A META-HEURISTIC OPTIMIZATION TOOL FOR SIMPLIFIED PROTEIN STRUCTURE PREDICTION

by

Gurpreet Lakha

PhD., Panjab University, Chandigarh, India, 2015

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS OF THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

UNIVERSITY OF NORTHERN BRITISH COLUMBIA

March 2019

Abstract

Meta-heuristic algorithms give a satisfactory solution of complex optimization problems in a reasonable time. They are among the most promising and successful optimization techniques. However, some problems are highly complex and require improved techniques. A careful analysis of the existing meta-heuristic algorithms and hybridization among them may facilitate the research in this direction. To test this hypothesis, the author of the thesis developed a computational tool using a few meta-heuristic algorithms where these algorithms can be analyzed in detail and possible hybridization among them can be created. As a case study, the tool is developed for simplified protein structure prediction. The proper working of the software is demonstrated by optimizing the two sets of standard previously reported sequences. Along with testing and analyzing meta-heuristic algorithms, the tool can be used for simplified protein structure prediction.

Contents

List of Tables

List of Figures

Acknowledgement

I would like to express my deep sense of gratitude to my worthy supervisor Dr. Alex Aravind for allowing me to carry out my work under his invaluable guidance. His interest in scientific problems and moral inspiration have made current work possible. I am extremely indebted to the Chairperson, Department of Computer Science, Northern University of British Columbia, Prince George, for providing all the necessary facilities to complete this work. Also, a special thanks is extended to Dr. Stephen Rader and Dr. George Jones for offering their valuable assistance as my committee members.

I am also indebted to my labmates Apratim, Ashlin, Conan, Brae, Davis, Suresh, Shanthini, Darshik, Rahim, Ketan, Arthi, Raja and Rodrigo for their support, discussion, encouragement, and pleasant company. Special thanks to the teaching and nonteaching staff members, UNBC, for their help in all possible ways.

It is a pleasure to acknowledge the good company of fellow researchers and my friends Hardev, Gurjeet, Ashish, Ramneesh, Johny, Priyanka, Amit, Navjot, Bashet, Dorob, Bryan and many others.

Finally, I want to express my deepest gratitude to my family for their love, support and encouragement, without which this work would not have taken this shape.

Prince George, BC
December , 2018 *(Gurpreet Singh Lakha)*

Dedicated

to

My Parents

Chapter 1

Introduction

The work done in this thesis is primarily focused on protein structure prediction using meta-heuristic algorithms. Protein Structure Prediction (PSP) determines the configuration of a folded protein without actually going through the folding process [3]. The interest to work on PSP problem for my thesis was originated partly from a Bioinformatics course that I took in the first term, and partly from my background in physics. Molecular structure optimization was a part of my Ph.D. thesis in physics. In the Bioinformatics project, my group expanded the pre-developed tool in our lab for protein folding using a genetic algorithm. Once I decided to work on protein folding using meta-heuristics for my thesis, I was contemplating whether to expand the existing tool or develop a new tool from scratch. I decided on the latter approach for several reasons: (1) it is hard to work with code written by someone else, (2) the existing code was limited in features[1] and (3) I thought it would be a wonderful opportunity to build a comprehensive software from scratch. With this brief history of how I ended up choosing my research topic, next I will discuss the background information needed to present my research contributions.

[1]For example, (1) the graphical user interface is not effective for user interactions,(2) only one algorithm was implemented and hard to expand for multiple algorithms, etc.

1.1 Optimization

In general, the process of finding the best solution from an available set of alternatives under some conditions is called optimization. Most natural processes work to attain the best (optimized) state/solution from all the potential alternatives. Many of the physical, biological and psychological processes, and even cultural and historical events are inspired by the idea of optimization. Physical processes favor minimized energy states, and sometimes lead to symmetry, which exhibits optimization in physical phenomena. Similarly, evolutionary activities of animals and plants follow the concept of optimization for their survival. Moreover, problems related to profit/loss in business, some complex cognitive problems in psychology, social welfare in the context of some planning, etc., can be resolved under a suitable optimization framework. Since optimization is the core concept in a wide range of natural phenomena, it can be applied to solve and analyze a diverse range of complex problems [4–11] related to the fields of physics, mathematics, computer science, economics, etc.

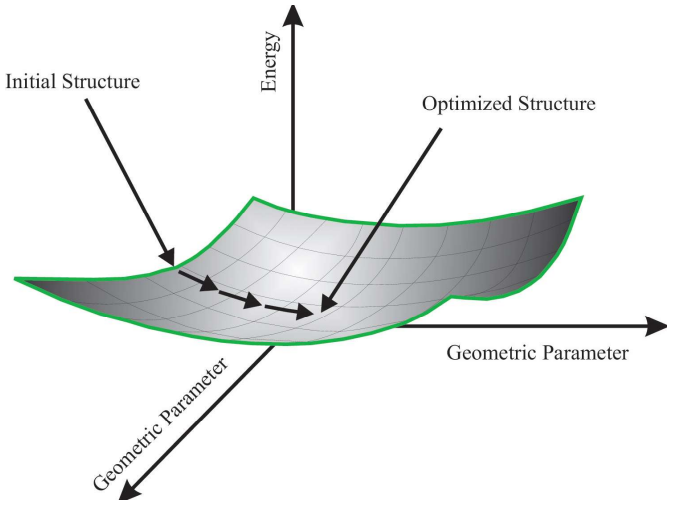

Figure 1.1: Energy minimization of a molecular structure *w.r.t.* two geometric parameters

Specifically, optimization in economics is the search for the highest achievable performance under given constraints by escalating the profit and minimizing undesired factors. In mathematics and computer science, optimization is the process of maximizing or minimiz-

ing a real function with the help of systematically allowed input values. Figure 1 explains the energy optimization process of a molecular structure under two geometric parameters (variables). Problems considered under optimization generally consist of three basic elements: 1) an objective function which is minimized or maximized in the optimization process, 2) a set of variables (unknowns) which changes the objective function and 3) a set of constraints [12]. Constraints are used for limiting the search space; otherwise, the search space will become very large. Usually, optimization problems are based on a single objective function, but a multi-objective function is also possible. In terms of these basic elements, the optimization problem can be described as the search for values that minimize or maximize the objective function(s) based on some constraints.

The applicability of optimization to a variety of problems gave birth to different optimization techniques. Some widely used techniques are dynamic programming, integer programming, stochastic programming, evolutionary algorithms, etc. In linear programming, both objective functions and constraints, are linear in nature, whereas, non-linear programming includes at least one non-linear function which could be the objective function or some/all constraints. Geometric Programming (GP) is a particular type of non-linear optimization where decision variables must be strictly positive quantities. Dynamic programming is an effective type of optimization technique which breaks down the problem into a collection of simpler sub-problems. Dynamic Programming is extensively used to solve different bioinformatic problems such as sequence alignment, protein-DNA binding, protein folding, etc.

In Integer Programming, variables are restricted to integers, and if the function and constraints are linear, it is called as Integer Linear Programming (ILP). This is the most common technique used to solve real world optimization problems. Some optimization problems require to optimize more than one objective functions which led to the development of multi-objective programming. There has been a rapid increase in the use of evolutionary computation to address difficult problems. These efforts are generally followed by three main lines of investigation: (1) genetic algorithms, (2) evolution strategies, or (3) evolutionary program-

ming. The solution in evolutionary techniques is searched by sampling from a probability distribution. Stochastic optimization is a type of programming that is helpful when some of the data incorporated into the objective or constraints are uncertain [13].

The search for these optimization techniques, appears to have started at least as early as the time of Newton and Leibnitz with the invention of iterative methods of differential calculus. Lagrange, Cauchy, Carroll, Fiacco, and McCormick have contributed a lot to Non-Linear programming. The Simplex method suggested by Dantzig is one of the most widely used algorithms in the field of linear programming. The dedicated work of these and other investigators consequently appeared in the form of a number of optimization algorithms. In computer science most of these methods are divided into three categories: (1) optimization algorithms which use some definite procedure to find the solution; (2) iterative methods that converge to a solution; and (3) heuristics (or meta-heuristics) that provide an approximate solution after a given number of steps. The choice of method or algorithm for an optimized solution is usually problem-dependent. Some conventional problems can be solved using traditional mathematical techniques, such as Linear Programming (LP), Non-Linear Programming (NLP), and Dynamic Programming (DP), and guarantees to reach global optima. However, most real world problems are non-linear, and the search for global optima is feasible only for small sized instances of the problems. These problems could require a huge amount of computational work. Heuristic/Meta-heuristic algorithms can simply handle many of these complex problems. In recent years, meta-heuristic algorithms are extensively used as an alternative approach to the classical methods for solving optimization problems that include uncertain, stochastic, and dynamic information. This motivates the new researchers to conduct further investigation in this direction. Developing a tool based on some meta-heuristic algorithms for the optimized solution will be a good step. The tool can be used to get the optimized solution using different algorithms and the research can be done by testing and analyzing these algorithms in their base forms or with addition/removal of different constraints or by mixing possible features of different algorithms. Therefore, we decided to build an interactive software tool using some meta-heuristic algorithm which can

be used for research as well as educational purposes. Building a generic tool is a complex task, and, therefore, we decided to build the tool for a specific problem.

