0.160802	0.40081	0.914919
	P Value	0.000712	0.001303	4.54E-05

Table 4.25: Comparison of NSGA-(GA) for Mosaic virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	3.666667	1.933333	-138.667
	STD	6.477066	16.79484	85.02997
	C.I.	0.104869	0.271922	1.376705
	P Value	0.045739	0.662532	1.9E-05
Steady State	mean	11.33333	23.53333	-95.5333
	STD	7.047458	24.61668	42.0966
	C.I.	0.114104	0.398564	0.681579
	P Value	2.21E-05	0.002365	4.51E-07
GA Elit	mean	10.33333	16.46667	-100.467
	STD	6.16055	18.6619	73.25091
	C.I.	0.099744	0.302152	1.185992
	P Value	1.41E-05	0.004166	0.00011
GA	mean	12.86667	20.73333	-78.4
	STD	5.262627	15.50791	51.46122
	C.I.	0.085206	0.251086	0.833199
	P Value	1.83E-07	0.00014	3.87E-05
NSGA	mean	10.2	22	-94.2667
	STD	7.032577	19.55943	52.55673
	C.I.	0.113863	0.316683	0.850937
	P Value	6.35E-05	0.000658	6.8E-06

Table 4.26: Comparison of NSGA-(HC) for Mosaic virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	2.6	7.866667	-18.5029
	STD	5.901574	14.9803	9.214369
	C.I.	0.095551	0.242543	0.149188
	P Value	0.11003	0.061373	1.9E-06
Steady State	mean	5.333333	6.666667	-10.688
	STD	8.173709	20.93414	7.42064
	C.I.	0.132339	0.338941	0.120146
	P Value	0.024166	0.237739	6.8E-05
GA Elit	mean	7.4	11.93333	-12.9213
	STD	4.852098	19.02054	7.68836
	C.I.	0.078559	0.307958	0.124481
	P Value	3.82E-05	0.02915	1.38E-05
GA	mean	9.533333	14.6	-16.9105
	STD	7.909006	23.78114	10.01884
	C.I.	0.128053	0.385036	0.162213
	P Value	0.000362	0.032212	1.32E-05
NSGA	mean	7.466667	15.86667	-16.1295
	STD	7.86372	23.06781	11.79627
	C.I.	0.12732	0.373487	0.190991
	P Value	0.002486	0.018518	0.000113

Table 4.27: Comparison of NSGA-(SAHC) for Mosaic virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	0.333333	-1.93333	-16.7512
	STD	4.592955	22.06635	10.54646
	C.I.	0.074364	0.357272	0.170756
	P Value	0.782759	0.739398	2.51E-05
Steady State	mean	3.066667	-3.13333	-8.93633
	STD	6.659758	21.3604	8.880483
	C.I.	0.107827	0.345842	0.143782
	P Value	0.096203	0.57895	0.00161
GA Elit	mean	5.133333	2.133333	-11.1696
	STD	6.947422	24.53239	8.810454
	C.I.	0.112484	0.3972	0.142648
	P Value	0.012557	0.741268	0.00023
GA	mean	7.266667	4.8	-15.1588
	STD	7.814516	22.08814	10.86969
	C.I.	0.126523	0.357625	0.175989
	P Value	0.00289	0.414134	9.34E-05
NSGA	mean	5.2	6.066667	-14.3777
	STD	6.888914	21.46581	12.87474
	C.I.	0.111537	0.347549	0.208453
	P Value	0.011114	0.292174	0.000699

Table 4.28: Comparison of NSGA-(GA) for Mosaic virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	0	2.933333	-15.4755
	STD	7.483315	22.46097	11.16324
	C.I.	0.121161	0.363662	0.180742
	P Value	1	0.620867	9.9E-05
Steady State	mean	2.733333	1.733333	-7.6606
	STD	10.53204	27.38735	9.967465
	C.I.	0.170522	0.443424	0.161381
	P Value	0.331888	0.80992	0.010004
GA Elit	mean	4.8	7	-9.89389
	STD	7.408489	21.4576	9.235144
	C.I.	0.11995	0.347416	0.149525
	P Value	0.025012	0.227058	0.000983
GA	mean	6.933333	9.666667	-13.8831
	STD	8.737985	23.40228	11.90527
	C.I.	0.141475	0.378902	0.192756
	P Value	0.008262	0.131961	0.000484
NSGA	mean	4.866667	10.93333	-13.102
	STD	10.11976	29.24397	14.03677
	C.I.	0.163847	0.473484	0.227267
	P Value	0.083644	0.169646	0.002813

Table 4.29: Comparison of NSGA-(HC) for HIV virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	0.733333	-3.46667	-260.867
	STD	7.563698	17.56566	220.8713
	C.I.	0.122462	0.284402	3.576087
	P Value	0.712915	0.457348	0.000433
Steady State	mean	7.866667	9.933333	-123.267
	STD	7.443757	14.22506	183.945
	C.I.	0.120521	0.230315	2.978221
	P Value	0.001097	0.017105	0.021166
GA Elit	mean	6.2	7.666667	-155.067
	STD	6.646159	18.22348	166.3548
	C.I.	0.107607	0.295053	2.693421
	P Value	0.002825	0.12552	0.002841
GA	mean	7.8	11.26667	-130
	STD	8.436316	17.81038	116.5614
	C.I.	0.136591	0.288365	1.887225
	P Value	0.003011	0.028043	0.000706
NSGA	mean	5.266667	4.733333	-67.8667
	STD	8.447034	16.10886	111.4571
	C.I.	0.136764	0.260816	1.804583
	P Value	0.030007	0.274215	0.033432

Table 4.30: Comparison of NSGA-(SAHC) for HIV virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	-0.2	-5.26667	-232.067
	STD	7.542262	20.3837	205.9148
	C.I.	0.122115	0.330029	3.33393
	P Value	0.919657	0.333961	0.000647
Steady State	mean	6.933333	8.133333	-94.4667
	STD	7.620149	22.29627	176.9713
	C.I.	0.123376	0.360995	2.86531
	P Value	0.003371	0.179558	0.057704
GA Elit	mean	5.266667	5.866667	-126.267
	STD	8.004166	22.52258	153.5241
	C.I.	0.129594	0.364659	2.485681
	P Value	0.02319	0.330174	0.00661
GA	mean	6.866667	9.466667	-101.2
	STD	8.95119	23.79636	109.9923
	C.I.	0.144927	0.385283	1.780866
	P Value	0.010115	0.145672	0.003117
NSGA	mean	4.333333	2.933333	-39.0667
	STD	8.557926	19.4182	98.23403
	C.I.	0.13856	0.314397	1.590489
	P Value	0.070071	0.56782	0.145794

Table 4.31: Comparison of NSGA-(GA) for HIV virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	2.933333	0.066667	-282.867
	STD	4.25049	11.44219	194.0909
	C.I.	0.068819	0.185259	3.14249
	P Value	0.0182	0.982315	6.05E-05
Steady State	mean	10.06667	13.46667	-145.267
	STD	5.270764	15.30577	161.6197
	C.I.	0.085338	0.247813	2.616755
	P Value	3.37E-06	0.004248	0.00367
GA Elit	mean	8.4	11.2	-177.067
	STD	3.418437	11.1047	138.7564
	C.I.	0.055347	0.179794	2.246581
	P Value	1.72E-07	0.001582	0.000217
GA	mean	10	14.8	-152
	STD	5.818689	12.75707	89.66286
	C.I.	0.094209	0.206547	1.451715
	P Value	1.09E-05	0.000506	1.26E-05
NSGA	mean	7.466667	8.266667	-89.8667
	STD	5.221749	14.27519	93.79298
	C.I.	0.084544	0.231127	1.518585
	P Value	7.31E-05	0.041613	0.002327

Table 4.32: Comparison of NSGA-(HC) for HIV virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	2.666667	-0.2	-25.0522
	STD	5.232681	16.12097	11.72749
	C.I.	0.084721	0.261012	0.189878
	P Value	0.068482	0.962356	9.24E-07
Steady State	mean	10.6	13.26667	-17.0674
	STD	4.188419	12.43536	8.403189
	C.I.	0.067814	0.201339	0.136055
	P Value	1.2E-07	0.001017	1.67E-06
GA Elit	mean	8.533333	9.8	-20.4406
	STD	5.054936	14.64436	13.87968
	C.I.	0.081844	0.237104	0.224723
	P Value	1.32E-05	0.021315	5.45E-05
GA	mean	10.66667	12.26667	-24.8785
	STD	5.740416	14.0939	12.29904
	C.I.	0.092942	0.228192	0.199132
	P Value	4.59E-06	0.00457	1.75E-06
NSGA	mean	7.666667	13.86667	-16.2189
	STD	6.454972	18.48886	14.16939
	C.I.	0.104511	0.29935	0.229414
	P Value	0.000412	0.011533	0.000567

Table 4.33: Comparison of NSGA-(SAHC) for HIV virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	-0.66667	-8.8	-21.4282
	STD	4.237025	10.5911	10.69218
	C.I.	0.068601	0.171479	0.173115
	P Value	0.55203	0.006195	1.94E-06
Steady State	mean	7.266667	4.666667	-13.4434
	STD	4.300609	15.15004	9.679617
	C.I.	0.06963	0.245292	0.156721
	P Value	1.3E-05	0.252698	9.72E-05
GA Elit	mean	5.2	1.2	-16.8166
	STD	6.559617	19.10946	13.84882
	C.I.	0.106206	0.309398	0.224224
	P Value	0.008309	0.811371	0.000339
GA	mean	7.333333	3.666667	-21.2545
	STD	5.498918	15.38862	12.93027
	C.I.	0.089032	0.249154	0.209352
	P Value	0.000143	0.371732	1.75E-05
NSGA	mean	4.333333	5.266667	-12.5948
	STD	5.851333	18.99799	14.53067
	C.I.	0.094738	0.307593	0.235264
	P Value	0.012396	0.301126	0.004698

Table 4.34: Comparison of NSGA-(GA) for HIV virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	-0.6	-9.4	-25.6067
	STD	4.032015	15.81048	16.89731
	C.I.	0.065282	0.255985	0.273581
	P Value	0.573539	0.037163	4.08E-05
Steady State	mean	7.333333	4.066667	-17.6218
	STD	5.380742	16.67105	13.93094
	C.I.	0.087119	0.269918	0.225553
	P Value	0.000117	0.360802	0.000235
GA Elit	mean	5.266667	0.6	-20.995
	STD	6.076496	15.8601	16.90752
	C.I.	0.098383	0.256788	0.273747
	P Value	0.0047	0.885601	0.000278
GA	mean	7.4	3.066667	-25.433
	STD	6.84314	17.28941	17.03318
	C.I.	0.110796	0.27993	0.275781
	P Value	0.000911	0.503329	4.74E-05
NSGA	mean	4.4	4.666667	-16.7733
	STD	5.275279	14.28119	16.59362
	C.I.	0.085411	0.231224	0.268664
	P Value	0.006044	0.226322	0.001555

Table 4.35: Comparison of NSGA-(HC) for Hepatitis C virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	3.733333	1.066667	-310.867
	STD	6.850096	20.20066	211.7609
	C.I.	0.110909	0.327065	3.428583
	P Value	0.053258	0.8409	5.63E-05
Steady State	mean	10.26667	20.93333	-207.2
	STD	6.974102	23.37724	189.4285
	C.I.	0.112916	0.378497	3.067003
	P Value	5.47E-05	0.003767	0.00083
GA Elit	mean	8.733333	13.86667	-227.6
	STD	6.984336	20.68425	141.2257
	C.I.	0.113082	0.334895	2.286559
	P Value	0.000261	0.021123	2.16E-05
GA	mean	10.13333	17.8	-237.667
	STD	7.558029	19.66941	215.1769
	C.I.	0.122371	0.318464	3.48389
	P Value	0.000136	0.003501	0.000766
NSGA	mean	8.6	17.2	-189.2
	STD	6.780223	20.67158	137.0522
	C.I.	0.109777	0.33469	2.218988
	P Value	0.000229	0.006139	0.000103

Table 4.36: Comparison of NSGA-(SAHC) for Hepatitis C virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	4.466667	7.2	-315.733
	STD	5.680376	19.60394	219.9849
	C.I.	0.09197	0.317404	3.561736
	P Value	0.008728	0.176795	7.05E-05
Steady State	mean	11	27.06667	-212.067
	STD	6.047432	22.1439	192.8765
	C.I.	0.097913	0.358528	3.122829
	P Value	5.82E-06	0.00032	0.000795
GA Elit	mean	9.466667	20	-232.467
	STD	4.96943	20.57044	143.1472
	C.I.	0.080459	0.333052	2.317671
	P Value	3.47E-06	0.002088	1.99E-05
GA	mean	10.86667	23.93333	-242.533
	STD	4.778922	11.27239	205.1967
	C.I.	0.077375	0.182509	3.322303
	P Value	4.4E-07	9.93E-07	0.00043
NSGA	mean	9.333333	23.33333	-194.067
	STD	6.275424	23.06409	139.0891
	C.I.	0.101604	0.373427	2.251967
	P Value	4.94E-05	0.001545	9.3E-05

Table 4.37: Comparison of NSGA-(GA) for Hepatitis C virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	0.733333	-3.93333	-347.6
	STD	5.257195	17.89839	204.765
	C.I.	0.085118	0.28979	3.315313
	P Value	0.597516	0.409023	1.24E-05
Steady State	mean	7.266667	15.93333	-243.933
	STD	9.261801	30.54614	187.0147
	C.I.	0.149956	0.494567	3.027921
	P Value	0.008846	0.062923	0.000177
GA Elit	mean	5.733333	8.866667	-264.333
	STD	7.610769	23.79936	144.3649
	C.I.	0.123225	0.385331	2.337387
	P Value	0.011244	0.171044	5.41E-06
GA	mean	7.133333	12.8	-274.4
	STD	8.975098	24.87741	207.092
	C.I.	0.145314	0.402786	3.35299
	P Value	0.008179	0.066157	0.000153
NSGA	mean	5.6	12.2	-225.933
	STD	7.790104	22.53632	127.395
	C.I.	0.126128	0.364882	2.06263
	P Value	0.01463	0.054673	7.7E-06