1.2 Protein Structure Prediction (PSP)

As mention in [14], PSP is one of the top 125 problems, and is still unsolved. PSP is a complex task, and scientists from last sixty years have persistently tested different experimental as well as theoretical, techniques to search their optimized structures. Actually, the experimental work on proteins was started in the nineteenth century, but was somewhat enhanced with denaturation studies under different experimental conditions during 1930s [15]. A very first report on high-resolution protein structures was published by Kendrew and Perutz in 1958 [16]. For their work, they were awarded 1962's Nobel Prize in Chemistry. However, on the theoretical side, not too much work was reported until 1969 when Levinthal posed a very important question related to folding speed. He pointed out that protein folding speed is too high as compared to the exhaustive search in the conformation space [17]. This is known as the Levinthal paradox. After the Levinthal paradox, it was suggested that protein folding speeds up due to local interactions, and experimental evidence verified the existence of intermediate states present during protein folding. Some tests with calorimetric protein folding did not observe any intermediate state as recorded for tRNA [18], but later partially folded states were observed during protein unfolding. Garel and Baldwin also confirmed the presence of intermediate between native and unfolded state [19]. Hughson *et al.* designed a model of molten globule [20], and explained that only a portion of the polypeptide has resulted through the intermediate states. Some studies also accepted the existence of partially folded transmission states [21–24]. Nowadays, there are huge numbers of reports on PSP, both experimental and theoretical. Different models have also been suggested to accomplish PSP in fast and easy ways.

1.3 Protein Structure Prediction Using HP Model

The HP model was proposed by Dill in 1989 [25]. It is one of the simple and effective models for protein folding. The model is based on the hydrophobic effect [2]. It divides the amino acids into two categories based on their hydrophobic (H) and hydrophilic (polar) (P) nature. Since the model is based on the nature of amino acids, for simplicity, it considers amino acids' residues of the proteins as beads to construct the molecular structure of proteins. The structure is similar to the molecular structure of small molecules, whereas, in that case atoms are combined as beads. The search space in this model can be restricted by the use of different lattice types, *i.e.*, the positions of beads are constrained to these lattice sites. The PSP is a NP-hard problem [26, 27] and with the use of HP model under such constraints, it is proved to be NP-complete in 2D as well as 3D lattices [28, 29]. The model is widely used, as it is able to predict some behavioral features of the proteins and is well known for its simplicity. However, still, not much work has been done in developing software applications on the basis of the HP-model. Therefore, we decided it as the primary objective of my thesis work via incorporating various meta-heuristic algorithms.

1.4 Contribution of The Thesis

The main contribution of this thesis is the design and development of a software framework that enables us to implement various meta-heuristic techniques to study the optimal folding of proteins based on the HP-model. The benefit of this contribution is twofold: (1) for educational purposes and (2) as a research tool.

As an educational tool, it can be used to teach protein folding using the simplified HP model. Specifically, the tool can be used to:

- teach different meta-heuristics algorithms in the context of protein folding. Using

[2]Hydrophobic effect is referred as the tendency of a nonpolar (solute) molecules to get together in water (polar solvent) rather than to dissolve individually. This causes the poor solubility of nonpolar substances in water as compared to polar substances.

the tool, we can demonstrate the design of new and existing popular meta-heuristic algorithms by combining various tricks and sub-heuristics.

- demonstrate protein folding using the HP model. The HP model is one of the simple models that is suitable to teach hydrophobic and hydrophilic involvement in protein folding. The visualization component of the tool adds value for the effective teaching of the folding process. The 3-dimensional optimized structure can be viewed from different angles and easy to analyze. The graphical display of the structure is also helpful for checking the correctness of the meta-heuristic implementation.

As a research tool, it can be tuned and improved in a number of ways. For example:

- It can be used to compare the effectiveness of different meta-heuristic algorithms for protein folding.
- Better hybrid optimization procedures can be searched from the available algorithms and additional features.
- New algorithms/features can easily be added and tested along with existing ones in order to search for better optimized procedures.
- The conclusions drawn from newly constructed optimization procedures can further be applied and analyzed for other optimization problems.
- Different constraints, such as twin removal, mirror symmetry, etc., can be applied and tested with multiple algorithms and better conditions for optimizing under an algorithm can be searched.
- The predicted optimized structure can be further used as the input structure to carry out fast ab-initio calculations where consideration of hydrophobic effect can be omitted.

To achieve the above-stated goals and benefits we have to address the following research challenges:

1. How do we design a graphical user interface for interactive inputs and output?

2. How do we implement different algorithms and identify the common parts among different algorithms?

3. How do we identify and implement the main components for parallelism on computationally expensive steps?

4. What procedure can be used to escape from local optima?

5. What is the condition required for an algorithm/algorithmic step to be used in hybridization, and what criteria can be applied to implement them for hybridization?

6. What procedure(s) have to be used to save and display the output results based on user requirements?

The rest of the thesis is organized as follows. Chapter 2 provides a comprehensive background by explaining PSP, HP model, meta-heuristics, and related work. Next, in Chapter 3, I present different meta-heuristic algorithms employed in my work. Chapter 4 presents the design and implementation of the software tool. The correctness of the software has been confirmand by optimizing the previously reported proteins' sequences. A brief analysis of the results of the implemented algorithms is given in Chapter 5. The conclusions and future work are presented in Chapter 6.

Chapter 2

Related Work

This Chapter is divided into five sections. Section 2.1 discusses Protein Structure Prediction (PSP) and major interactions involved in protein folding. The next section covers the HP model, its importance and some of its limitations. Meta-heuristic is discussed in Section 2.3. Section 2.4 presents some reports related to PSP optimization techniques. Finally, the summary of the chapter is given in Section 2.5.

2.1 Protein Structure Prediction (PSP)

Proteins are the building blocks of our body which form the essential components of all tissue structures of cell membranes and genetic material. In the living organisms, proteins serve different functions, such as muscle contraction, transmit nerve impulses, building DNA and RNA, immune protection, cell growth [30], transport ions, and molecules [31], etc. It has been observed in a number of reports that proteins function have a parallel relationship either with the sequence of amino acids or with the structure of folded protein [32, 33]. A small change in the sequence, can cause inimical disease [34] and a number of diseases such as sickle cell anemia, neurodegenerative disorders, age-related diseases (for example Alzheimer's and Parkinson's diseases), etc., are considered to be caused by incorrect protein folding [35–39]. These studies insist to understand the proteins folding and their functional

dependency on their structures.

Protein formation is a two-step process of transcription and translation. A messenger RNA, formed by the unwinding of a DNA strand (transcription), when interacts with ribosome results in a protein or a sequence of amino acids. This sequence of amino acids is called the primary structure of proteins. At a proper temperature and solvent composition, the primary structure folds to minimized energy structure called the native or tertiary protein structure. Proteins always fold to their unique shapes, however, the mechanism involved in the folding is not yet understood. Also, the misfolding of proteins is responsible for a number of health related issues. Therefore, it is important to understand how a sequence acquires a particular minimized energy structure. Knowledge of the folding mechanism, when employed to a given primary structure aids prediction of the native structure. This search of native structure from a given sequence of amino acids (primary structure) is known as protein structure prediction.

Protein Structure Prediction (PSP) is important for both biochemistry as well as medicinal purposes. Proteins perform a variety of tasks in the human body, and their functions depend on their structure [32, 33]. Therefore, knowledge about the structure of proteins can help in many ways to improve/new drugs. The selective structure-based drug design, for example; a drug designed targeting the parasites based on structural differences between host and parasite can exploit the parasites without causing damage to the host. Structure prediction will also be helpful for comparing proteins and be used to establish the evolutionary relationship with other proteins and protein families.

After realizing the importance of PSP, a number of research groups started to work in this direction. Some reports suggested that folding is a step-by-step process and confirmed the existence of intermediate states [40]. The presence of intermediate states also supports Levanthal's argument that protein follows the shortest path to native state. It can be assumed that there might be some genetic code that resides in the sequence or somewhere in the generic material which helps proteins to get their unique shape. Ramachandran tried to figure out this hypothesis by using the dihedral angle between different residues. He suggested a

range of dihedral angle regions, called Ramachandran plot, which might be followed by the amino acid sequence during folding process [41]. Later some arguments against these regions have also been reported [42]. Reports also purposed that the tertiary structures are followed by secondary structures [43], as a significant amount of secondary structures have been observed at intermediate states. Subsequently, this argument has also been ruled out and confirmed that it is true only for certain proteins and not in general. However at present, as a consequence of these efforts, there is a significant amount of articles pertaining to PSP and with the aid of the advanced experimental and theoretical techniques PSP is somewhat achievable.

The experimental techniques mainly employ; Electron microscopy, X-ray crystallography, and NMR-spectroscopy. X-ray crystallography collects the information about electron density through 2D diffraction pattern at different orientations. Electron microscopy has higher resolution than a microscope and is used to obtain the image of the sample. In NMR spectroscopy different electron shielded environments around the nuclei give rise to different chemical shifts and is best suitable for studying the structure and protein dynamics [44–46].

Experimental analysis of large molecules, however, is not an easy job. Therefore, more efforts are needed to improve theoretical approaches. Theoretical studies for PSP need to take care of different types of interactions (electrostatic, Van der Wall, hydrogen bonding, hydrophobic effect, peptide bonds, etc.) and surrounding environmental factors (pH, ionic composition, and concentration, solvents dielectric constant, etc.). The role of each type of interaction is still not clear, but evidence shows that the hydrophobic effect plays a major role in protein folding [15]. However, this conclusion has been drawn after a series of experiments and observations. In early studies, the folding was considered to be due to long-range electrostatic interaction but later attributed to hydrogen bonding. Over time, the role of the hydrophobic effect became clearer and is considered as a major factor responsible for protein folding [15].

2.1.1 Major Interactions Involved in Protein Folding

Predicting native protein structure is a collective response from different interactions and environmental factors originated from a large number of atoms/molecules/ions. During the time of the 1920s, only acids and bases were known and the interactions involved in protein folding were considered between oppositely charged ionic side chains (called salt-bridges) [15]. Apparently, the first idea of interaction responsible for protein folding could be electrostatic interactions. The first electrostatic interactions' based model was purposed by Linderstrom-Lang [47] and later improved by some other researchers [48–50]. There are some observations which favor this model calculation [51], however, some contradictory results have also been reported [52, 53].

Actually, the role of electrostatic interaction is difficult to analyze because the method used which changes the concentration (adding salt) and produce two-way effects; increase of dielectric constant increases the net charge on the native state and hence destabilize the native state but at the same time it can lead to stability if it increases the ion pairing in the protein structure [15]. Also, the dielectric property alone cannot be used to diagnose the electrostatic interactions because it is also associated with other solvent properties [15]. Some observations cleared the role of ion binding on stability [54, 55]. However, reports from Jacobsen and Linderstrom-Lang [51] and structural study by Barlow and Thornton [56] explained that ion-pairing is not the major force in protein folding. Barlow and Thornton [56] study reveal that only 17% of ion pairs exist inside the core and this amount is too small to be considered as the major cause of protein folding.

The second major type of force responsible for protein folding could be hydrogen bonding (H-bonds). This type of interactions can occur when hydrogen attached to a highly electronegative atom come closer to another electronegative atom. Hydrogen bonding is common in inorganic molecules such as water as well as in organic molecules like DNA. In proteins, hydrogen bonds are considered to be important because the backbone atoms of each residue can form two or more hydrogen bonds. Since helices are the common fea-

ture of globular proteins, it is obvious to consider hydrogen bonding as the folding driving force [57].

In 1936, Mirsky and Pauling [58] predicted folding in alpha helical and beta sheet structures by keeping in mind hydrogen bonding as the major force between carbonyl and amide groups which later supported by X-ray crystal structures of globular proteins [59]. It was suggested that protein folding follows a hierarchy from a primary structure to secondary structure and then leads to tertiary structure. This hierarchy has been widely adopted in predicting the native state using the computational strategy from a given sequence. Based on hydrogen bonding, different models [60–62] have been suggested for understanding the helix formation and beta-sheet formation. However, Kauzmann [63] insists that hydrogen bonding is not the dominating force because there is no explanation related to the inter-chain H-bonds at folded state having lower free energy than denaturant state.

Hydrogen bonding analysis was also not consistent with the other observations such as alcohol being more hydrophobic than water and if hydrogen bonding is the dominant force it should destabilize proteins [15]. Instead, alcohol enhances helix formation. Also, one percent of dodecyl sulfate can unfold the protein, however, it does not destabilize the helix [15]. Dioxane is a hydrogen-bond acceptor and therefore should not denature proteins, but it does. These solvent studies point out the fact that hydrogen bonding is not the dominant force responsible for protein folding. However, about 66% of backbone polar groups are buried in the hydrophobic interiors of proteins which are, generally, hydrogen bonded to a partner as burying polar groups without hydrogen bonding in non-polar media is energetically expensive [64]. These observations indicate that hydrogen bonding must play an important role during folding or for stability but it can not be considered as the major force responsible for protein folding. Its role still requires a proper comprehensive explanation.

The above discussion imply that the hydrogen bonding and electromagnetic interactions are not the driving force for protein folding. Consequently, the possible cause could be found in the surrounding environment. It was stated that almost 85% of non-polar side chains in proteins are buried in the interior of the protein [65]. These non-polar groups shielded from

solvent and tightly packed with other polar and nonpolar groups. Also, most of the content in the cell is water and aversion to non-polar molecule from water (polar) molecules can be considered as the possible cause of non-polar interior and can further lead to protein folding. Based on the aversion to water (hydrophobic effect) from the interior of the protein, Dill proposed the HP model [25] which supported by a number of observations.

2.2 HP Model

The HP model [25] was developed using lattice statistical mechanics for heteropolymer folding such as a protein [15]. In the model, twenty amino acids are divided into hydrophobic (H) and hydrophilic (P) types and energy is calculated based on non-bonded hydrophobic interactions. Aversion to water molecules from the hydrophobic amino acids was used as the underline concept for protein folding. The idea of aversion to water from non-polar residue was first considered in case of micelle formation and the same was supported by Kauzmann [66, 67] for explaining protein folding. Some experimental observations such as non-polar solvents denaturing proteins and stability of proteins is affected at high as well as at low temperature, are also consistent with Kauzzmann's view.

The idea was developed from the fact that globular proteins are somewhat soluble in water and not like fibrous or membrane proteins. However, not all proteins are soluble; a molecule with majority of hydrophilic residues will prefer to dissolve. Folding is assumed to be driven by the association of hydrophobic monomers to avoid solvent molecules and opposed by the chain configurational entropy [15]. The water molecules near polar residues become more ordered and increase the entropy of the water molecules. This leads to collection of non-polar side chains in the interior of proteins, which in turn causes the collapse of the protein [15]. In other words, as described in [68], when an a nonpolar molecules come closer to the water molecules, the hydrogen bonds between water molecules get disturbed, and water molecules get collected around the nonpolar molecules. This form a cage around the non-polar molecules which is not favored by the Second Law of Thermody-

namics. Therefore, the non-polar molecules start accommodate together. This tendency of nonpolar molecules to get-together around themselves rather than to dissolve individually in the polar solvent is called the hydrophobic effect. The effect actually arises due to the H-bonding nature of water and not because of solute-solute interactions. However, for convenience, the biochemists use the term hydrophobic effect [68].

The protein structures minimization process minimizes the free energy of proteins instead of potential energy, as it involves thermodynamic quantities, and in configurational space, we deal with the degree of freedom and hence entropy is an important parameter to consider. From the chemical reaction viewpoint, a reaction is favored if the change in free energy is negative. In this sense, a solvent can dissolve a solute if the change in free energy (ΔG) is negative. The change in free energy of a reaction is defined as $\Delta G = \Delta H - T\Delta S$. ΔH is the change in enthalpy, ΔS is the change in entropy and T is the temperature. The reaction is enthalapically favorable if ΔH is responsible for negative ΔG (and magnitude $>\Delta S$) and said to be entropically favorable if the second term has a larger contribution for negative free energy, e.g., freezing of water to ice lead to increase in entropy of water molecules.

2.2.1 Significance of HP Model

HP model is subjected to a huge amount of literature, however, there are very few studies [69] which examine the success of the HP model in terms of correlations between predicted and actual structures. These studies also indicate that the simulated structures with HP model are far from the optimal stage. The reason can be attributed to the presence of other forces of interaction which are not considered in this model. However, the model only considered the hydrophobic effect. The hydrophobic effect as the driving force in protein folding is supported by a number of studies [15]. One of the important findings of the HP model is the prediction of the hydrophobic core. There are other findings of proteins structure such as uniqueness in structure, two-state-cooperativity, compactness of secondary structures, etc., which support the model.

The uniqueness of native protein structure is an important characteristic of protein structure which differentiates them from other polymers. Although the exact cause of uniqueness is still unknown, the simulated results using HP model show some agreement with the uniqueness property of the native state [15]. The relationship between the length of the sequence and number of possible conformations shows that conformations' number calculated with the HP model is a strongly decreasing function of the increase in the length of the sequence. This implies that maximizing the number of H-H contacts in a sequence results in fewer possible conformations and thus hydrophobic interactions might lead to a unique tertiary structure.

Some experimental studies show that secondary structure is correlated with protein compactness [57]. Studies have confirmed that compactness stabilizes secondary structures, although, in the absence of hydrogen bonding, only a small correlation has been observed between secondary structures and those in proteins. The lattice simulations also predicted the collapse of polymer chains which helps in the formation of secondary structure. It is pointed out that if helices and sheets are defined using proper methods, compactness also results in some secondary structure. This demonstrates that the compactness and steric constraints are responsible for hydrophobic collapse and further responsible for the secondary structure. The hydrophobic collapse is also responsible for the decrease of internal dielectric constant, which increases the hydrogen bonding, helical dipoles, and other electrostatic interactions within the core, and stabilizes the secondary structures [57].