Table 4.38: Comparison of NSGA-(HC) for Hepatitis C virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	3.866667	10.93333	-31.4812
	STD	7.029191	21.09999	20.66382
	C.I.	0.113808	0.341626	0.334564
	P Value	0.051348	0.064477	3.86E-05
Steady State	mean	10.66667	24.06667	-32.1434
	STD	7.077799	24.27658	23.49456
	C.I.	0.114595	0.393058	0.380396
	P Value	4.32E-05	0.001805	0.000112
GA Elit	mean	9.733333	20.73333	-27.806
	STD	7.932453	25.8055	18.32525
	C.I.	0.128433	0.417812	0.296701
	P Value	0.000309	0.007652	4.03E-05
GA	mean	10.86667	25.46667	-33.9358
	STD	6.780926	20.85962	25.81331
	C.I.	0.109789	0.337734	0.417939
	P Value	2.29E-05	0.000323	0.000164
NSGA	mean	9.733333	25.86667	-28.7145
	STD	8.697016	29.57766	21.75567
	C.I.	0.140812	0.478887	0.352242
	P Value	0.000686	0.004425	0.000158

Table 4.39: Comparison of NSGA-(SAHC) for Hepatitis C virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	4.4	9.266667	-31.4098
	STD	4.777925	18.94528	17.94244
	C.I.	0.077359	0.30674	0.290503
	P Value	0.003097	0.07902	8.88E-06
Steady State	mean	11.2	22.4	-32.072
	STD	4.329302	13.99898	18.5103
	C.I.	0.070095	0.226655	0.299697
	P Value	9.12E-08	2.33E-05	9.93E-06
GA Elit	mean	10.26667	19.06667	-27.7346
	STD	6.261751	17.88242	13.33012
	C.I.	0.101383	0.289531	0.215826
	P Value	1.8E-05	0.001022	1.26E-06
GA	mean	11.4	23.8	-33.8645
	STD	5.526559	20.33013	22.66483
	C.I.	0.08948	0.329162	0.366962
	P Value	1.39E-06	0.000468	4.71E-05
NSGA	mean	10.26667	24.2	-28.6431
	STD	7.045431	26.29286	18.96655
	C.I.	0.114071	0.425703	0.307084
	P Value	6.06E-05	0.003109	4.23E-05

Table 4.40: Comparison of NSGA-(GA) for Hepatitis C virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	5.133333	14.33333	-33.932
	STD	5.208052	15.90448	16.8636
	C.I.	0.084323	0.257507	0.273035
	P Value	0.001885	0.003603	1.86E-06
Steady State	mean	11.93333	27.46667	-34.5942
	STD	4.233652	14.57428	16.71076
	C.I.	0.068546	0.235969	0.270561
	P Value	3.12E-08	3.92E-06	1.33E-06
GA Elit	mean	11	24.13333	-30.2568
	STD	5.451081	19.94588	14.21424
	C.I.	0.088257	0.32294	0.23014
	P Value	1.79E-06	0.00035	9.63E-07
GA	mean	12.13333	28.86667	-36.3867
	STD	5.938574	18.81438	20.2831
	C.I.	0.09615	0.30462	0.3284
	P Value	1.55E-06	3.6E-05	6.79E-06
NSGA	mean	11	29.26667	-31.1653
	STD	6.129554	22.2051	18.28797
	C.I.	0.099243	0.359519	0.296097
	P Value	6.76E-06	0.00016	1.19E-05

Table 4.41: Comparison of NSGA-(HC) for Bronchitis virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	0.933333	-10.9333	-17.5333
	STD	1.980861	9.160994	67.90631
	C.I.	0.032072	0.148324	1.099459
	P Value	0.089431	0.000395	0.334282
Steady State	mean	6.133333	2.2	-144.333
	STD	3.41983	12.61632	361.5095
	C.I.	0.05537	0.204269	5.853135
	P Value	6.81E-06	0.510448	0.144335
GA Elit	mean	3	0.2	-20.2
	STD	2.13809	8.849536	70.1144
	C.I.	0.034617	0.143281	1.13521
	P Value	8.8E-05	0.93149	0.28329
GA	mean	5.066667	-2.6	-226.333
	STD	2.865227	6.822233	450.1717
	C.I.	0.04639	0.110458	7.288649
	P Value	7.95E-06	0.162074	0.071852
NSGA	mean	1.2	1.533333	0
	STD	2.366432	8.724896	0
	C.I.	0.038314	0.141263	NAN
	P Value	0.069708	0.507196	NAN

From Tables 4.53 to 4.58, a large virus, namely Shrimp White Spot Syndrome viral sequence is examined with our proposed algorithms with the traditional meta-heuristics. All the mean values of Δf_1 of these tables are positive which indicates that our proposed algorithms can insert more restriction enzymes that traditional meta-heuristics. The mean values of Δf_2 are all positive which indicate that our algorithms do not give better results in terms of base pair changes. Again, all values of Δf_4 (or Δf_3) are negative which indicates that our proposed algorithms can give better synthesized sequence in terms of 'standard deviation' or 'maximum gap'. The statistical P Value for Δf_1 and Δf_4 (or Δf_3) is less than 0.01 for every value. This statistics reveals that our algorithms give better results than traditional meta-heuristics with respect to f_1 and f_4 (or f_3).

In Table 4.59, initial experimental results with polio virus are shown. The mean values of this table reveal that it can reconstruct a sequence with less base pair change and minimize the maximum gap comparing the traditional meta-heuristics. But it can not insert much enzymes. The statistical results reveal that these results are not good in terms of statistics.

Initial experiments results of Goal Programming with Polio Virus are shown in Ta-

Table 4.42: Comparison of NSGA-(SAHC) for Bronchitis virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	-0.4	-13.2667	-17.5333
	STD	3.621365	11.83498	67.90631
	C.I.	0.058633	0.191618	1.099459
	P Value	0.675309	0.000677	0.334282
Steady State	mean	4.8	-0.13333	-144.333
	STD	4.570089	13.08143	361.5095
	C.I.	0.073993	0.211799	5.853135
	P Value	0.001152	0.969069	0.144335
GA Elit	mean	1.666667	-2.13333	-20.2
	STD	5.094348	11.50693	70.1144
	C.I.	0.082482	0.186307	1.13521
	P Value	0.225798	0.484547	0.28329
GA	mean	3.733333	-4.93333	-226.333
	STD	3.692979	8.084082	450.1717
	C.I.	0.059792	0.130888	7.288649
	P Value	0.001554	0.0331	0.071852
NSGA	mean	-0.13333	-0.8	0
	STD	4.120795	11.1304	0
	C.I.	0.066719	0.18021	NAN
	P Value	0.902056	0.784796	NAN

Table 4.43: Comparison of NSGA-(GA) for Bronchitis virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	-1.33333	-17.2	-17.5333
	STD	3.154739	10.63149	67.90631
	C.I.	0.051078	0.172133	1.099459
	P Value	0.123926	2.07E-05	0.334282
Steady State	mean	3.866667	-4.06667	-144.333
	STD	4.323799	12.67431	361.5095
	C.I.	0.070006	0.205207	5.853135
	P Value	0.003801	0.234397	0.144335
GA Elit	mean	0.733333	-6.06667	-20.2
	STD	4.317186	10.20131	70.1144
	C.I.	0.069899	0.165168	1.13521
	P Value	0.521287	0.037121	0.28329
GA	mean	2.8	-8.86667	-226.333
	STD	4.021371	7.845533	450.1717
	C.I.	0.065109	0.127026	7.288649
	P Value	0.017369	0.000632	0.071852
NSGA	mean	-1.06667	-4.73333	0
	STD	3.673587	9.931671	0
	C.I.	0.059478	0.160802	NAN
	P Value	0.279686	0.086168	NAN

Table 4.44: Comparison of NSGA-(HC) for Bronchitis virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	-0.33333	-11.2	-64.0956
	STD	3.1773	9.711554	30.8741
	C.I.	0.051443	0.157238	0.499877
	P Value	0.690652	0.000532	1.29E-06
Steady State	mean	4.733333	-1.46667	-82.0505
	STD	3.575046	10.50079	49.60455
	C.I.	0.057883	0.170016	0.803138
	P Value	0.000154	0.597047	1.64E-05
GA Elit	mean	3.4	-0.73333	-59.8431
	STD	3.089152	6.238895	31.29442
	C.I.	0.050016	0.101013	0.506682
	P Value	0.000788	0.655917	3.32E-06
GA	mean	4	-1.93333	-60.3888
	STD	3.644957	7.459095	38.79212
	C.I.	0.059015	0.120769	0.628076
	P Value	0.000808	0.33249	3.1E-05
NSGA	mean	3.866667	1.333333	-80.3095
	STD	3.377799	8.251984	46.96404
	C.I.	0.054689	0.133606	0.760386
	P Value	0.000567	0.541523	1.15E-05

Table 4.45: Comparison of NSGA-(SAHC) for Bronchitis virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	0	-7.8	-66.5038
	STD	3.891382	12.76267	27.99707
	C.I.	0.063005	0.206638	0.453296
	P Value	1	0.03288	2.6E-07
Steady State	mean	5.066667	1.933333	-84.4588
	STD	3.788454	12.44684	41.03692
	C.I.	0.061338	0.201525	0.664421
	P Value	0.00014	0.557071	1.43E-06
GA Elit	mean	3.733333	2.666667	-62.2513
	STD	2.737743	9.714986	38.78676
	C.I.	0.044326	0.157294	0.62799
	P Value	0.000116	0.305733	2.25E-05
GA	mean	4.333333	1.466667	-62.797
	STD	2.94392	9.295672	33.56267
	C.I.	0.047664	0.150505	0.543407
	P Value	5.48E-05	0.550942	4.25E-06
NSGA	mean	4.2	4.733333	-82.7177
	STD	3.144156	11.85909	52.54541
	C.I.	0.050906	0.192008	0.850753
	P Value	0.000141	0.144447	2.76E-05

Table 4.46: Comparison of NSGA-(GA) for Bronchitis virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	-1.26667	-11.9333	-69.7987
	STD	4.667007	12.62914	36.38511
	C.I.	0.075563	0.204476	0.589105
	P Value	0.310991	0.002575	3.2E-06
Steady State	mean	3.8	-2.2	-87.7536
	STD	4.427189	13.09962	57.86997
	C.I.	0.07168	0.212094	0.936962
	P Value	0.005014	0.525937	4.05E-05
GA Elit	mean	2.466667	-1.46667	-65.5462
	STD	3.226379	9.233995	41.5551
	C.I.	0.052238	0.149506	0.672811
	P Value	0.010318	0.548319	2.7E-05
GA	mean	3.066667	-2.66667	-66.0918
	STD	3.807261	10.02616	34.69774
	C.I.	0.061643	0.162332	0.561785
	P Value	0.007533	0.320431	3.47E-06
NSGA	mean	2.933333	0.6	-86.0125
	STD	3.127451	10.21763	49.64497
	C.I.	0.050636	0.165432	0.803793
	P Value	0.002717	0.823378	9.94E-06

Table 4.47: Comparison of NSGA-(HC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	5.8	93.06667	-6101.2
	STD	8.010707	196.9501	3222.181
	C.I.	0.1297	3.188784	52.16975
	P Value	0.014065	0.088599	3.71E-06
Steady State	mean	9.933333	136.2667	-451.067
	STD	7.629517	195.6551	1177.011
	C.I.	0.123528	3.167817	19.05678
	P Value	0.00018	0.017345	0.159907
GA Elit	mean	7.4	81.33333	-1340
	STD	7.038669	133.4994	1384.511
	C.I.	0.113962	2.161465	22.41636
	P Value	0.001143	0.033349	0.00216
GA	mean	11.4	160.8667	-1307.4
	STD	7.365945	168.57	1092.061
	C.I.	0.119261	2.729286	17.68137
	P Value	3.29E-05	0.002396	0.000385
NSGA	mean	5.666667	43.86667	-1143.87
	STD	6.914443	150.3262	1422.017
	C.I.	0.111951	2.433905	23.02363
	P Value	0.00676	0.277394	0.007596

Table 4.48: Comparison of NSGA-(SAHC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	6.733333	127.7333	-6115.53
	STD	3.990465	166.4471	3039.13
	C.I.	0.064609	2.694915	49.206
	P Value	1.32E-05	0.010093	1.85E-06
Steady State	mean	10.86667	170.9333	-465.4
	STD	4.580497	158.5918	1041.945
	C.I.	0.074162	2.567731	16.86994
	P Value	2.64E-07	0.000936	0.105617
GA Elit	mean	8.333333	116	-1354.33
	STD	4.117327	136.6173	1326.559
	C.I.	0.066663	2.211946	21.47808
	P Value	1.73E-06	0.005384	0.00144
GA	mean	12.33333	195.5333	-1321.73
	STD	4.386125	139.0837	842.7289
	C.I.	0.071015	2.251879	13.64447
	P Value	3.22E-08	8.63E-05	2.87E-05
NSGA	mean	6.6	78.53333	-1158.2
	STD	4.102264	164.5218	1346.703
	C.I.	0.066419	2.663742	21.80423
	P Value	2.2E-05	0.085723	0.004949

Table 4.49: Comparison of NSGA-(GA) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	2.6	74.33333	-6729.8
	STD	5.840744	138.9284	3123.043
	C.I.	0.094566	2.249364	50.56463
	P Value	0.106698	0.057192	8.34E-07
Steady State	mean	6.733333	117.5333	-1079.67
	STD	6.19293	139.7057	985.1131
	C.I.	0.100269	2.261951	15.94979
	P Value	0.000872	0.005717	0.000816
GA Elit	mean	4.2	62.6	-1968.6
	STD	6.00238	75.85776	1353.911
	C.I.	0.097183	1.228199	21.92093
	P Value	0.016921	0.006471	6.19E-05
GA	mean	8.2	142.1333	-1936
	STD	5.771853	113.1206	922.3367
	C.I.	0.093451	1.831515	14.93339
	P Value	7.79E-05	0.00025	1.14E-06
NSGA	mean	2.466667	25.13333	-1772.47
	STD	7.049485	154.5121	1419.926
	C.I.	0.114137	2.501677	22.98977
	P Value	0.196822	0.538848	0.000265