Cooperativity can be divided into one-state or two-state transitions. One state cooperativity means that the population vs temperature distribution has only one peak over states (native, first excited, second excited), however, in two state cooperativity, there are two major peaks; one corresponds to the denaturant state and other correspond to the native state. This implies that there are two types of transitions, *i.e.*, at the mid-point two identifiable states are populated. This implies that there exists a free energy barrier between these two states. Protein folding show two-state-cooperativity. Homopolymers do not have this ability to form two states. This two-state nature of proteins is now explained on the basis of the

ability of a sequence consisting of a hydrophobic core and a polar surface [70].

Some experimental structures correlation explained that a protein can be mutated substantially without changing its fold [57]. This degeneracy in structure means that hydrophobic monomers are largely interchangeable with each other, and polar monomers are interchangeable with other polar monomers. Thus the fold is heavily dependent upon the hydrophobic and polar nature of the amino acids rather than the actual amino acids. Mutation in denatured states is also explained on the basis of the HP model [15].

2.2.2 Limitations of HP Model

Even with the above-mentioned successes, the hydrophobic effect fails to explain a number of important observations. Based on HP model prediction a protein should fold more strongly as the temperature increased, however, opposite behavior has been observed above room temperature. Also, hydrophobic interactions are able to explain the origin of secondary structures present inside the protein core. The model also fails to explain the pressure dependence of protein stability [15].

In 1962, even before the development of the HP model, Tanford [71] and Brandts [72] calculated the stability and showed that hydrophobicity predicts protein stability. However, have greater strength than the measured values. This implies that the magnitude of the hydrophobic force is much larger or there exists an opposite force which opposes folding [15].

The stability of the globular state also depends on the composition and length of the chain. Theoretical analysis also predicts that there should be optimal hydrophobicity for maximum stability [73]. Stability also decreases if hydrophobic residues distribute in excess over the surface, which requires filling the core for stability. However, it was still a question that whether the hydrogen bonded polar groups make a favorable contribution to globular protein stability; some experimental studies concluded the significant contribution of hydrogen-bonding groups to protein stability [74]. Lazaridis *et al.* concluded that stability

has no or very little contribution from polar groups [75].

No doubt hydrophobicity give a number of successful explanations, however, there is still room for questions because of unknown contribution from other types of interactions. Even with these questions, there are a number of triumphs of the model and further research on HP model will be helpful for better handling the questions related to PSP. Therefore, taking PSP using HP model, as a problem under consideration for comprehensive analysis of optimization algorithms, we conducted a literature survey about the available PSP tools and especially, HP model algorithms/tools, so that, some possible features can also be put together for conducive optimization. This is discussed in Section 2.4. However, before that, there is a brief discussion about meta-heuristics.

2.3 Meta-Heuristic

Heuristic and meta-heuristics are popular techniques for solving optimization problems. These optimization techniques are intuitive methods as they make few assumptions about the problem. Specific heuristics are problem dependent; they are designed and applicable to a particular problem [76]. However, meta-heuristics deal with more general approximate algorithms and a single algorithm can be utilized to solve a number of optimization problems.

The word meta-heuristic is composed of the two Greek words " heuriskein" and "meta". *Heuriskein* means the art of discovering new strategies and the suffix *meta* means upper level methodology [77]. Thus the nomenclature of the words clears that meta-heuristic is an upper level methodology that can be used as a guided technique to design heuristic for solving a specific optimization problem. The term meta-heuristic was first time introduced by F. Glover in his paper [78]. Meta-heuristic is used to solve many real life problems within a reasonable amount of time which are hard to solve otherwise. These problems exist in numerous areas of science, engineering, economics, and business. Meta-heuristics efficiently explore the search space. In addition to fast and robust solutions, meta-heuristics are simple to design and implement. In some cases, the meta-heuristic algorithms can be

designed in such a way that they completely avoid to trap into the local minima. Meta-heuristics are flexible algorithms and can be applied to a variety of problems.

Most of the widely used meta-heuristic algorithms such as immune system, genetic algorithm, annealing process, ant-colony, particle swarm, bee colony, wasp swarm, etc., are inspired from natural metaphors. Generally, these algorithms are implemented in their base form, but that is not always the case to obtain the best solution. Different strategies such as hybridization of algorithms, the addition of some problem specific constraints, etc., have also been tested to improve the results. In some of the attempts, these strategies proved better than the base algorithms [79–83].

Optimization has a wide domain of applications in different fields of engineering design, business, logistics, networking, finance, transportation, bioinformatics and computational biology, data mining, etc. Researchers are facing new and more complex problems pertaining to optimization in each field. Therefore, scientists are working to improve meta-heuristic algorithms to solve these complex problems. However, in most cases, solving a complex problem is time consuming and expensive. Therefore, heuristic/meta-heuristic algorithms are usually applied at the early stages to solve the problems which make it considerably easier and cheaper to fix the problem.

Meta-heuristic, however, requires a certain level of knowledge and experience for their proper implementation. Sometimes, it is difficult and expensive to find a domain expert. Also, the evaluator of results and algorithm should be aware of technical limitations on the design of the algorithm. Heuristic algorithms are loosely structured and may not always be able to produce the best results. Despite these disadvantages, their evaluations play an important role in solving a large number of optimization problems and if implemented properly, they can give efficient results.

2.4 Related Work

Optimization techniques for PSP mainly employ comparative modeling, physics-based methods and some alternative physics-based models such as HP Model [25] and Miyazawa-Jernigan (MJ) matrix model [84, 85]. Comparative modeling compares the primary protein structure with those available in the databases. It includes homology and threading. Homology modeling predicts the structure of target protein by comparing its sequence with the similar sequences of amino acids from the database. It has been observed that more than 30% correlation in sequences is better for homology structure prediction and is helpful for generating hypotheses about the protein's function. Some of the important tools used for homology modeling are FASTA [86, 87], NCBI BLAST [88], CABS, MODELLER and EasyModeller [89], SWISS-MODEL [90], FoldX [91], HHpred [92], Prime, Yasara, etc.

Threading, however, is more effective than homology and is employed when homology fails to predict the protein sequence. Homology modeling uses the sequence information for the alignment, whereas, protein threading extracts both the structure as well as sequence information to recognize the folds. The predicted structure is strongly dependent on the size and details of the library chosen for threading. Some important tools available for threading are LOMETS [93], TASSER [94] and I-TASSER [95, 96], Phyre and Phyre2 [97], HHpred [92], SUPERFAMILY, RaptorX [98], MUSTER [99], etc. LOMETS [93] which stands for Local Threading Meta-Server, is developed for quick and automated redactions of tertiary protein structures and spatial constraints. The data obtained from LOMETS can be used to guide the ab-initio procedures such as TASSER [94] for further PSP.

If comparative methods are unable to predict the structure, models have to be constructed from scratch. Physics-based modeling, ab-initio modeling, free modeling, or de novo modeling are some of the procedure used in this case. The term ab-initio modeling is generally used for quantum mechanical modeling. In the physics-based modeling approach, the search is conducted under the guidance of energy function. The generated conformations

are called structure decoys and the minimized energy structure among these decoys is selected as the final structure. The energy function can be physics-based or knowledge-based potentials. Physics-based functions are selected on the basis of molecular dynamics or quantum mechanical calculations, whereas in knowledge-based, the function is derived from the statistical analysis of the structures submitted to the database.

TASSER and I-TASSER [94–96] and ROSETTA [100] are the main tools available for de-novo modeling. ROSETTA [100] uses both physics as well as knowledge-based energy functions. It is a fragment-based method used to compare the protein structure fragments of sizes three to nine (amino acids) with the unrelated protein structures of similar local sequences. I-TASSER stands for Iterative Threading Assembly Refinement and is a hierarchical method for protein structure and function predictions [101]. I-TASSER was ranked as the No. 1 server for protein structure prediction in recent experiments of CASP7, CASP8, CASP9, CASP10, CASP11 [95]. The other de-novo algorithms are CABS, CABS-FOLD, ABALONE, PEP-FOLD, BHAGEERATH, Rossetta@ home, FALCON@ home [102], etc.

The two techniques mentioned above, comparative modeling and physics-based approach, have their own advantages and disadvantages. The comparative modeling is an alternative approach and does not follow the actual physiochemical process. The physics-based tools, on the other hand, require high-performance computing facilities because the calculations start from scratch and have to deal with astronomical data points. However, they are most reliable to map the realistic protein's structure. Hence, protein structure prediction using physics-based simulations are favorable only for small protein. However, their realistic approach to search for minimized energy led to the development of different alternative models. These kinds of models employ fundamental physics principles under some controlled situation, *i.e.*, restricting the optimization search within some parameters of energy, space, size, etc. HP (Hydrophobic-Polar) model [25], Miyazawa-Jernigan (MJ) matrix model [84, 85], Nuclear Condensation model, Diffusion Collision models [103], etc., are some examples of alternative models. Due to constraints on parameters, these type of models are computationally fast but are not considered successful as the gap remains between the prediction

structure and the actual native structure, however, these studies are helpful to build the first level approximate structure.