Table 4.50: Comparison of NSGA-(HC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	7.666667	86.73333	-701.697
	STD	7.603257	137.8392	327.8959
	C.I.	0.123103	2.231729	5.308904
	P Value	0.001585	0.028753	9.05E-07
Steady State	mean	8.4	41.2	-255.235
	STD	6.695414	134.1812	194.8704
	C.I.	0.108404	2.172504	3.155112
	P Value	0.000253	0.254144	0.00017
GA Elit	mean	7.466667	64.86667	-259.701
	STD	5.343443	137.2791	261.1531
	C.I.	0.086515	2.222661	4.228282
	P Value	9.16E-05	0.088613	0.001763
GA	mean	12.33333	85.6	-400.274
	STD	6.410557	152.6611	222.6374
	C.I.	0.103792	2.471708	3.604683
	P Value	3.1E-06	0.047557	6.62E-06
NSGA	mean	11.53333	92.46667	-403.969
	STD	7.140095	135.8575	405.8021
	C.I.	0.115604	2.199645	6.570269
	P Value	2.11E-05	0.019556	0.001749

Table 4.51: Comparison of NSGA-(SAHC) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	6.466667	108.7333	-758.307
	STD	5.221749	116.8806	348.2609
	C.I.	0.084544	1.892392	5.63863
	P Value	0.000285	0.002881	7.37E-07
Steady State	mean	7.2	63.2	-311.844
	STD	5.918494	105.0552	178.7413
	C.I.	0.095825	1.70093	2.893969
	P Value	0.000334	0.035286	9.21E-06
GA Elit	mean	6.266667	86.86667	-316.31
	STD	5.270764	111.3751	244.3976
	C.I.	0.085338	1.803254	3.956997
	P Value	0.000409	0.009167	0.00019
GA	mean	11.13333	107.6	-456.884
	STD	5.617151	123.4566	222.3141
	C.I.	0.090946	1.998864	3.599448
	P Value	2.21E-06	0.004528	1.45E-06
NSGA	mean	10.33333	114.4667	-460.578
	STD	4.760952	84.36897	399.5436
	C.I.	0.077084	1.366003	6.468938
	P Value	7.66E-07	0.000122	0.000534

Table 4.52: Comparison of NSGA-(GA) for Glossina Pallidipes Salivary Gland Hypertrophy virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	7.133333	140.3333	-841.218
	STD	7.089899	141.6503	313.3317
	C.I.	0.114791	2.293435	5.073097
	P Value	0.001612	0.001814	5.75E-08
Steady State	mean	7.866667	94.8	-394.756
	STD	6.356849	155.1222	164.3602
	C.I.	0.102923	2.511555	2.661126
	P Value	0.000286	0.032886	2.27E-07
GA Elit	mean	6.933333	118.4667	-399.222
	STD	5.775152	146.1075	219.2174
	C.I.	0.093504	2.365601	3.54931
	P Value	0.000375	0.00723	5.74E-06
GA	mean	11.8	139.2	-539.795
	STD	5.198901	108.9772	170.1621
	C.I.	0.084174	1.76443	2.755065
	P Value	4.5E-07	0.000215	6.91E-09
NSGA	mean	11	146.0667	-543.49
	STD	4.105745	74.13353	344.2347
	C.I.	0.066475	1.200283	5.573443
	P Value	5.9E-08	2.36E-06	2.67E-05

Table 4.53: Comparison of NSGA-(HC) for Shrimp White Spot Syndrome virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	6.4	317	-10323.7
	STD	5.329165	419.9143	4697.395
	C.I.	0.086284	6.798757	76.05467
	P Value	0.000374	0.011107	6.6E-07
Steady State	mean	8.933333	357.9333	-1317.67
	STD	6.318529	375.4164	1565.258
	C.I.	0.102302	6.0783	25.34281
	P Value	8.17E-05	0.002412	0.005694
GA Elit	mean	5	244.0667	-1566.2
	STD	6.256425	367.812	1418.56
	C.I.	0.101297	5.955177	22.96764
	P Value	0.007907	0.022239	0.000768
GA	mean	8.666667	405.1333	-1327.87
	STD	6.410557	436.0506	1867.419
	C.I.	0.103792	7.060017	30.23504
	P Value	0.000126	0.002908	0.015524
NSGA	mean	5.733333	251	-637.733
	STD	4.697517	286.3178	1339.118
	C.I.	0.076057	4.635719	21.68143
	P Value	0.000324	0.004354	0.08638

Table 4.54: Comparison of NSGA-(SAHC) for Shrimp White Spot Syndrome virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	8.266667	459.2	-10083.7
	STD	4.131182	337.7167	5168.292
	C.I.	0.066887	5.467911	83.67888
	P Value	1.98E-06	0.000119	2.64E-06
Steady State	mean	10.8	500.1333	-1077.6
	STD	6.981608	320.66	2003.521
	C.I.	0.113038	5.191749	32.43865
	P Value	3.3E-05	3.04E-05	0.056057
GA Elit	mean	6.866667	386.2667	-1326.13
	STD	3.398879	238.0175	1844.697
	C.I.	0.055031	3.853699	29.86716
	P Value	1.77E-06	2E-05	0.014627
GA	mean	10.53333	547.3333	-1087.8
	STD	3.481926	338.5245	2222.978
	C.I.	0.056375	5.480989	35.99184
	P Value	1.27E-08	2.08E-05	0.078901
NSGA	mean	7.6	393.2	-397.667
	STD	4.837355	346.0329	1859.185
	C.I.	0.078321	5.602556	30.10173
	P Value	2.81E-05	0.000604	0.42134

Table 4.55: Comparison of NSGA-(GA) for Shrimp White Spot Syndrome virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	5.733333	294	-11089.9
	STD	3.575046	368.9222	5189.984
	C.I.	0.057883	5.973153	84.0301
	P Value	2.27E-05	0.008046	9.21E-07
Steady State	mean	8.266667	334.9333	-2083.8
	STD	5.430952	383.7001	1973.196
	C.I.	0.087932	6.21242	31.94766
	P Value	3.9E-05	0.004481	0.001103
GA Elit	mean	4.333333	221.0667	-2332.33
	STD	4.237025	305.9399	1875.171
	C.I.	0.068601	4.953418	30.36055
	P Value	0.001421	0.014221	0.000274
GA	mean	8	382.1333	-2094
	STD	4.39155	341.3266	2169.55
	C.I.	0.071103	5.526357	35.1268
	P Value	5.73E-06	0.000684	0.002205
NSGA	mean	5.066667	228	-1403.87
	STD	4.463609	385.9456	1794.138
	C.I.	0.072269	6.248776	29.04857
	P Value	0.000609	0.03821	0.008991

Table 4.56: Comparison of NSGA-(HC) for Shrimp White Spot Syndrome virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	6.266667	167.9333	-1138.81
	STD	4.83243	416.1257	469.4022
	C.I.	0.078241	6.737417	7.600007
	P Value	0.000187	0.14037	2.01E-07
Steady State	mean	4.866667	76.53333	-378.718
	STD	6.232022	379.9975	304.3385
	C.I.	0.100902	6.152472	4.92749
	P Value	0.009099	0.448356	0.000272
GA Elit	mean	4.4	139.5333	-292.604
	STD	5.382511	353.778	353.6516
	C.I.	0.087147	5.727955	5.72591
	P Value	0.006869	0.1489	0.006364
GA	mean	9.2	377.2667	-456.435
	STD	5.759464	404.0455	318.3675
	C.I.	0.09325	6.541828	5.154632
	P Value	2.37E-05	0.002806	7.12E-05
NSGA	mean	7.133333	294.3333	-428.747
	STD	7.689201	478.4174	545.372
	C.I.	0.124494	7.745971	8.830021
	P Value	0.002939	0.031905	0.00874

Table 4.57: Comparison of NSGA-(SAHC) for Shrimp White Spot Syndrome virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	7.533333	284.4	-1215.41
	STD	2.065591	282.707	422.6859
	C.I.	0.033444	4.577259	6.843631
	P Value	1.12E-09	0.001614	2.42E-08
Steady State	mean	6.133333	193	-455.321
	STD	3.313752	191.2123	239.404
	C.I.	0.053652	3.095884	3.876147
	P Value	4.8E-06	0.001573	3.53E-06
GA Elit	mean	5.666667	256	-369.207
	STD	4.820591	301.0078	312.4569
	C.I.	0.078049	4.873564	5.058934
	P Value	0.000451	0.005327	0.000431
GA	mean	10.46667	493.7333	-533.039
	STD	2.503331	298.5839	295.4748
	C.I.	0.040531	4.834319	4.78398
	P Value	1.84E-10	1.64E-05	6.38E-06
NSGA	mean	8.4	410.8	-505.35
	STD	5.461554	382.4947	521.0562
	C.I.	0.088427	6.192902	8.436327
	P Value	3.51E-05	0.000963	0.002127

Table 4.58: Comparison of NSGA-(GA) for Shrimp White Spot Syndrome virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	6.666667	42.46667	-1352.93
	STD	3.221949	433.1265	338.9831
	C.I.	0.052166	7.012673	5.488415
	P Value	1.34E-06	0.709841	3.42E-10
Steady State	mean	5.266667	-48.9333	-592.836
	STD	4.267262	352.7955	148.8373
	C.I.	0.06909	5.712048	2.409798
	P Value	0.000293	0.599571	3.51E-10
GA Elit	mean	4.8	14.06667	-506.722
	STD	5.759464	343.1261	220.4333
	C.I.	0.09325	5.555492	3.568996
	P Value	0.006075	0.876114	3.86E-07
GA	mean	9.6	251.8	-670.554
	STD	3.065942	435.9573	235.5901
	C.I.	0.04964	7.058506	3.814397
	P Value	8.17E-09	0.042075	2.76E-08
NSGA	mean	7.533333	168.8667	-642.865
	STD	6.58859	496.8495	440.4848
	C.I.	0.106675	8.044401	7.13181
	P Value	0.000573	0.209208	5.97E-05

Table 4.59: Comparison of Weighted Sum Methods for Polio Virus Considering Objective Set 1

Algorithm		Δf_1	Δf_2	Δf_3
Single State	mean	-9.46667	-30.9333	-108.733
	STD	5.18055	10.89211	121.1884
	C.I.	0.083877	0.176352	1.96214
	P Value	5.53E-06	2.84E-08	0.003716
Steady State	mean	-2.93333	-9.73333	-37.9333
	STD	4.216747	17.4457	108.559
	C.I.	0.068273	0.28246	1.75766
	P Value	0.017454	0.048528	0.197402
GA Elit	mean	-3.53333	-14	-83.2
	STD	5.629852	18.73881	115.5238
	C.I.	0.091152	0.303397	1.870425
	P Value	0.029103	0.011791	0.014482
GA	mean	-1.73333	-3.73333	-65.9333
	STD	4.350151	16.29402	74.55148
	C.I.	0.070433	0.263814	1.20705
	P Value	0.145079	0.38986	0.004101
NSGA	mean	-4.66667	-5.93333	-68.5333
	STD	8.346656	24.13198	127.5522
	C.I.	0.135139	0.390717	2.065174
	P Value	0.048115	0.357122	0.056281

ble 4.60. In this table, we see that since we set a goal, the value of Δf_2 is better compared to our proposed elitist approaches. Mean values of Δf_1 and Δf_4 reveal that we can increase the number of enzymes and decrease the standard deviation of gaps comparing with the tradition meta-heuristics. But this result can not have good P Values which indicate that we do not get good result for every time.

Table 4.60: Comparison of Goal Programming for Polio Virus Considering Objective Set 2

Algorithm		Δf_1	Δf_2	Δf_4
Single State	mean	-4.06667	-23.6	-10.2889
	STD	7.045431	21.20916	11.39161
	C.I.	0.114071	0.343394	0.18444
	P Value	0.04219	0.00072	0.003549
Steady State	mean	3.333333	-1.2	-11.4791
	STD	6.955642	24.31695	13.63165
	C.I.	0.112618	0.393711	0.220708
	P Value	0.084618	0.851172	0.005682
GA Elit	mean	1.533333	-5.8	-13.6309
	STD	7.11002	17.41182	11.64767
	C.I.	0.115117	0.281912	0.188585
	P Value	0.417609	0.217913	0.000469
GA	mean	1.266667	-6.4	-8.89238
	STD	5.133457	14.7348	10.05629
	C.I.	0.083115	0.238568	0.16282
	P Value	0.355462	0.114688	0.004106
NSGA	mean	0.8	-6.26667	-11.7692
	STD	3.967727	15.68196	12.34534
	C.I.	0.064241	0.253904	0.199881
	P Value	0.447867	0.144002	0.002414

Notably, in most cases, Δf_1 have positive value and Δf_4 (or Δf_3) have negative value. In both cases, P Value is lower which indicates that our approaches give better result against traditional meta-heuristics.

The two objectives, f_1 and f_2 are inversely related to each other. Therefore, as we increase the number of insertions of restriction sites, the performance with respect to f_1 increases whereas the same with respect to f_2 decreases.

Now some discussion on the objective f_2 is in order. We believe, the direct count of changed base pairs might not be that interesting and useful from biological point of view as argued below. This is because the count of changing base pairs is not directly related to the cost. The cost is related to number of times we need to apply a bio-method to change a block of sequence because these changes are not made base by base. So, it's

not logical to say for example that 80 base pairs changes are always better than 100 base pairs changes. Rather, how many attempts were required to change those 100 base pairs seem more important. For example, if it takes 12 attempts to change 80 base pairs and 11 attempts to change 100 base pairs, the latter would be more desirable. For this we give more concentration on f_1 and f_3 (or f_4) rather than f_2. Finally we note that it would be interesting to formulate a new objective where we could incorporate the idea of 'attempts made' rather than the actual change of base pairs. At the current setting, we could not use/formulate this new objective, because we do not know how an attempt can be used to change a number of base pairs. Moreover, we need to know the base pairs that can be changed by an individual attempt. This would require some more information from the biologist which must be provided to the algorithm as input.

From Tables 4.5 to 4.58, our approaches always give positive and negative mean values for Δf_1 and Δf_4 (or Δf_3) respectively, most of the P Values of those Tables are less than 0.01 for Δf_4 (or Δf_3) and Δf_1 respectively. These outcome imply that our results outperform previous results of [19, 20] with respect to f_1 and f_3 (or f_4). Comparing among 3 approaches of this thesis, none can give significant difference for Δf_1. But for Δf_4 (or Δf_3), generating population sets with genetic algorithm gives better results. For example, for Shrimp White Spot Syndrome Virus in Tables 4.53 through 4.58, the mean differences are always better for the approach which uses genetic algorithm than the other two approaches. Comparing with GA in Tables 4.56 and 4.57, Δf_4 is -456.435 and -533.039 respectively where as Δf_4 in Table 4.58 is -670.554 which is significantly better. We can identify similar results from Table 4.5 to 4.58. The results of our proposed algorithms considering viral sequences are shown in Table 4.61 to 4.69.

From Figure 4.2 to 4.13, graphical representations of the comparison among the traditional meta-heuristics and our proposed algorithms for HIV virus and Shrimp White Spot Syndrome Virus are shown. Here mean of every objective values for each algorithms are shown by charts. Recall that higher number for f_1 is better where as for f_2 and f_3 lower number is better.

Table 4.61: Results of proposed Algorithms for Polio Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
84	153	256	97	176	262	85	141	274	90	166	62.0	87	161	58.1	73	110	67.1
88	166	249	92	178	258	91	164	261	92	185	55.7	84	163	55.7	80	146	58.3
82	149	275	93	186	241	75	123	296	88	166	62.8	86	157	60.2	78	115	74.5
76	142	297	92	174	274	92	174	263	89	165	67.0	81	136	65.8	85	161	58.9
86	172	277	88	163	284	93	179	260	84	174	59.2	82	162	55.5	79	151	58.6
91	176	298	92	174	281	89	176	251	82	150	68.4	95	174	65.1	75	129	74.1
91	179	302	94	185	261	95	196	242	70	130	67.0	90	186	56.2	76	142	70.0
92	184	299	99	198	255	96	192	260	84	173	61.3	83	168	58.2	83	169	63.6
93	192	294	82	158	282	81	148	306	88	169	69.1	87	171	64.2	69	125	72.8
93	185	310	83	163	277	86	176	278	87	175	65.6	77	124	79.7	89	176	67.0
90	189	297	96	197	252	71	134	308	83	170	64.6	86	175	65.4	78	151	69.2
94	194	315	94	188	269	89	155	346	93	183	67.9	80	159	68.3	83	160	71.1
85	168	336	90	182	283	82	162	311	67	98	83.2	78	165	65.5	77	171	64.1
82	162	336	95	192	322	92	195	311	92	189	67.3	76	132	79.2	78	151	77.0
93	171	376	78	145	371	69	128	394	81	163	69.8	72	137	73.8	79	156	76.5

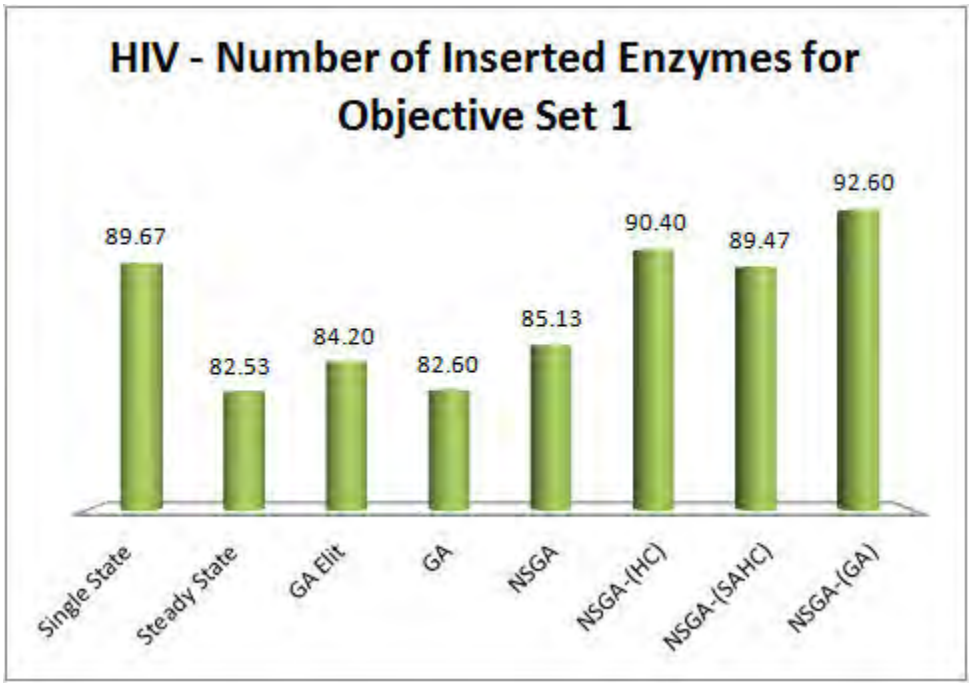

Figure 4.2: Comparison Considering Objective Set 1 for HIV Virus in terms of f_1 (higher is preferred)

Table 4.62: Results of proposed Algorithms for Rubella Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
106	775	354	94	836	315	102	877	302	91	790	64.2	92	740	64.6	90	810	64.8
110	984	320	112	1029	311	113	1050	283	94	901	64.5	90	906	57.9	91	806	69.8
111	1045	306	106	965	323	104	943	298	101	982	65.1	104	988	64.2	107	1045	64.8
102	995	290	105	896	349	101	910	328	103	944	70.2	95	782	74.3	100	978	66.4
101	881	340	107	1065	304	102	990	298	96	853	71.8	96	973	61.6	82	802	70.4
106	941	336	97	1030	304	100	1012	293	101	852	76.5	89	753	74.5	101	925	77.2
104	1057	319	102	1048	332	108	893	403	104	1041	68.8	97	973	65.5	96	849	79.8
96	952	345	98	977	355	106	1061	307	92	838	76.3	95	897	71.3	96	952	71.5
100	956	363	108	1060	369	102	1030	305	92	826	77.3	90	835	72.3	91	847	82.9
85	803	396	102	1018	366	87	830	368	105	1034	73.6	95	924	69.2	83	754	85.8
107	1129	367	98	975	371	96	894	387	101	989	74.7	90	806	78.8	85	837	80.1
103	983	425	82	928	326	85	926	325	99	917	79.2	100	969	74.1	87	964	71.9
87	1021	356	78	844	347	96	1045	318	92	939	74.6	96	1032	66.3	87	944	74.2
82	971	361	103	1049	370	87	925	338	92	974	71.8	91	956	71.0	89	1003	72.7
103	1011	449	103	1051	371	84	778	402	102	1050	75.2	96	1009	71.1	86	856	84.2

Table 4.63: Results of proposed Algorithms for Papilloma Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
92	140	266	88	138	273	92	134	290	83	111	66.5	82	125	67.4	88	123	68.4
94	145	297	74	111	290	76	101	317	91	145	62.8	86	122	72.8	79	110	67.6
75	98	369	91	136	316	85	123	315	84	118	74.8	89	150	62.2	89	132	65.9
96	164	290	93	158	277	87	125	330	81	106	80.0	84	125	71.9	89	144	59.2
77	118	329	95	138	342	89	144	284	74	93	83.6	90	163	57.9	85	139	60.0
86	148	291	93	162	284	89	140	302	92	153	64.2	91	150	66.7	88	144	60.7
95	162	296	76	117	327	75	112	318	90	147	66.6	86	144	66.3	90	136	67.7
90	148	315	89	150	306	88	137	309	91	149	67.5	91	158	63.6	85	129	68.6
93	157	309	85	138	323	92	154	290	76	107	79.9	93	161	65.2	90	151	62.0
94	151	335	92	171	270	87	145	293	94	150	72.1	88	145	70.3	87	138	67.1
86	133	348	89	154	304	90	136	339	90	144	71.7	79	128	72.7	90	144	68.4
88	135	355	85	142	318	93	133	395	89	149	67.5	87	138	76.2	84	138	69.0
90	147	335	86	137	337	79	104	418	83	135	73.1	90	148	74.9	80	138	66.1
90	159	321	93	149	342	87	134	385	91	157	69.0	76	123	75.5	82	113	84.0
83	143	336	90	137	369	88	124	430	82	136	74.5	78	119	80.1	82	137	70.8

Table 4.64: Results of proposed Algorithms for Mosaic Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
107	206	180	102	180	206	94	170	198	90	174	43.6	95	162	44.0	96	195	41.8
101	193	192	87	168	207	91	166	226	90	161	49.1	84	137	52.2	95	183	48.7
103	193	219	96	191	212	95	181	216	99	194	44.3	92	169	48.0	92	184	47.2
104	189	248	89	178	210	98	201	197	96	207	42.8	92	179	44.9	98	197	48.0
100	198	233	100	205	210	101	200	210	94	185	49.1	90	179	45.5	93	190	48.1
101	207	230	101	204	227	97	198	200	97	191	49.4	96	185	50.0	95	190	51.4
95	197	231	108	225	218	103	197	226	92	173	52.7	95	179	51.9	86	174	50.8
96	206	220	94	185	239	97	197	209	95	196	47.1	90	188	47.3	90	183	50.7
98	198	248	98	189	247	80	156	226	90	178	50.1	93	171	55.7	80	148	56.4
99	194	259	104	218	220	99	195	224	96	192	51.6	90	177	52.2	81	154	55.3
82	158	264	90	182	241	95	186	232	102	199	53.6	81	144	59.5	93	184	55.9
87	168	272	85	163	258	95	189	229	79	155	53.0	95	213	48.3	98	202	55.2
103	218	245	103	199	265	100	194	254	88	174	54.7	93	171	61.0	91	185	56.9
90	190	253	103	207	259	98	205	232	88	171	55.8	93	198	53.6	74	127	66.8
88	157	305	96	200	249	100	205	246	93	196	50.9	76	147	59.9	88	176	60.1

Table 4.65: Results of proposed Algorithms for HIV Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
98	143	283	96	134	326	91	124	315	95	129	77.8	88	113	81.8	86	109	77.2
91	125	314	77	93	427	93	128	342	91	128	80.7	91	121	89.3	90	122	75.4
91	134	328	93	139	355	95	135	329	98	142	80.4	89	126	83.6	89	114	86.4
92	128	374	94	132	409	92	137	313	95	134	85.2	89	127	86.5	95	133	80.0
98	146	343	80	102	452	92	118	408	92	134	84.2	93	132	89.9	91	132	77.4
82	112	410	97	141	407	95	120	430	88	124	87.6	86	124	87.9	89	131	76.7
96	138	411	95	150	357	88	131	341	96	143	83.6	92	134	87.7	89	132	78.4
92	130	425	80	106	467	95	139	357	89	124	91.2	92	137	85.8	93	127	91.5
90	136	400	88	124	452	94	132	412	86	129	85.1	86	128	85.9	89	130	84.6
78	110	430	93	142	407	94	139	385	98	154	82.8	84	115	93.2	89	136	80.5
99	145	419	90	126	470	97	151	356	92	134	91.1	81	94	107.4	87	129	83.8
85	128	410	93	142	421	97	142	409	88	122	95.5	91	139	87.2	89	131	87.6
93	120	490	94	141	440	89	127	421	88	124	94.8	90	126	95.3	91	139	87.1
79	106	474	96	139	471	85	125	410	88	129	91.2	85	123	92.3	90	116	105.3
92	134	451	76	97	533	92	140	404	97	156	82.0	94	138	93.8	75	87	113.0

Table 4.66: Results of proposed Algorithms for Hepatitis C Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
92	161	414	92	165	422	91	166	364	90	163	88.3	91	146	104.7	96	175	92.2
85	145	432	90	156	448	82	136	442	97	186	86.0	97	178	92.9	89	160	97.5
89	149	487	91	178	390	82	140	438	87	150	99.8	86	151	101.4	94	159	106.0
92	170	415	96	197	367	93	174	386	84	142	103.2	92	170	97.2	96	165	104.8
96	180	407	94	170	443	85	157	391	96	176	94.8	97	184	95.7	93	171	97.0
93	172	424	93	167	460	95	165	448	94	181	93.8	85	152	102.6	93	161	104.6
94	171	443	90	179	409	77	138	410	96	177	106.0	93	165	104.7	94	185	89.4
92	166	466	81	155	443	90	168	406	91	170	104.5	94	180	96.8	92	179	92.1
90	168	448	89	152	512	90	162	438	83	144	113.3	89	175	94.3	91	160	104.9
75	135	474	84	150	485	83	157	408	94	174	106.8	89	156	107.8	90	183	89.7
79	139	494	88	166	467	93	189	352	96	188	99.6	93	170	104.2	90	175	101.6
95	180	443	94	187	457	84	150	456	94	185	99.2	90	169	101.2	92	177	103.2
96	182	467	93	162	534	94	173	442	79	135	117.4	86	150	112.4	91	181	99.1
89	163	510	96	183	486	73	130	464	93	175	112.2	95	175	109.6	93	181	102.6
94	167	569	91	173	497	94	168	497	84	159	112.9	89	159	113.1	83	144	116.3

Table 4.67: Results of proposed Algorithms for Bronchitis Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
52	40	2324	38	19	2324	41	23	2324	48	41	467.4	48	40	483.9	53	45	440.5
52	42	2324	53	45	2324	51	38	2324	48	40	480.1	49	48	442.7	48	38	491.8
51	41	2324	51	43	2324	51	40	2324	51	45	471.3	51	45	486.0	48	47	435.3
53	45	2324	50	43	2324	51	42	2324	54	52	441.3	47	43	472.8	46	35	523.5
52	47	2324	51	45	2324	52	44	2324	52	46	486.5	50	44	496.8	47	41	486.0
51	46	2324	53	48	2324	51	43	2324	52	51	448.1	49	42	522.0	48	44	473.0
54	51	2324	52	48	2324	51	43	2324	48	45	457.1	51	52	467.6	52	52	443.5
52	49	2324	52	48	2324	45	37	2324	49	43	490.1	50	55	448.5	46	40	490.3
52	50	2324	53	50	2324	52	45	2324	52	53	455.1	53	55	481.1	47	47	450.3
52	50	2324	53	50	2324	51	44	2324	47	44	490.1	55	62	447.9	46	39	509.8
55	54	2324	51	49	2324	50	43	2324	51	50	484.0	52	56	467.0	54	54	467.1
52	52	2324	50	49	2324	49	43	2324	52	53	469.6	50	54	463.1	53	58	424.6
52	53	2324	54	54	2324	53	51	2324	53	56	461.4	54	60	456.6	52	57	436.2
54	56	2324	53	53	2324	52	50	2324	47	47	481.1	53	59	456.2	54	59	459.0
53	57	2324	53	54	2324	53	53	2324	56	56	509.7	53	58	464.7	52	55	476.4