Out of these models, as discussed in the previous chapter, HP model [25] is the most effective, and popularly used. Since the problem of finding the minimized structure even with the HP model is proved to be NP-complete, therefore a number of heuristic/meta-heuristic optimization algorithms are employed [82,82,104–112,112–120]. Some hybrid optimization procedures are also present in the literature [115, 116, 121–123]. Most of these algorithms limit the configurational space to lattice points and use the contact-based energy function to calculate the energy.

Some of the meta-heuristic algorithms such as evolutionary algorithms, ant colony algorithms, simulated annealing, particle swarm optimization, etc., are frequently used in literature. PSP using HP model is widely studied by employing genetic algorithm independently or in hybrid with other algorithms. The genetic algorithm is inspired by Darwin's theory of evolution. The key idea behind this algorithm is that evolution is an optimization process and optimization procedure follows the random evolutionary processes like mutation and crossover. Unger and Moult [104], employed genetic algorithm in their optimization search for HP model in 2D cubic lattice space. Subsequently, several 2D as well as 3D on-lattice HP model reports were published using genetic algorithm optimization search [82, 105–118].

Halm studied the effect of different GA's parameters in case of 2D cubic lattice [124]. Pull moves in protein structure prediction was described by Lesh *et al.* [113] and Berenboym and Avigal [112] employed them in the genetic algorithm. In different studies by Pedersen and Moult, the genetic algorithm was revised against three peptides. They also explored the application of GA for PSP study with a full atomic representation using the solvation model and free energy function [111, 125, 126]. A 2D and 3D guided search in GA was performed by Haque *et al.* [127,128]. Lin and Su used a hybrid genetic-based particle swarm optimization (PSO) [80]. Yi-Yao Huang *et al.* [129] applied GA to predict the target proteins from known primary sequence and secondary structure elements. Twin removal strategy is also employed to the simplified on-lattice HP model [119,120] and some modification using

quasi-random sequences for creating the initial population has also been reported [114]. Shatabda *et al.* [130] applied the genetic algorithms in case of FCC lattice and in local search approaches [123, 131]. The group also reported the local search using embedded GA [130]. A hybrid study of Hill-climbing and genetic algorithm (HHGA) based on elite-based reproduction strategy in case of 2D triangular lattice [121] is studied by Su *et al.*

The other algorithms use different search method for PSP such as Tabu search [132], Tabu Search with Hill Climbing [115], Tabu Search with GA [116] and Tabu-based Stochastic Local Search [122, 123]. Tabu search uses the local search approaches for optimized solutions which consider a potential solution and searches for an improved solution from immediate neighbors. Zhang *et al.* [82] applied the combined tabu search with crossover and mutation operators. Mansour *et al.* [118] applied the particle swarm optimization (PSO) to PSP and Kondov and Berlich [117] studied the PSP employing PSO with distributed parallel programming. Ant colony optimization (ACO) [133] is another meta-heuristic method to study PSP. It is a kind of swarm intelligence approach inspired by foraging of ants. The ACO algorithm for protein folding was first applied by Shmygelska and Hernandez [134]. Islam *et al.* worked on memetic algorithms [135–137] and also employed niching technique on clustered architecture [138, 139].

Tsay and Su [140] proposed a combined method which includes local search, lattice rotation for crossover, k-site move for mutation, and generalized pull move in case of FCC. Recently, Mahmood *et al.* [141] applied the GA in cases of FCC lattice for protein structure prediction. Along with GA, they applied the random walk strategy to recover from stagnation. Kern and Lio [142] calculated the score based on the conformation generated by the genetic algorithm by taking the case of residue's positions w.r.t the hydrophobic core on HP, HPNX and hHPNX lattice models. In another hybrid approach, Ullah *et al.* [143] proposed a two-stage optimization by combining constraint programming using CPSP and simulated annealing.

The optimization under HP model within the square, triangular, and diamond lattices was tested by Krasnogor *et al.* [144]. They further applied fuzzy-logic [145]. Several other

optimization algorithms such Monte-Carlo simulation [146], simulated annealing [147], Immune Algorithms [148], firefly algorithm [2] and constraint programming [149, 150], are also reported by different researchers.

Progressive search algorithms are the type of chain growth algorithms. The different chain algorithms are categorized as Roenbluth methods. These methods apply the self-avoiding walk strategy by adding a single monomer at a time. The core directed chain growth strategy and Pruning Enrichment Rosenbluth Method (PERM) are also used to search for the native protein structure. A new PERM is proposed and applied on lattice heteropolymers to search the minimized energy state [151]. The new improvement in PERM shows that it outperforms when compared with the previous versions of the algorithm.

Out of these different optimization algorithms, very few have been utilized for development as application tools and the developed ones are mainly focused on constraint-based approaches. One such tool and the web server is CPSP, which stands for Constraint-based Protein Structure Prediction (CPSP), was introduced by Backofen and Will [152]. This approach is based on the hydrophobic core construction and follows three steps, *i.e.*, bounding, core construction, and threading. COLA, which stands for Constraint Solver for Lattices (COLA), is another such application which is also based on constraint programming. LATFold is a Monte-Carlo simulation tool based on metropolis criterion and energy Landscape library. The pull moves and pivot moves procedure is also utilized in its simulation. Dubey *et al.* also suggested an algorithm for HP model calculations and utilized the same for developing a free tool [153], however, this tool only does optimization for square and triangular lattices and no three dimension structure prediction is available. The suggested algorithm has also not reached the optimal H-H contacts claimed by other authors [139, 154–156]. However, we have not encountered any application where we can use and test different meta-heuristic algorithms to predict optimized proteins structure using HP-model.

2.5 Summary

In this chapter, we explained the complexity involved in PSP and importance of PSP, HP model and meta-heuristic algorithms and literature review about PSP tools/algorithms. The study gives an idea to build a comprehensive meta-heuristic optimization tool for simplified proteins structure prediction based on the HP model. In fact, the optimized structure obtained from HP model is far from the actual protein's structure, however, resultant structure can further be used to get fast and realistic optimized structure when employed along with ab-initio methods. In recent studies, it has been suggested that a real space output can be obtained by hierarchically transfer of the on-lattice fold (obtained from simple model) to off-lattice one [119, 157].

A number of attempts have already been made for HP model-based PSP, especially, using the base as well as hybrid heuristic/meta-heuristic algorithms. The wide range of heuristic/ meta-heuristic algorithms, discussed in the literature review, are employed because they have many advantages over traditional mathematical techniques. The easy, fast and cheap (user-based) implementation of the heuristic/ meta-heuristic algorithms have increased their usability in solving PSP and other complex problems. Also, the literature review about PSP tools revealed that most of the PSP developed tools are based on comparative modeling. Thus from the best of our knowledge, except for some single algorithm-based applications, we have not found any other application available for testing and analyzing basic and/or hybrid algorithmic procedures for HP model-based optimization study of PSP. Therefore, we designed a comprehensive tool comprising automation and visualization facilities to test and analyze some of the optimized meta-heuristic procedures in the context of PSP.

The automation and visualization tools are important because they allow higher productivity. Graphical user interfaces allow users to interact with computers using a mouse and other input tools. This allows the user to interact more freely with how they display information, images, and other attributes. In the same context, the graphical user interface,

automation characteristic provide great accessibility and the user does not need to remember the procedure to solve the problem. Similarly, the graphical user interface provides the facility to input parameters and the chances of mistakes are possibly zero. A less skilled person can also use these types of tools with minor knowledge about the problem and solution of the procedure.

Furthermore, a careful analysis of the reports suggests that a number of meta-heuristic algorithms for PSP are hybridized [79–83, 158] to improve the results, however, hybridization is limited to two algorithms in a particular manner. This also evokes the possibility of further improvement in results when the hybridization among algorithms can be done by mixing them in other possible ways or using more than two algorithm's steps. Some features such as pull moves, twin removal, mirror symmetry, local optimization, etc. which are suggested for particular algorithms, can also play an important role to improve the minimized energy state when employed with other algorithms. Therefore, we have also included hybridization and some constraints' applicability options in our software framework. Thus, the software can work for base algorithms as well as design a demand-based optimization procedure. The next chapter talks about the implemented meta-heuristic algorithms and optional constraints.

Chapter 3

Meta-Heuristic Algorithms

Proteins are long molecules that consist of hundreds to thousands of amino acids' residues. The search space to optimize these big atomic structures of a high degree of freedom is very large. Therefore, Protein Structure Prediction (PSP), optimized protein's structure, is computationally very expensive and alternative approaches are used. In our work, we restrict the search space by the use of cubic lattice to obtain the simplified structure. The use of meta-heuristic algorithms further speeds up the optimization process. The following discussion includes the three-dimensional design of the cubic lattice and the five implemented meta-heuristic algorithms used for HP model-based structure optimization. Some optional characteristics introduced in the developed tool are also discussed at the end of the chapter.