Table 4.68: Results of proposed Algorithms for Glossina Pallidipes Salivary Gland Hypertrophy Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
97	1100	7993	111	1238	8105	107	1348	7078	115	1357	1834.2	112	1436	1660.5	110	1344	1773.3
116	1392	7974	107	1270	7947	107	1295	7500	113	1462	1700.7	111	1348	1775.8	109	1446	1639.1
114	1492	7645	110	1308	8517	100	1315	7367	109	1172	2065.3	111	1381	1794.5	112	1515	1644.6
114	1389	8895	110	1340	8298	106	1313	7993	112	1426	1817.7	113	1403	1846.2	112	1386	1844.4
116	1559	7690	112	1416	8496	116	1534	7655	108	1333	1873.1	109	1348	1919.2	112	1496	1721.6
111	1361	9060	114	1511	8089	110	1428	7846	120	1573	1792.8	108	1432	1815.4	113	1421	1884.8
106	1379	8276	113	1448	8594	111	1510	7491	112	1369	1960.0	105	1342	1886.6	104	1336	1840.4
111	1412	8697	115	1506	8642	107	1478	7370	106	1297	1980.3	106	1462	1796.1	104	1308	1877.4
115	1553	7940	114	1489	8688	111	1386	8298	115	1451	1953.6	116	1491	1954.4	112	1491	1808.3
116	1537	8496	111	1566	7929	109	1453	8079	113	1457	1979.4	109	1416	1948.7	117	1565	1812.8
97	1140	9715	106	1475	8198	92	1191	8258	111	1517	1891.9	112	1532	1854.3	108	1356	1921.1
112	1580	7795	108	1466	8918	106	1435	8010	108	1436	1989.0	110	1505	1870.9	115	1568	1792.3
112	1515	8615	109	1523	8715	112	1504	8108	115	1511	2016.9	112	1521	1934.0	115	1601	1789.5
110	1349	9885	109	1496	9191	102	1282	8668	100	1204	2179.0	106	1340	2087.5	110	1489	1862.9
101	1336	9009	113	1562	9143	104	1341	8535	114	1485	2084.1	113	1423	2124.6	110	1532	1812.6

Table 4.69: Results of proposed Algorithms for Shrimp White Spot Syndrome Virus

Objective Set 1									Objective Set 2								
NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)			NSGA-(HC)			NSGA-(SAHC)			NSGA-(GA)		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_4	f_1	f_2	f_4	f_1	f_2	f_4
111	4161	12852	107	4460	11626	108	4362	10198	109	4695	2402.5	108	4508	2566.3	107	3753	2632.9
108	4347	11555	109	4418	12137	105	3860	11392	108	4299	2720.8	107	4332	2684.2	107	3745	2775.2
108	4284	12752	107	4446	11760	103	4179	10830	112	4610	2750.1	105	4382	2617.0	103	4071	2559.9
111	4281	13501	108	4127	13225	109	4436	10889	103	3853	3060.7	109	4351	2759.9	108	3979	2765.0
109	4441	12460	105	4392	11929	103	4026	11511	107	4351	2798.4	108	4400	2714.9	110	4417	2576.8
103	4216	12395	109	4578	11982	106	4186	12089	107	4271	2970.5	107	4156	2916.1	107	4172	2651.5
103	4544	11002	110	4646	12087	103	4311	11386	109	4569	2804.8	111	4601	2831.2	106	3873	2833.0
110	4646	11949	110	4508	12619	107	4501	11485	101	4001	2997.5	107	4544	2765.3	108	4471	2562.6
106	4472	12129	110	4465	13220	108	4576	11503	105	4408	2824.2	105	4244	2926.9	100	3911	2695.3
105	4478	12129	102	4494	12061	106	4546	11627	110	4510	2940.2	108	4470	2895.9	111	4186	2863.3
105	4259	13295	109	4631	13061	105	4406	12178	109	4477	2940.0	109	4503	2906.5	108	4558	2584.2
97	3653	14640	111	4600	14057	109	4362	13036	106	4272	3007.2	112	4646	2896.0	107	4400	2667.8
110	4420	13732	105	4291	14384	105	4167	13717	95	3482	3276.5	108	4564	2848.3	107	4505	2620.9
104	4340	12917	111	4657	14243	105	4596	12846	112	4800	2797.5	107	4519	2885.0	107	4486	2666.4
105	4581	12772	110	4543	15290	103	4264	13901	106	4432	3027.1	107	4557	2955.4	109	4621	2651.5

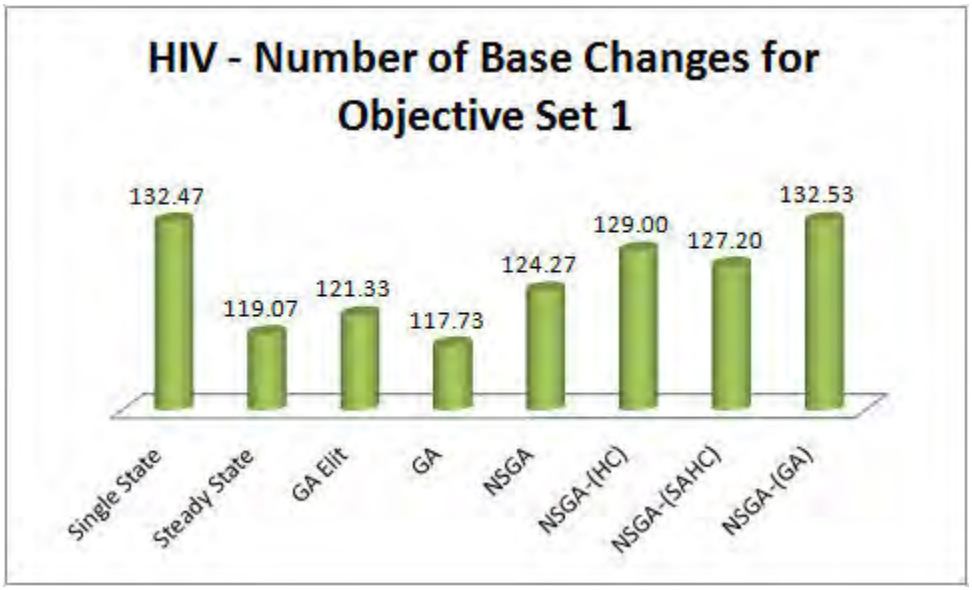

Figure 4.3: Comparison Considering Objective Set 1 for HIV Virus in terms of f_2 (lower is preferred)

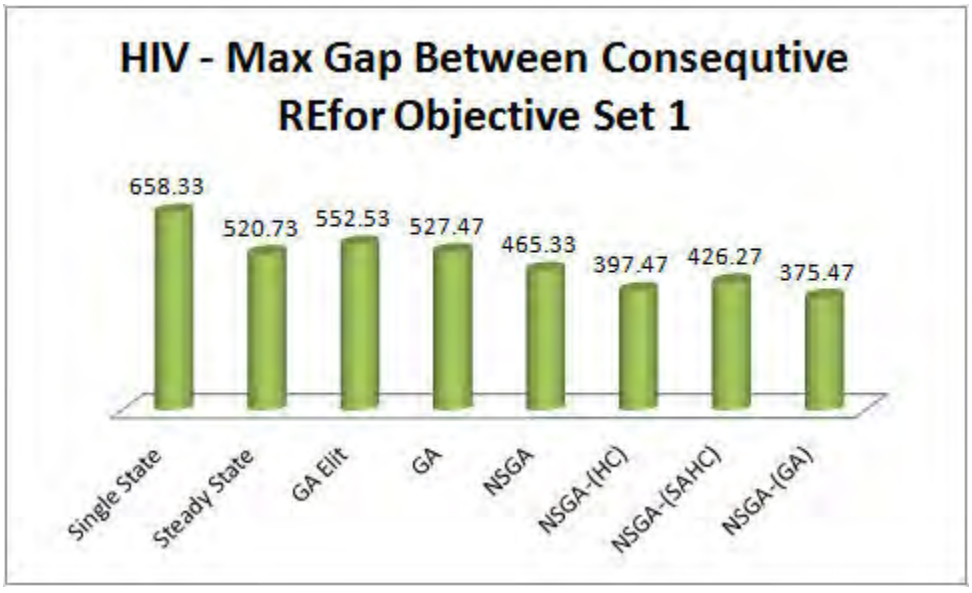

Figure 4.4: Comparison Considering Objective Set 1 for HIV Virus in terms of f_3 (lower is preferred)

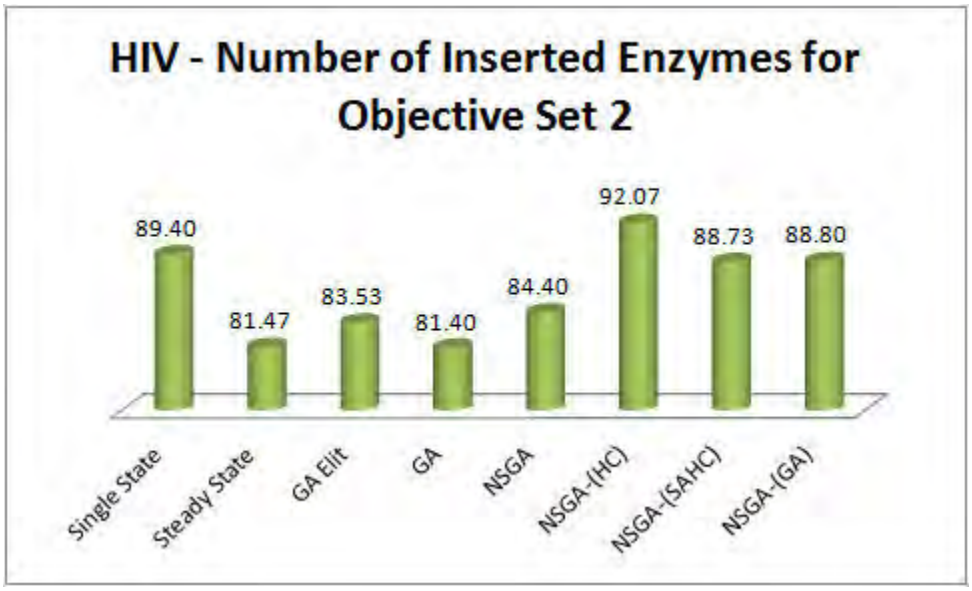

Figure 4.5: Comparison Considering Objective Set 2 for HIV Virus in terms of f_1 (higher is preferred)

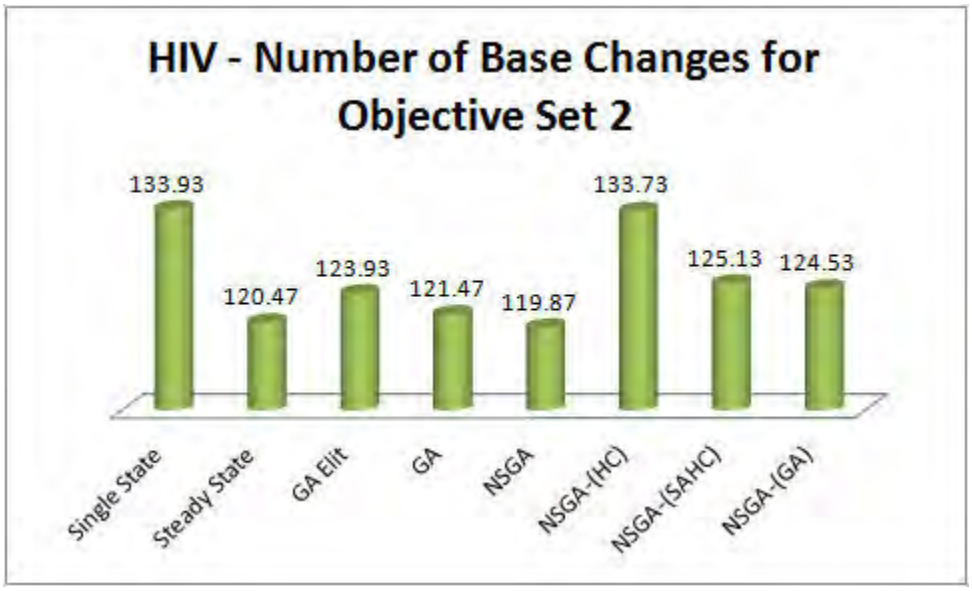

Figure 4.6: Comparison Considering Objective Set 2 for HIV Virus in terms of f_2 (lower is preferred)

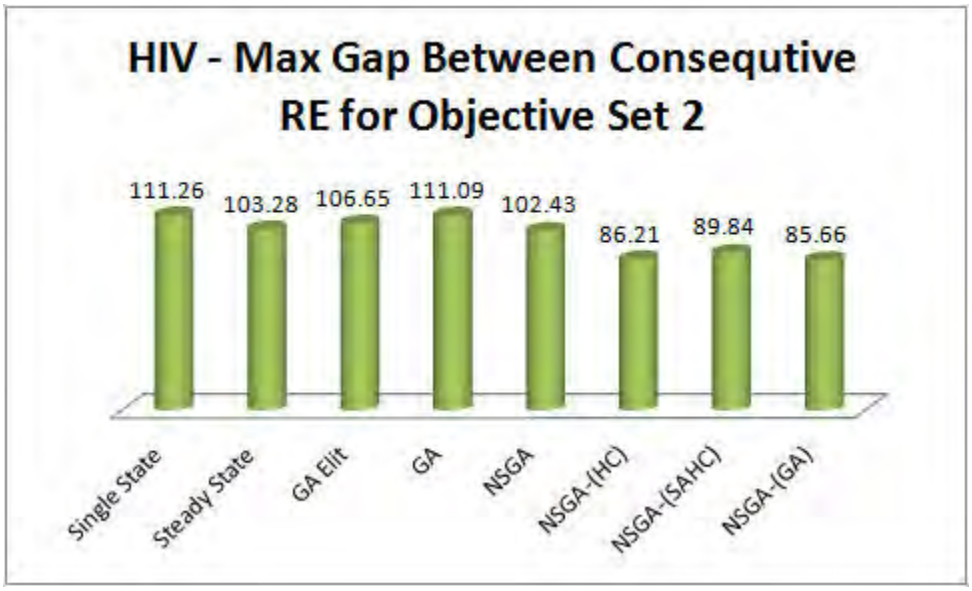

Figure 4.7: Comparison Considering Objective Set 2 for HIV Virus in terms of f_4 (lower is preferred)

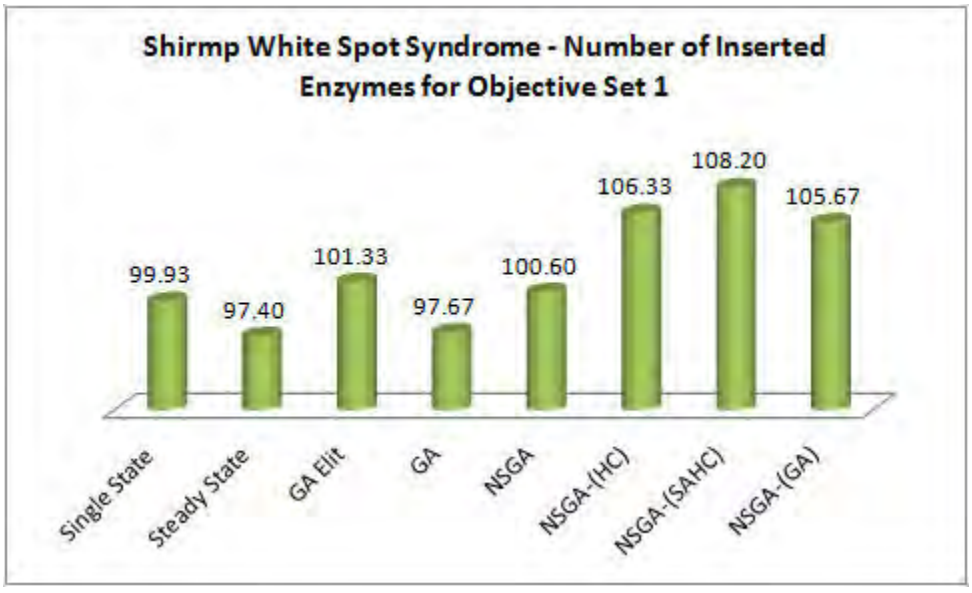

Figure 4.8: Comparison Considering Objective Set 1 for Shrimp White Spot Syndrome Virus in terms of f_1 (higher is preferred)

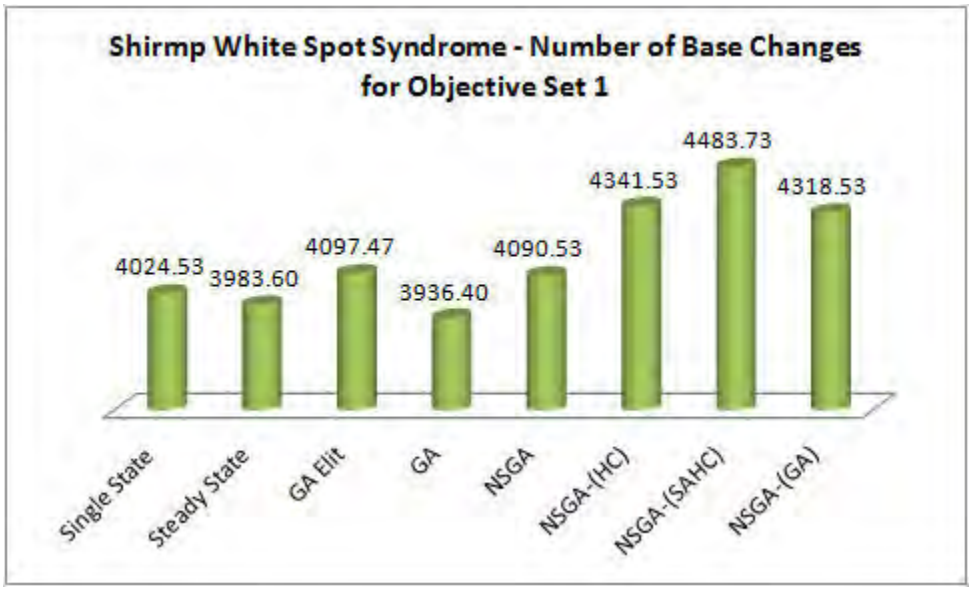

Figure 4.9: Comparison Considering Objective Set 1 for Shrimp White Spot Syndrome Virus in terms of f_2 (lower is preferred)

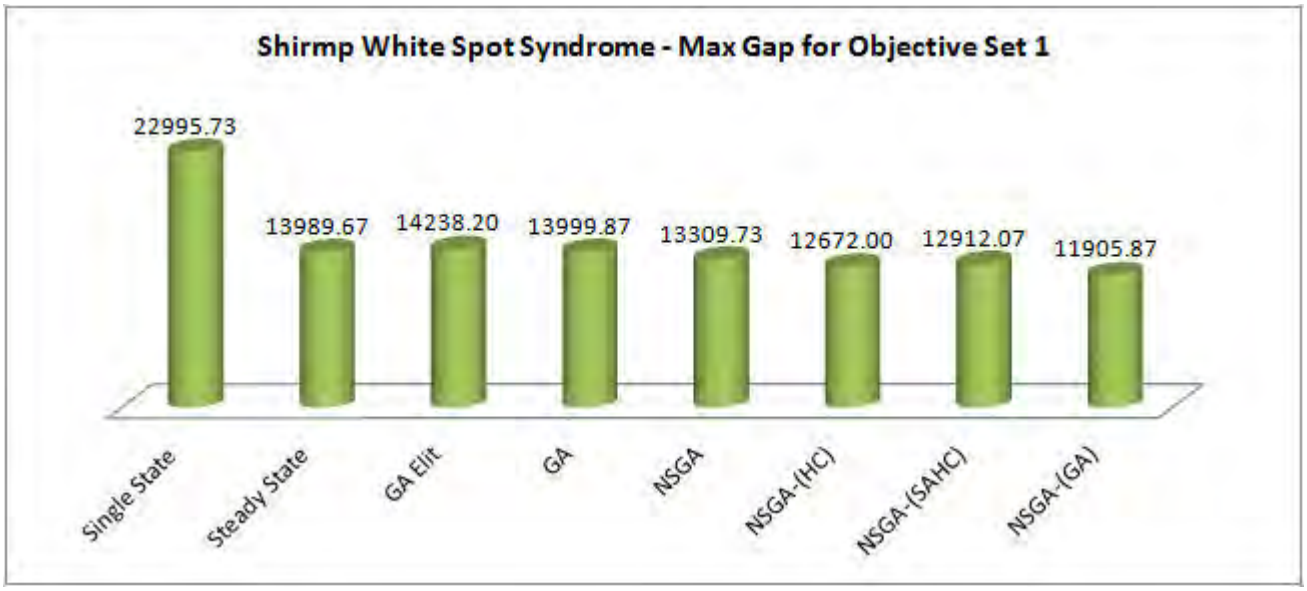

Figure 4.10: Comparison Considering Objective Set 1 for Shrimp White Spot Syndrome Virus in terms of f_3 (lower is preferred)

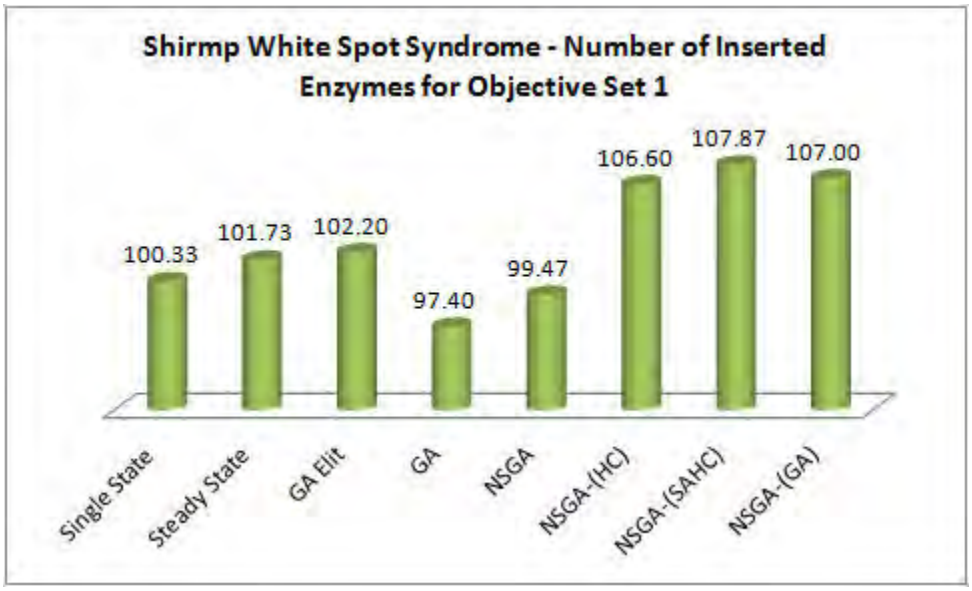

Figure 4.11: Comparison Considering Objective Set 2 for Shrimp White Spot Syndrome Virus in terms of f_1 (higher is preferred)

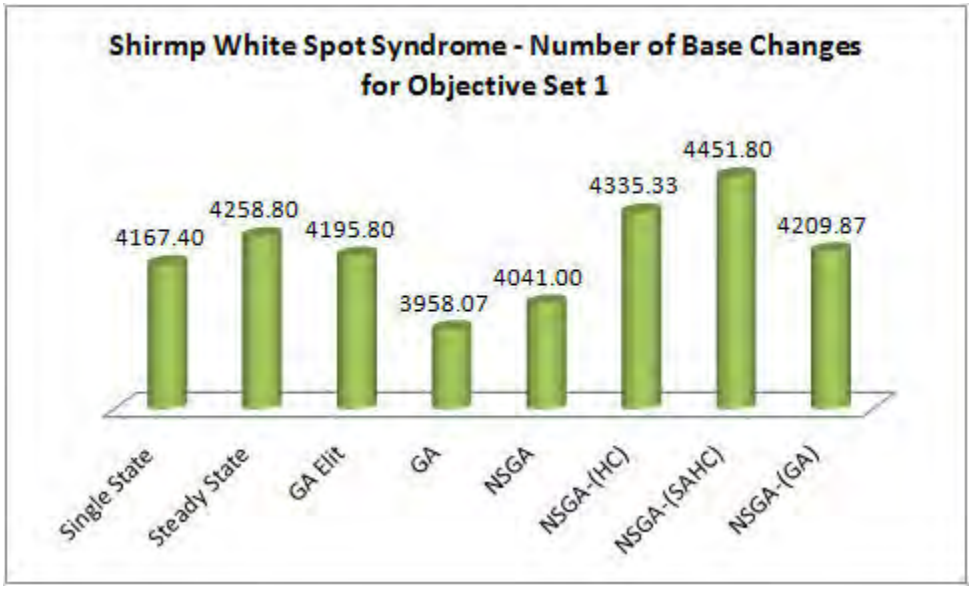

Figure 4.12: Comparison Considering Objective Set 2 for Shrimp White Spot Syndrome Virus in terms of f_2 (lower is preferred)

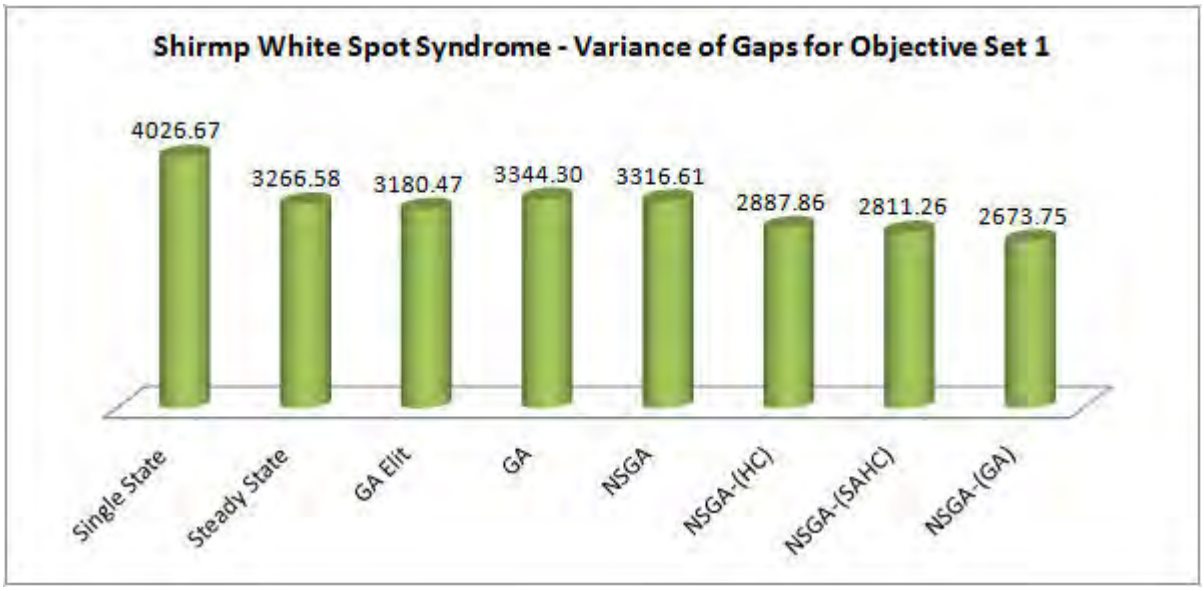

Figure 4.13: Comparison Considering Objective Set 2 for Shrimp White Spot Syndrome Virus in terms of f_4 (lower is preferred)

4.3 Summary

This chapter mainly deals with our experimental techniques and the output of simulation results. We have done our experiments with some clinically important and large viral sequences. We have also given chart foe some viral sequences to show a clear concept of our results. We have also given experimental results for other multi-objective optimization techniques that we have used. These are not only shown in tabular format but also a discussion have been done revealing the reasons behind their behavior. our experimental results have given better results than traditional meta-heuristics to solve URSPP.

CHAPTER 5
Software

We have developed a software tool combining the meta-heuristic approaches to solve URSPP. So our tool includes the traditional meta-heuristic techniques proposed in [19, 20]. We belief, this tool will help the biologists to insert restriction enzymes of any genome sequence and can help the biologists in the field of gene synthesis as well as synthetic biology.

5.1 Programming Environment

We have developed the tools by Java programming Language. We have used Java SDK 4.2 to compile our codes. To write the code, we have chosen Java Eclipse environment. For GUI interface, we have used Jigloo plugin. Jigloo creates and manages code for all the parts of Swing or SWT GUIs as well as code to handle events, and shows the GUIs as they are being built.