3.1 HP Model Using Cubic Lattice

3-dimensional cubic lattice is simple and widely used in literature for HP model based structure prediction. The structure prediction using 2-dimensional representation have also been reported in the literature. However, in the present tool we preferred to use 3-dimensional representation because it is more realistic and hence, constructed structures can easily be compared with the actual structures. In lattice based HP model residues are considered as beads. The locations of the residues are restricted to the lattice sites and bonds between

residues are represented using sides of the unit cell. In cubic lattice, a lattice site can connect with other by one out of six possible sides, *i.e., in (i), out (o), up (u), down (d), left (l) and right (r)* (shown in Figure 3.1). Thus, the structure representation, instead of using coordinates of residues, can also be represented in terms of these six possible directions.

In our present implementation, the first residue is placed at the origin (0,0,0) and others are specified by one out-of six directions which connect them with their previous residues in the sequence. The bonding must follow self-avoiding criteria, *i.e.*, no two resides should overlap each others' position. Thus, there are N-1 directions for N residues, and the structure of a protein is represented as a sequence of these directions. A structure with directions *"lilouiildrruooulirriiuoldi"* for HP sequence ("HHPPHHHPPPHHHHHHPHPHHHHPPHH") of length 27 is illustrated in Figure 3.2.

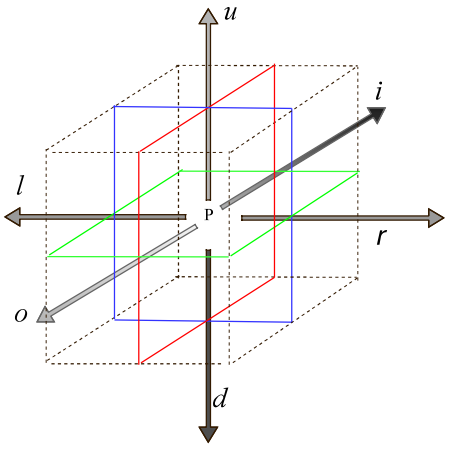

Figure 3.1: Six possible directions in cubic lattice.

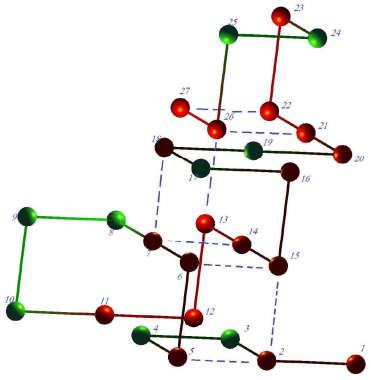

Figure 3.2: Structure for the direction list *"lilouiildrruooulirriiuoldi"*.

Structure optimization is obtained by minimizing the energy of the structure. However, in our present work of HP model-based optimization, we try to maximize the fitness score which is just the negative of the energy value. Fitness score of the structure measured as the number of non-bonded hydrophobic-hydrophobic contacts of distances equal to the unit length and calculated as

$$fitness\ \ score(fs) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}e_{ij} \tag{3.1}$$

and the energy is given by

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}e_{ij} \tag{3.2}$$

Here c_{ij} equals to 1, if i and j are non-bonded and consecutive (neighbors), otherwise, its value is 0. Also, e_{ij} is equal to 1 if both i^{th} and j^{th} residues are hydrophobic and 0 in all other cases. For example, the fitness score of the structure given in Figure 3.2 is equal to 8 and is calculated as $fs = (0+2+0+0+1+1+2+0+0+0+0+0+1+1+2+0+0+1+0+0+1+1+0+0+0+2+1)/2 = 16/2 = 8$. The eight hydrophobic-hydrophobic contacts are also shown in the figure using dotted lines.

3.2 Meta-Heuristic Algorithms

The tool presented in this thesis incorporated with five meta-heuristic algorithms. All these are population based meta-heuristic algorithms which initially creates a set of user-defined (*Pop*) protein's structures called the initial population (*popList*). The term individual is used to indicate the structure of a protein (sequence of directions or direction list) as suggested for genetic algorithm [159] and set of individuals are called as individuals' population or simply population. The basic criteria for optimization in all the implemented algorithms are almost similar. In the beginning, they all select some individuals from the existing population, perform their operations on selected individuals to create new individuals and finally, update the population with higher scored individuals. In order to search for optimized structure, the process repeated again and again for a fixed number of cycles. These number of cycles are called the number of generations or simply generations (*Gens*). Thus, during each generation, new structures are created and their scores are compared with the previous population. If fitness scores of the newly generated structures are higher, they replace the lower scored individuals from the population. This may create a better population during each generation. The process is repeated for all generations, or until no improvement occurs for a certain number of generation (*stopGen*). It is clear that the basic optimization approach in all implemented algorithms is similar but they all have distinct operations. The following

discussion talks, specifically, about each algorithm in detail:

3.2.1 Genetic Algorithm

Genetic Algorithm (GA) is the outcome of John Holland's research work during 1960 but it becomes popular in the '90s. The algorithm is an alternative approach to solve large and complex problems. The basic GA is based on the mechanism of natural selection and natural reproduction criteria. The pseudo-code for GA is given in **Algorithm** 1.

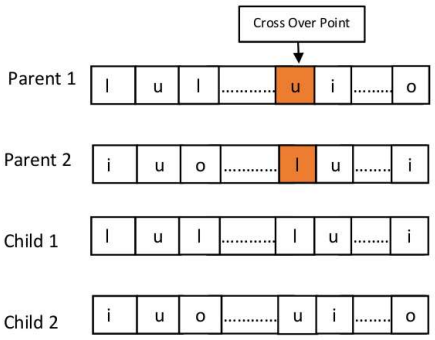

Figure 3.3: One point Crossover process in GA.

The algorithm starts with the creation of an initial population (*popList*). During each generation, GA performs two genetic operations, known as Crossover and Mutation, on the selected number of individuals called parents (*parentList*). The newly obtained structures are called offsprings (*offSpringList*). The Crossover operation is similar to the genetic crossover where different chromosomes mix together, exchange their genes, and give birth to new offspring. In HP model, Crossover operation exchanges the subparts of the parents between each other (shown in Figure 3.3) and produces the same number of offsprings. Mutation operation also similar to a genetic mutation. Here, it alters the direction of the individual at a random position (Figure 3.4).

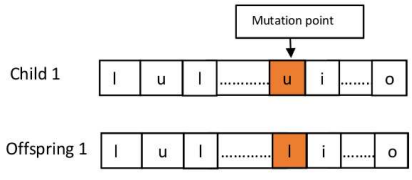

Figure 3.4: One point Mutation in GA.

It might happen that the offsprings obtained after these operations, do not satisfy the self-avoiding condition. Therefore, a procedure is used to remove the overlapped positions of the residues. The process is called offspring adjustment (AdjustOffSpring). If these adjusted offsprings are superior in fitness scores than the individuals in the population they replaced

input : $HPSeq, Pop, Gens, CrPt, MuPt, MuStrt, MuEnd, PrNo, StopGen$
output: $Optimized\ Individual(directionList)$

popList(0) $\leftarrow$ get Pop no. of individuals
for $i \leftarrow 1$ **to** $Gens$ **do**
  $parentsList \leftarrow$ SelectParents($popList(i-1), PrNo$)
  $offSpringList \leftarrow$ Crossover($parentsList, CrPt$)
  $offSpringList \leftarrow$ Mutation($offSpringList, MuPt, MuStrt, MuEnd$)
  $offSpringList \leftarrow$ AdjustOffSpring($offSpringList$)
  $popList(i) \leftarrow$ Replacement($offSpringList, popList(i-1)$)
end
return $BestIndividual(popList(Gens))$

Algorithm 1: Genetic Algorithm's pseudo code.

those individuals (Replacement). The process runs for a given number of generations and during each generation, attempts are made to form a better population.

In traditional GA, only two parent individuals are selected to perform genetic operations. However, in the present tool, the user can choose the number of parents (*PrNo*). Therefore, instead of two individuals, a list of parent individuals is selected. The Crossover operation on this list replaces the sub-part of each parent individual with the next individual in the parents' list. The operation is performed once in a cyclic manner and sub-part of the last individual goes with the first individual in the parents' list. Options are also available for the use of multi-point Crossover and Mutation operations, *i.e.*, Crossover is performed for user-specified (*CrPt*) numbers at randomly selected positions and Mutation randomly alter the user specified (*MuPt*) directions at random positions in each parent. The Mutation operation can also be chosen on a rate basis. The rate of mutation during each generation depends on the generation number. The user needs to enter the initial (*MuStrt*) and final (*MuEnd*) values of mutation rate. The rate of mutation for each generation is calculated as

$$rate = (MuStrt - MuEnd * f) * 100 \tag{3.3}$$

Here $f = (generation\ number)/(total\ \ generations)$ and $MuStrt > MuEnd$.

3.2.2 Parallel Genetic Algorithms

We have implemented two types of parallel models of genetic algorithms. The first type is similar to the parallel Island model (PGA-IM) discussed in [160]. The pseudo code of the PGA-IM is shown in **Algorithm** 2. The optimization search in this method is done by dividing the population into subpopulations called Islands. The number of islands chosen is based on the number of threads (*Threads*) selected by the user. The genetic algorithm is applied, separately, on each island for a user-specified number of cycles per generation (*CylPerTh*). After completion of these cycles populations of all Islands merged together as the total population. This process is repeated for user-defined number of generation, *i.e.*, new islands again formed for next generation and run again for cycles per generation (shown in Figure 3.5 (a)). There are three island formation methods (*IslandMethod*) available in the present tool, *i.e.*, "Random", "Close" and "Far". The "Random" method divides the total population just using the random individuals, whereas, "Close" and "Far" island methods form the islands based on the fitness scores of the individuals. In the former selection procedure the individuals are chosen together which have similar/close fitness scores, and in the latter case, the population of each island is created from individuals with diverse fitness scores.

The second parallel genetic algorithm is similar to the parallel grid model (PGA-GM) [160]. In this parallel approach, each thread separately performs genetic operations on the user-specified number of parent individuals. The process of parent selection in this approach randomly selects the individuals and their neighbors[1]. Finally, the adjusted offsprings from all threads get together and replace the inferior individuals with superior offsprings. The process is repeated for all generations, or until stropping conditions are met. The individual with the highest fitness score from the final population is taken as the minimized energy structure. Figure 3.5 (b) illustrates the calculation process of this method. The pseudo code is given in **Algorithm** 3.

[1]Since the calculation process uses the population as a sorted list of individuals in decreasing order of their fitness scores, the neighbor(s) of an individual are selected from both sides of that individual from the sorted list and might have same or close value of fitness scores.

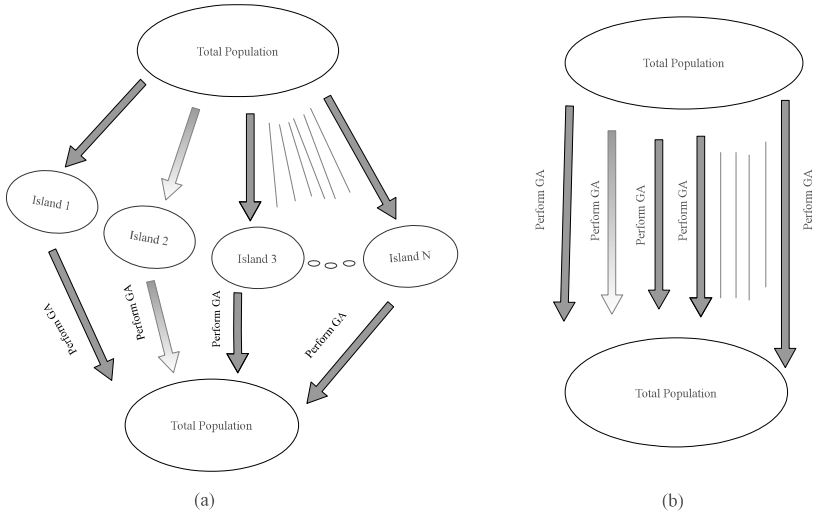

Figure 3.5: Calculation process for single generation in case of (a) PGA-IM and (b) PGA-GM.

input : $HPSeq, Pop, Gens, CrPt, MuPt, MuStrt, MuEnd, CylPerTh, IslandMethd, Threads, StopGen$
output: $Optimized\ Individual(directionList)$

popList(0) $\leftarrow$ get Pop no. of individuals
for $i \leftarrow 1$ **to** $Gens$ **do**
  $islandSizeList \leftarrow$ GetIslandsList($popList(i-1), Threads$)
  for $th \leftarrow 1\ to\ Threads$ **do**
    $islandPop(th) \leftarrow$ SelectPop($IslandSize(th), IslandMethd), popList(i-1)$)
    $islandPop(th) \leftarrow$ GA($HPSeq, islandPop(th), CylPerTh, CrPt, MuPt, MuStrt, MuEnd, PrNo, StopGen$)
  end
  Synchronize
  $popList(i) \leftarrow popList(i) + islandPop(th)$
  for $th \leftarrow 1\ to\ Threads$ **do**
    Join threads
  end
  $popList(i) \leftarrow$ Arrange($popList(i)$)
end
return $BestIndividual(popList(Gens))$

Algorithm 2: Pseudo code of PGA-IM Algorithm.

input : $HPSeq, Pop, Gens, CrPt, MuPt, MuStrt, MuEnd, Threads, StopGen$
output: $Optimized\ Individual(directionList)$

popList(0) $\leftarrow$ get Pop no. of individuals
for $i \leftarrow 1$ **to** $Gens$ **do**
  for $th \leftarrow 1\ to\ Threads$ **do**
    $offSpringList(th) \leftarrow$ GA($HPSeq, popList(i-1), 1, CrPt, MuPt, MuStrt, MuEnd, PrNo, StopGen$)
  end
  Synchronize
  $popList(i) \leftarrow$ Replacement($offSpringList(th), popList(i-1)$)
  for $th \leftarrow 1\ to\ Threads$ **do**
    Join threads
  end
  $popList(i) \leftarrow$ Arrange($popList(i)$)
end
return $BestIndividual(popList(Gens))$

Algorithm 3: Pseudo code of PGA-GM Algorithm.

3.2.3 Immune Algorithm

Immune Algorithms are based on the process of the biological immune system. Artificial Immune Systems (AIS) are adaptive systems which are a sub-field of Computational Intelligence inspired by theoretical immunology. Artificial Immune systems can be divided into three sub-categories, *i.e.*, clonal selection, negative selection, and immune network algorithms. These techniques are used in the field of pattern recognition, optimization, clustering, and other machine learning domains. In the present work of HP model based optimization, we have implemented the clonal selection Algorithm suggested by Vincenzo and Giuseppe [148].

The theoretical basis of clonal selection was first proposed by Burnet [161]. The theory explains the response of adaptive immune system on antigens[2] detection. In our physical system, the primary function of the immune system is to protect our bodies from pathogens, *i.e.*, viruses, bacteria and other parasites, which can disturb our normal biological system. When

[2]Antigens are parts of the pathogen that tell your body it's being attacked. Pathogens are the germs that make you sick and antigens are other foreign substance that induces an immune response in the body. The immune system uses antigens to detect the pathogens.

input : $HPSeq, Pop, Gens, CloSize, c, \tau_B, StopGen$
output: $Optimized\ Individual(directionList)$

popList(0) $\leftarrow$ get Pop no. of individuals
for $i \leftarrow 1$ **to** $Gens$ **do**
  $P^{clo} \leftarrow$ StatiCloning$(popList(i-1), CloSize)$
  $P^{hyp} \leftarrow$ HyperMutation(P^{clo}, c, ln)
  $P^{macro} \leftarrow$ HyperMacroMutation(P^{clo})
  $P^{tot} \leftarrow P^{clo} + P^{hyp} + P^{macro}$
  $P^{tot} \leftarrow$ RemoveDuplicate(P^{tot})
  $P^{age} \leftarrow$ AgingOperator(P^{tot}, τ_B)
  $popList(i) \leftarrow (\lambda + \mu)$ Selection(P^{age}, Pop)
  $popList(i) \leftarrow$ Arrange$(popList(i))$
end
return $BestIndividual(popList(Gens))$

Algorithm 4: Immune Algorithm's pseudo code

the immune system recognizes antigen, it produces antibodies against pathogens. The immune system, actually, composed of diverse sets of cells and molecules that work together to maintain homeostatic state. The idea behind the clonal selection principle reveals that those cells that are capable of recognizing an antigen will increase rapidly in numbers while other cells are selected against. Clonal selection works on both B and T cells. When the antibodies of the B cells bind with an antigen, clones of B cells are produced and undergo somatic hyper-mutation, and the B cells get differentiated into plasma or memory cells. Plasma cells produce antigen-specific antibodies that will work against pathogens. We have used Static Clonal, Hyper-mutation, Hyper-macro-mutation and Aging operations for simplified PSP as suggested by Cutello *et al.* [148]. The pseudo code in **Algorithm** 4 describe these operations in our implemented algorithm and different algorithmic steps are explained as below:

1. Initial Population (P^t or $popList$): The step is similar to GA where the initial population is created by generating random individuals.