5.2 Prerequisite Files

Basically, our tool needs 4 files to complete a successful run with a genome sequence. The restriction site file must be located where the root jar stays. The other 3 files are related to the specific genome sequence. These 3 files must be located in one folder.

The restriction enzymes file name is 'RE.txt' and a snapshot of this file is given in figure 5.1. We can change the set of restriction enzymes by modify the file. The changed file must be in the same format so that our tool can understand the file. The 3 files related to specific genome sequence are as follows:

1) Sequence file named as 'sequence.txt' and a snapshot of the file is shown in Figure 5.2 .

2) Locked Region file named as 'locked.txt' and a snapshot of the file is shown in Figure 5.3.

3) The Genes file named as 'gene.txt' and a snapshot of the file is shown in Figure 5.4.

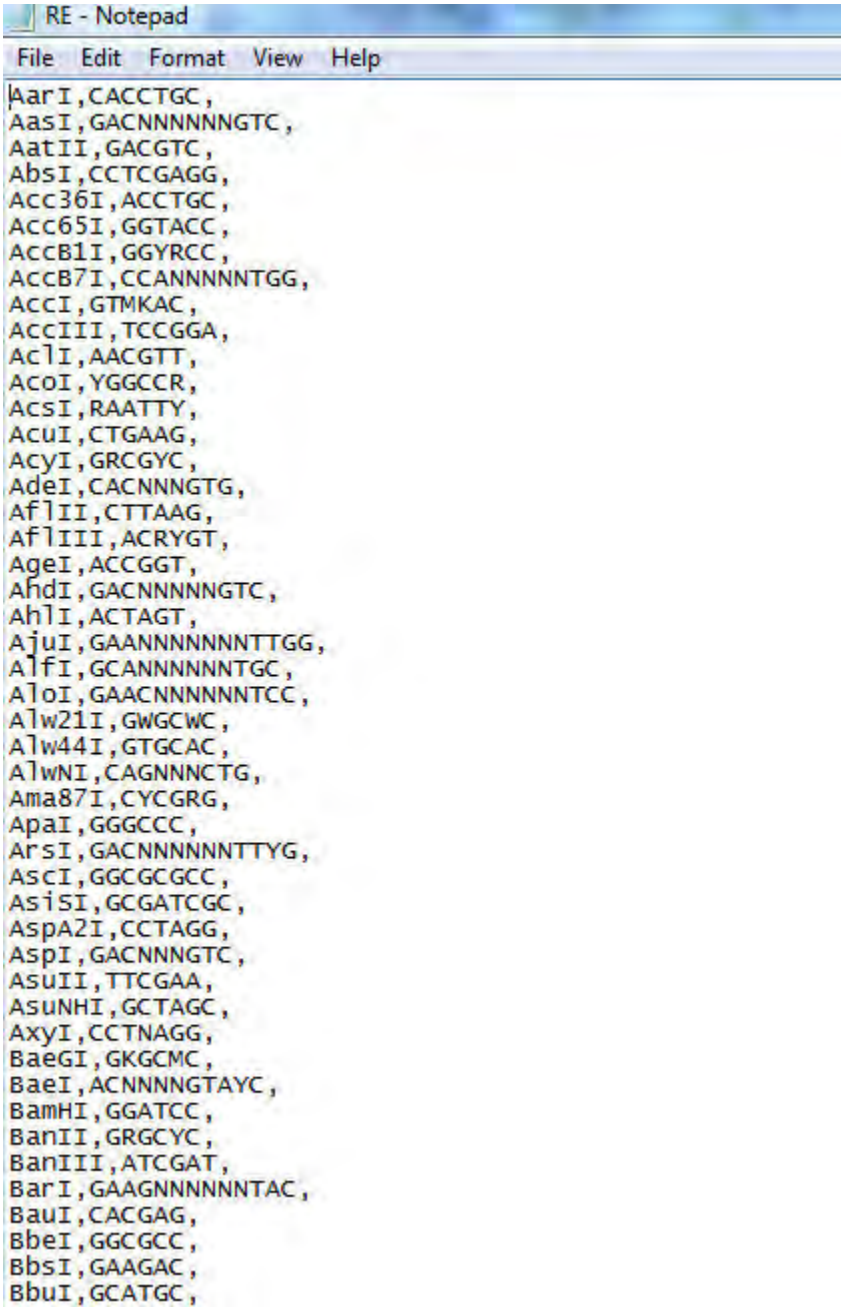

Figure 5.1: RE.txt file.

Figure 5.2: sequence.txt file.

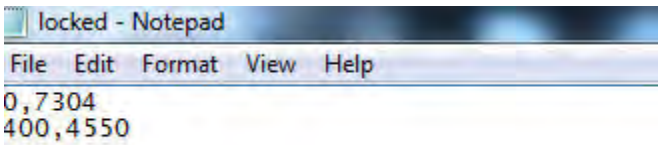

Figure 5.3: locked.txt file.

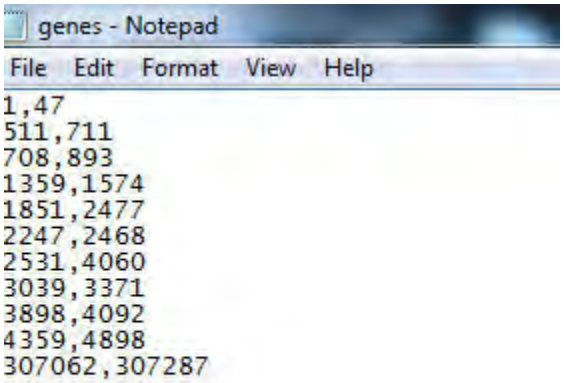

Figure 5.4: genes.txt file.

5.3 Description of the Software

The proposed tool is developed in java. Figure 5.5 shows the outlook of the software.

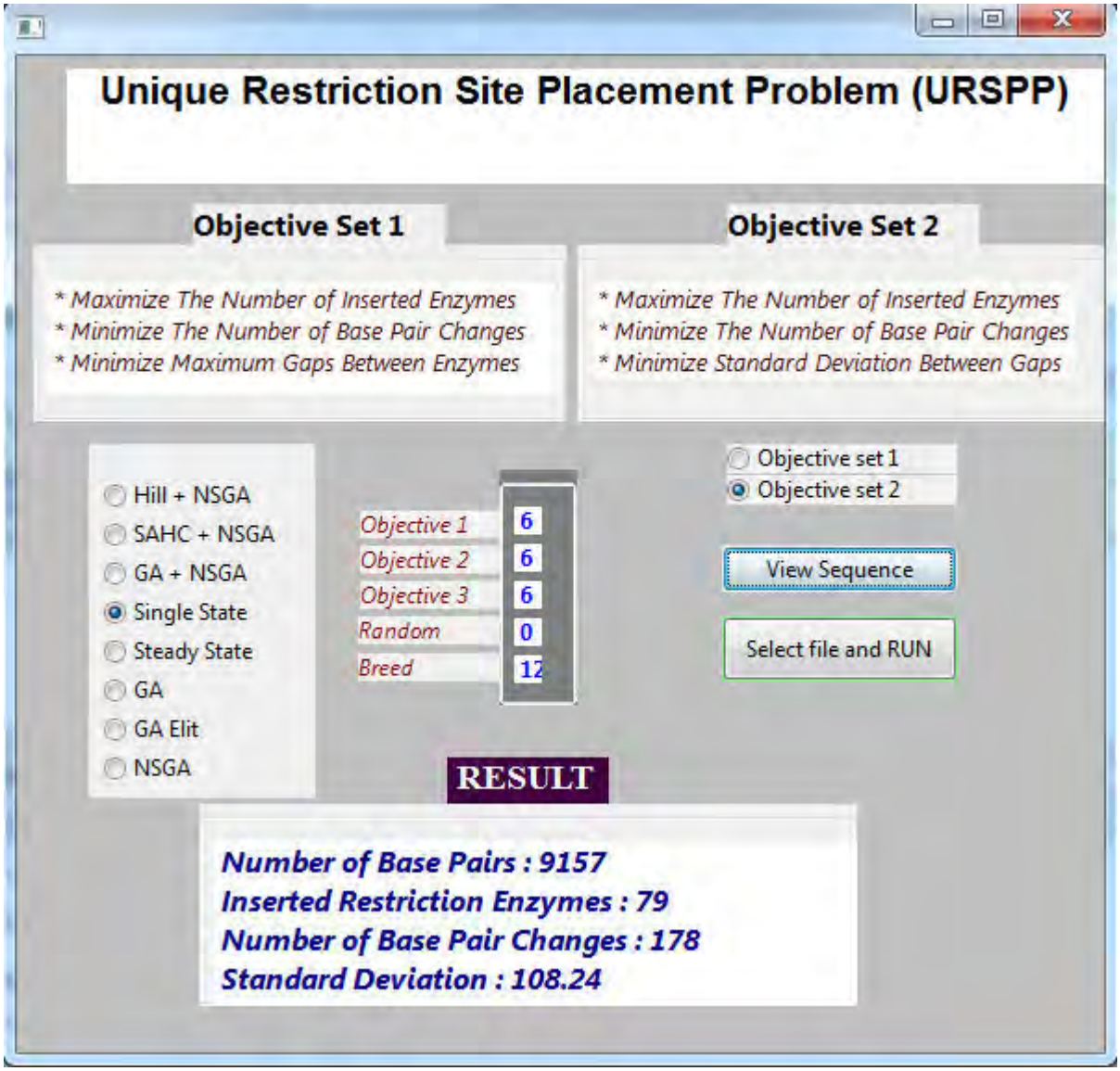

Figure 5.5: Software Tool.

In our proposed tool, we put options of running with any of our proposed approaches as well as the approaches proposed in [19, 20]. The approaches which can be run by our tool are given in Table 5.1. The user of this tool can select any of the approaches and run the algorithms. The zoom picture of algorithm selection is shown in figure 5.6.

Table 5.1: Algorithms incorporate in software tool

Algorithms
Hill Climbing considering single objective and then NSGA
Steepest Ascent considering single objective and then NSGA
Genetic Algorithm considering single objective and then NSGA
Basic Single State Method
Basic Steady State Method
Basic Genetic Algorithm
Genetic Algorithm with Elitism
NSGA

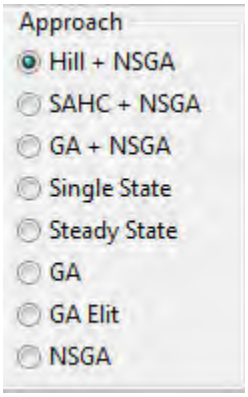

Figure 5.6: Approaches in Software

Figure 5.7: Tuning Parameters.

As our proposed approaches initially run with a single objective, the user can tune the parameters. For example, the user may wants to give more populations set for objective 1 than objective 2. Or he wants to add some more randomness to generate initial population sets. This tool can provide this option to select these tuning parameters. User can also select the number of breeding times for NSGA. The zoom picture to provide this option is shown in figure 5.7. This tuning is availabe in our approaches, selecting/changing

tuning parameter does not provide any extra effect for the works of [19, 20].

Recall that, we have 2 sets of objectives. In our tool, user can choose his desirable objective set. The zoom picture to provide this option is shown in figure 5.8.

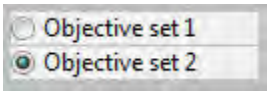

Figure 5.8: Objective Sets.

After selecting the desirable approach and tuning the parameter as well as objective sets, user have to select the genome sequence file by double clicking a button named as "Select file and RUN". After clicking the button, a java file chooser is open and user has to select the 'sequence.txt' file for the specific genome sequence. Recall that, the 'locked.txt' and 'gene.txt' must also be located in the same folder where 'sequence.txt' exists.

Figure 5.9: Result

After selecting the file, our tool runs with the selected approach. After finishing the run, our tool shows the output generated by the selected algorithm. Figure 5.9 shows, how a result is shown in our tool. After Finishing a successful run, our tool generates a output file with the result/outcome of the last successful run. The output file contains the reconstructed sequence and the objective wise output. Figure 5.10 partially shows the output.txt file. Note that, output file will be generated in the same folder where the sequence/genes/locked files exist.

User can also view the new sequence constructed by our tool. For this, he has to click a button named as "View Sequence" and a new window will open. Figure 5.11 shows this new form which compares the original and constructed sequence. The sequence is colored by green where as the changed base pair is colored by black. A slider is also given in the form to see the whole sequence.

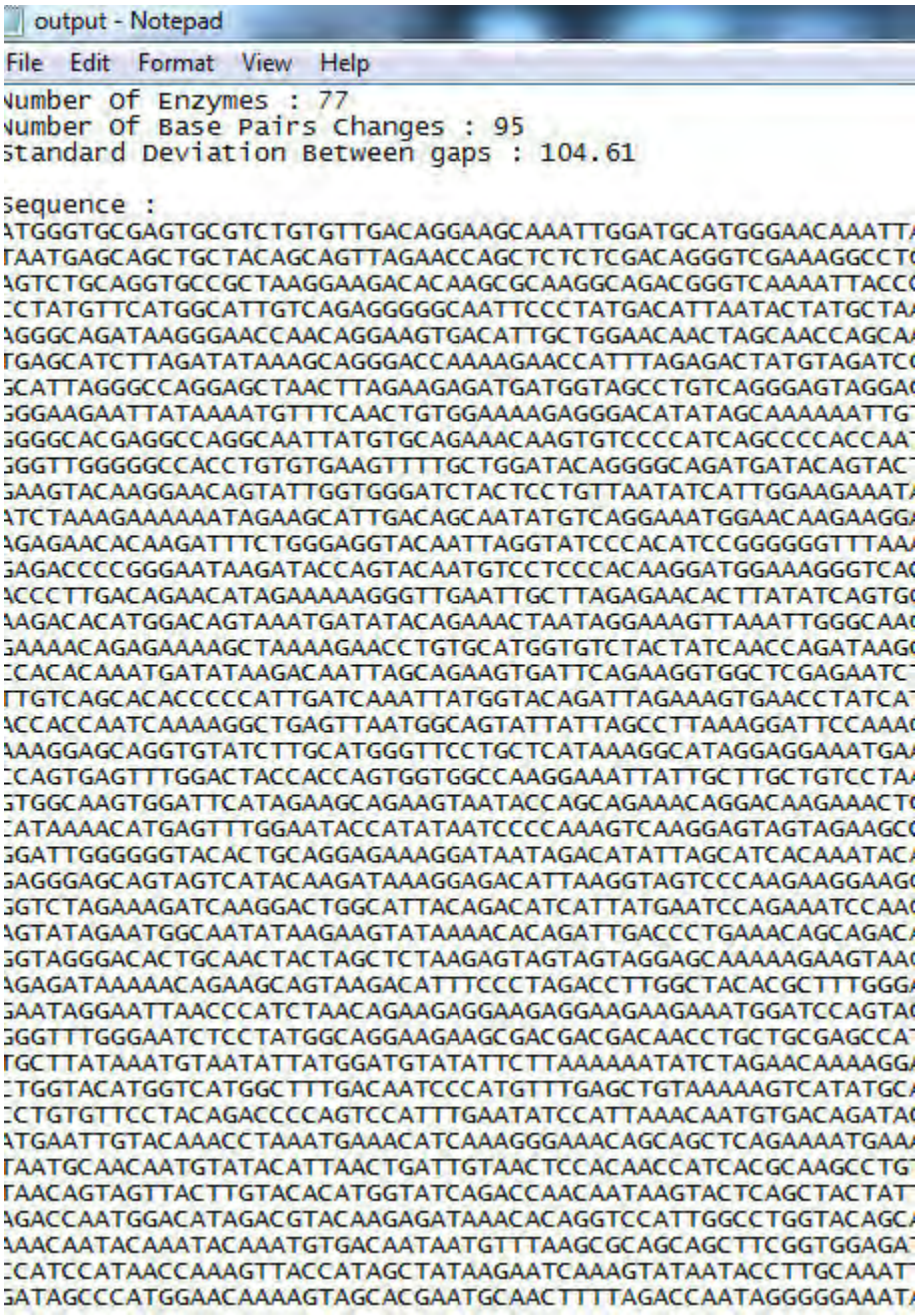

Figure 5.10: output.txt file.

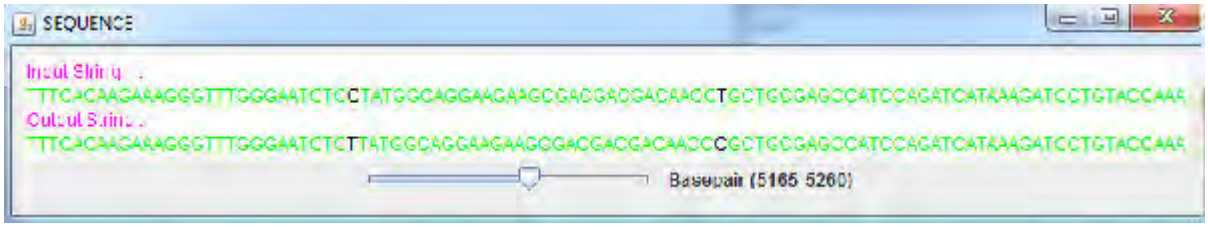

Figure 5.11: View Sequence

5.4 Summary

In this chapter, we presented an overview of our developed software tool. The interface of the software is given here and a short description of how to use the software and the required files to run for a viral sequence. Biologists can easily use this software by providing any viral sequences. Biologists can also see the base pair changes by a graphical view to clearly understand the changes. We belief that this tool will guide the biologists to reconstruct a viral genome and thus it can lead a new dimension for vaccine development and gene therapy.

CHAPTER 6
Conclusion

This thesis deals with a topic which falls under a subfield of bioinformatics, namely synthetic biology. The main idea of this study is to reconstruct a genome sequence which aids the biologists to obtain maximum amount of flexibility and independence to conduct experiments with the sequence in question. One of the goal of this thesis is to introduce a new objective of URSPP which is more meaningful and interesting for biologists. Another aim is to give opportunity for investigation of vaccine invention. To this end, for making a genome sequence uniquely prone to enzymes, we proposed to apply elitist meta-heuristics process. We have done extensive experiments with some large viral sequences to test our works. We have also provide an open source software tool for gene synthesis. We belief, this tool can add values in the field of vaccine development, gene therapy and recombinant DNA technology.

Our thesis starts with an overview of bioinformatics, synthetic biology and basic meta-heuristic techniques. The relationship of synthetic biology with computer science is also discussed for clear perception to even novice reader. As in bioinformatics, we have to face the challenge of converting a biological problem to a easily manipulable and maintainable combinatorial one so that we can handle and solve it mathematically.

In chapter 2, we have discussed some concepts helping to meet this challenge. Here, we state the meaning of various biological terms which aids the reader to get an clear idea on what we are going to deal with. This chapter also introduces the reader with the concept of meta-heuristics. Here, we also discuss several types of local and global search algorithms which we use later.

In chapter 4, we have staged our algorithms for finding a genome sequence which contains large number of unique restriction sites and its functionality is same to the original sequence. Prior to that, we brief why we have motivated to apply elitist meta-heuristics. We have also discussed about the new objective introduced in this thesis.

The purpose of Chapter 5 is to paint a picture of performance of our suggested algorithms. We have conducted extensive experiments with small as well as large viral

sequence. We have presented experimental details with possible discussion about their behavior.

Chapter 6 provides a software tool along with the works of [19, 20]. A brief overview of the software has been discussed in this chapter.

6.1 Summary of Contribution

In this thesis, we apply some *elitist meta-heuristic techniques* to find synthesized genome sequences. To achieve this, we first map the problem into a *multi-objective optimization problem*. We have proposed 3 different algorithms to find these genome sequences. All of our proposed meta-heuristic algorithms work in two steps. Firstly, our algorithms generate different population sets based on different objectives. We have proposed three different meta-heuristic approaches/algorithms to generate these populations. Secondly, combine these populations and apply multi-objective meta-heuristic approach on those combined populations. Besides these elitists meta-heuristic algorithms, we have also used some other multi objective optimization techniques and conducted some experiments with these algorithms.

We have introduced a new objective of the problem which is more meaningful and significant from the biological viewpoint. To examine a genome sequence, biologists want to cut the sequence to equal sized subsequences. Equal sized subsequences are more helpful for biologists to examine. Say, a sequence is cut into some big and some small subsequences. Obviously, big subsequences are very hard to examine by biologists. On the other hand, the little sequences are not interesting for biologists. For this, we have introduced a new objective which can be more interesting for biologists.

We have carried out extensive experiments with different viral genome sequences considering our newly developed algorithms as well as existing meta-heuristic algorithms. So far, the existing algorithms were experimented with the small viral sequences in the literature. To analyze the performance of those algorithms, we have conducted our experiments with some clinically important viral sequences. Since the lengths of these clinically important viral sequences are not large enough, we have extended our experiments with some large viral sequences.

We have compared the performance of our newly designed and developed algo-

rithms with other traditional meta-heuristics in the literature. We have conducted suitable statistical tests for every comparison.

We have provided an open source java software tool for biologists. We believe that this tool will help biologists a lot for examining the viral plasmid sequences and will contribute for gene therapy and vaccine development. To the best of our knowledge, the synthesized genome designed by this tool would be most attractive to the biologists because of cost and other criteria.

6.2 Future Research

Recall that, number of base pair changes is not always meaningful for biological viewpoint. The number of attempts is meaningful instead. It needs to introduce new objective of URSPP. We could not formulate this as because we have not enough information about how this attempts works. At the current setting, we could not use/formulate this new objective, because we do not know how an attempt can be used to change a number of base pairs. It will be helpful for biologists to add this 'attempt' as a new objective to solve URSPP. So introducing a new objective such as 'Bio-Method needed to change a genome sequence' and working on it may lead a good idea for future researchers. For this, it needs to know the base pairs that can be changed by an individual attempt. This would require some more information from the biologist which must be provided to the algorithm as input. As it is seen that, elitist approaches have given better results than the previous works in the literature, it will be fine to apply other meta-heuristics through elitist approach.

Over the last few decades synthetic biology and computer science have made great strides. Biologists are now in a position of being faced with biological information of such volume that it is impossible to analyze the information by pen and paper. Hopefully this thesis along with the software tool will serve as a step in that direction. Moreover, researchers can get a new dimension of how to solve this problem. We believe that collaboration of biologists and computer scientists will do wonders for mankind in this century.

BIBLIOGRAPHY

[1] Hogeweg, P., "Simulating the growth of cellular forms," *Simulation*, vol. 31, no. 3, pp. 90–96, 1978.

[2] Hogeweg, P., and Searls, D. B., "The roots of bioinformatics in theoretical biology," *PLoS Computational Biology*, vol. 7, no. 3, 2011.

[3] Jones, N. C., and Pevzner, P. A., *Introduction to Bioinformatics Algorithms*. The MIT Press, 2004.

[4] Gerstein, M., *Bioinformatics Introduction*. University of Yale Press, 1999.

[5] http://www.bioinformatics.org/wiki/Bioinformatics, last accessed at 11:00 PM, 10 June, 2013.

[6] http://en.wikipedia.org/wiki/Bioinformatics, last accessed at 11:00 PM, 10 June, 2013.

[7] Dayhoff, M. O., Barker, W. C., Schwartz, R. M., Orcutt, B. C., and Hunt, L. T., "Data base for protein sequences," in *AFIPS National Computer Conference*, 1976, pp. 261–266.

[8] Waterfield, M. D., Scrace, G. T., Whittle, N., Stroobant, P., Johnsson, A., Wasteson, A., Westermark, B., Heldin, C. H., Huang, J. S., and Deuel, T. F., "Platelet-derived growth factor is structurally related to the putative transforming protein p28sis of simian sarcoma virus," *Nature*, vol. 304, pp. 35–39, 1983.

[9] Fleischmann, R. D., Adams, W. D., Clayton, R. A., Kirkness, E. F., Kerlavage, A. R., Bult, C. J., Tomb, J. F., Dougherty, B. A., Merrick, J. M., and et al., "Whole-genome random sequencing and assembly of haemophilus influenzae rd." *Science*, vol. 269, pp. 496–512, 1995.

[10] Doolittle, R., Hunkapiller, M. W., Le, H., Devare, S. G., Robbins, K. C., Aaronson, S. A., and Antoniades, H. N., "Simian sarcoma virus onc gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factorsimian sarcoma virus onc gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factor," *Science*, vol. 221, pp. 275–277, 1983.

[11] http://bioinformaticsweb.net/, last accessed at 11:00 PM, 10 June, 2013.

[12] Casey, R. M., "Bioinformatics in computer-aided drug design," 2005.

[13] http://www.synbiosafe.eu/uploads///pdf/EU-highlevel-syntheticbiology.pdf, last accessed at 11:00 PM, 10 June, 2013.

[14] http://royalsociety.org/, last accessed at 11:00 PM, 10 June, 2013.

[15] Gardner, T. S., Cantor, C., and Collins, J. J., "Construction of a genetic toggle switch in escherichia coli," *Nature*, vol. 403, no. 6767, pp. 339–342, 2000.

[16] Cai, Y., Lux, M. W., Adam, L., and Peccoud, J., "Modeling structure-function relationships in synthetic dna sequences using attribute grammars," *PLoS Computational Biology*, vol. 5, no. 10, 2009.

[17] Montes, P., Memelli, H., Ward, C. B., Kim, J., Mitchell, J. S. B., and Skiena, S., "Optimizing restriction site placement for synthetic genomes," in *CPM*, 2010, pp. 323–337.

[18] Kuhn, H. W., and Yaw, B., "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[19] Sharmin, M., Afrin, M., and Rahman, M. S., "Local search techniques for placing unique restriction sites in synthetic genomes," in *BICoB*, 2012.

[20] Sharmin, M., and Rahman, M. S., "Placement of unique restriction sites in synthetic genomes using multi-objective optimization," in *BIBM*, 2012, pp. 1–4.

[21] http://www.blueheronbio.com, last accessed at 11:00 PM, 10 June, 2013.

[22] http://www.origene.com/about/corporate/, last accessed at 11:00 PM, 10 June, 2013.

[23] http://www.geneart.com, last accessed at 11:00 PM, 10 June, 2013.

[24] Anand, I., Kosuri, S., and Endy, D., "Genejax: A prototype cad tool in support of genome refactoring," 2006.

[25] Chan, L. Y., Kosuri, S., and Endy, D., "Refactoring bacteriophage t7," *Molecular Systems Biology*, vol. 1, 2005.

[26] Evans, P. M., and Liu, C., "Sitefind: a software tool for introducing a restriction site as a marker for successful site-directed mutagenesis," *BMC Mol. Biol.*, vol. 6, no. 22, 2005.

[27] Kuldell, N., and Lerner, N., *Genome refactoring*. Morgan and Claypool Publishers, 2009.

[28] Andrianantoandro, E., Basu, S., Karig, D. K., and Weiss, R., "Synthetic biology: new engineering rules for an emerging discipline," *Molecular Systems Biology*, vol. 2, pp. 2–28, 2006.

[29] Heinemann, M., and Panke, S., "Synthetic biology:putting engineering into biology," *Oxford Bioinformatics*, vol. 22, pp. 2790–2799, 2006.

[30] Mens, T., and Tourwé, T., "A survey of software refactoring," *IEEE Trans. Software Eng.*, vol. 30, no. 2, pp. 126–139, 2004.

[31] Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J., "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing*, vol. 8, no. 2, pp. 239–287, 2009.

[32] Blum, C., and Roli, A., "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[33] Glover, F., "Future paths for integer programming and links to artificial intelligence," *Computers & OR*, vol. 13, no. 5, pp. 533–549, 1986.

[34] Reeves, C. R., *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Press, 1993.

[35] Osman, I. H., and Laporte, G., "Metaheuristics: A bibliography," *Annals of Operations Research*, vol. 63, pp. 511–623, 1996.

[36] Stützle, T., *Local search algorithms for combinatorial problems - analysis, improvements, and new applications*, ser. DISKI. Infix, 1999, vol. 220.

[37] Veldhuizen, D. A., and Lamont, G. B., "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125–147, 2000.

[38] Luke, S., *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[39] Holland, J. H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, illustrated ed. University of Michigan Press, 1975.

[40] Srinivas, N., and Deb, K., "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[41] Aho, A. V., and Corasick, M. J., "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[42] http://www.ebi.ac.uk/genomes/virus.html, last accessed at 11:00 PM, 10 June, 2013.

[43] Roberts, R. J., Vincze, T., Posfai, J., and Macelis, D., "Rebase:a database for dna restriction and modification: enzymes, genes and genomes," *Oxford University Press*, vol. 38.

[44] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in FORTRAN - The Art of Scientific Computing, 2nd Edition*. Cambridge University Press, 1992.