2. Static Cloning (P^{col}): During this operation user defined number of clones ($CloSize$) or

duplicate (dup) of each individual are created and produce a clonal population (P^{col}), *i.e.*,

$$P^{col} = P^t \times CloSize \tag{3.4}$$

3. Hyper-mutation: Hyper-mutation perform two operations on P^{col}, *i.e.*,

   (a) Inversely Proportional Hyper-mutation (P^{hyp}): The mutation operation is given by

   $$M_i(f(x)) = \begin{cases} ((1 - \frac{E^*}{-1}) \times \beta) + \beta), & \text{if } f(x) = 0 \\ ((1 - \frac{E^*}{f(x)}) \times \beta) + \beta), & \text{if } f(x) > 0 \end{cases} \tag{3.5}$$

   Here $f(x) = -E$ is the energy of the receptor (individual) x ε P^t and E^* the current best fitness value. $\beta = c \times l$, where c is constant and l is the length of the protein (number of residues).

   (b) Hyper-macro-mutation (P^{macro}): A random mutation is performed between two positions i and j such that $(i+1) \leq j \leq l$ while maintaining the self-avoiding property.

4. Aging Operation: In this process, the receptors (individuals) which are surviving more than a number of generations (τ_B) are removed from P^{col}, P^{hyp}, and P^{macro}, irrespective of their fitness scores.

5. $(\mu+\lambda)$-selection with Birth Phase: After aging operation, the number of receptors can be different than the initial population (P^t). If they are less than the initial population, the required number of individuals are created and merged, and if greater, the lower scored individuals are removed to keep the population same as initial population.

3.2.4 Ant Colony Optimization

We have also employed the ant colony optimization algorithm as suggested by Shmygelska and Hoos [162]. Ant Colony Optimization (ACO) uses the basic concept of ant stigmergy,

i.e., using the environment as a means of communication without direct communication between participants. The iterative process of the algorithm follows three steps, *i.e.*, Construction, Local Search, and Pheromone. The construction phase is similar to the creation of the initial population in GA, but here the individuals are generated using a chain growth procedure. The chain construction starts from a random position and grows in both directions so that the ratio of the residues in each side is roughly equal. The direction (d) for each residue is selected based on the probability ($\rho_{i,d}$) of the residue at i^{th} position, which is calculated as

$$\rho_{i,d} = \frac{[\tau_{i,d}]^{\alpha}[\eta_{i,d}]^{\beta}}{\sum_{e=1}^{5}[\tau_{i,e}]^{\alpha}[\eta_{i,e}]^{\beta}} \tag{3.6}$$

Here $\tau_{i,d}$ is the pheromone value, which is updated at each iteration, however, the initial values of $\tau_{i,d}$ are the same for all individuals. The quantity $\eta_{i,e}$ is called the heuristic function and is equal to $e^{-\gamma.h_{i,d}}$. γ is inverse temperature parameter and $h_{i,d}$ is the number of H-H contacts that can be made by placing i^{th} residue in direction d. During the construction process attrition[3] might happen at certain steps. In Shmygelska and Hoos's procedure, attrition is overcome by starting the folding process again with half of the length of the current sequence. A different starting point direction is taken from what it had been when attrition occurred. In our implementation, we try to preserve this higher probability structure by testing different directions starting from attrition point in the reverse direction. If any direction works at any point, the sequence will start moving forward. If the normal folding procedure does not start after testing a certain number of attempts at one point, the Shmygelska and Hoos's procedure is followed.

The next step is the local search, which is similar to the mutation step. Since the local search step suggested by Shmygelska and Hoos [162] is time consuming, we have considered this similar to GA mutation step, *i.e.*, create a local list of individuals (*localList*) by randomly changing one direction in of the selected individuals. This step is generally performed on half of the population, however, in the present software, the user can choose the number of

[3]The sequence cannot run folding due to overlapping of residues positions with each other.

input : $Pop, Gen, AASeq, \alpha, \beta, \gamma, \eta, \delta, LocIndNo$
output: $Optimized\ Individual(directionList)$

popList(0) $\leftarrow$ get Pop no. of individuals using eq(3.5)
for $i \leftarrow 1$ **to** $Gens$ **do**
  $popList(i-1) \leftarrow constructPheremone(popList(i-1), \alpha, \beta, \gamma, \eta, \delta, \Delta\tau(i-1), popList(i-1))$
  $localListt \leftarrow localSearch(popList(i-1), LocIndNo)$
  $popList(i) \leftarrow update(localList, \rho, \Delta\tau(i-1))$
end
return $BestIndividual(popList(Gens))$

Algorithm 5: Pseudo code for ACO Algorithm.

individuals for local search. The last step updates the $\tau_{i,d}$ value using the following equations

$$\tau_{i,d} = \rho.\tau_{i,d} \tag{3.7}$$

$$\tau_{i,d} = \tau_{i,d} + \delta_{i,d} \tag{3.8}$$

ρ lies between 0 and 1, and it is a parameter which moves the information gathered forward in the last step. $\delta_{i,d} = E/E^*$, called the relative solution quantity, is the ratio of energy during the last cycle (E) to the possible minimum energy (E^*) of the protein (usually taken from previously reported sequences). The pseudo code for the ACO algorithm is shown in **Algorithm** 5.

3.2.5 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm was first used by Kennedy and Eberhart to study social and cognitive behavior. The algorithm is based on the flocking behavior of birds. Each participant has their own position and speed to move in the search space. We have not implemented the algorithm in its exact form, but used the concept of global best and the best positions individual achieved by every participant. In each generation, every direction of individuals is adjusted on the basis of corresponding directions of the global best individual and the best-scored structure (local best) achieved by the individual

under consideration. The PSO converges very fast, but it can get stuck in a local minima (or maxima). To avoid this situation we have used move back strategy, *i.e.*, the global best individual is replaced with the lowest scored individual for a certain number of generations. The pseudo code of the algorithm is written in **Algorithm** 6.

The first step in PSO, similar to other algorithms, is to generate random solutions (individuals) called particles which construct the initial population. The algorithm selects the individuals with highest and lowest scores as *gBest* and *gLowest*, and marks initially each individual as local best, *i.e.*, *pBest*. The algorithm selects two random positions, *r1 and r2* such that $1 \leq r1 < N-1$ and $1 < r2 \leq N-1$,, of each individual, and alter the directions between these two positions to increase the fitness score. These directions can be random or the same as in the *pBest* or *gBest* individuals at corresponding positions. During different generations, the algorithm keeps track of the best scored individuals of each particle *pBest* and also keeps updating the *gBest*, *i.e.*, global best solutions. If the global best score does not increase for a given number of generations (*mF*), *gBest* is replaced by *gLowest* for a certain number of generations (*mB*). This introduces diversity in the structures of the individuals.

3.3 Optional Characteristics

Along with the implementation of the above algorithms, the code also consists of some additional steps, constraints or features which can be applied during optimization search calculations. In the current software, we have introduced options for Secondary Structure, Twin Removal, Mirror Symmetry, and Same Coordinate Search.

The concept of Secondary Structure is taken as suggested by Bui and Sundarraj [159]. They have used it for the 2-dimensional case, however, we further extended the idea to three-dimensional. The secondary structures (α-helix, β-strand or β-sheet, and turns or loops that connect the helices and strands) are the backbone conformations, and along with their side chains, they are packed tightly to form the three-dimensional tertiary structure. Bui and Sundarraj used that idea, and take the parts of protein structure which consist of only

input : $HPSeq, Pop, Gens, mF, mB, StopGen$
output: $Optimized\ Individual(directionList)$

popList ← get Pop no. of individuals
$gBest$ ←individual with highest score
$gBestScr$ ←Score($gBestInd$)
$pBestList$ ←popList

for $i \leftarrow 1$ **to** $Gens$ **do**
  select random nos $r1$ and $r2$, $r1 < r2$
  if $(i! = gBest\ and\ i! = gLowest)$ **then**
    for $j \leftarrow r1$ **to** $r2$ **do**
      $I \leftarrow$ popList.get(i)
      $pBest \leftarrow pBestList.get(i)$
      $pBestScr$ ←Score($pBestInd$)
      $rI \leftarrow$ Replace I's j^{th} direction with random direction
      $pI \leftarrow$ Replace I's j^{th} direction with j^{th} $pBest$'s direction
      if *(moveBack=false)* **then**
        $gI \leftarrow$ Replace I's j^{th} direction with j^{th} $gBest$'s direction
      end
      else
        $gI \leftarrow$ Replace I's j^{th} direction with j^{th} $gLowest$'s direction
      end
      I←$MaxScoredValid(I, rI, pI, gI)$
      scr←Score(I)
      if $scr > pBestScr$ **then**
        $pBest \leftarrow I$
        $pBestScr \leftarrow scr$
        set $pBest$ to $pBestList$
      end
      if $scr > gBestScr$ **then**
        $gBest \leftarrow I$
        $gBestScr \leftarrow scr$
      end
    end
    if *moveBack=false and gBestScr is not improved for last mF generations* **then**
      moveBack=true for next mB generations
    end
  end
end
return $BestIndividual(popList(Gens))$

Algorithm 6: Pseudo code for PSO Algorithm with Move Back strategy.

hydrophobic residues as secondary structures. A separate calculation was suggested for these secondary structures. We have implemented the same idea and identify the hydrophobic sequences in the main sequence with user-defined minimum and maximum sizes. We call these sequences as HH sequences. Separate calculations are performed for these sequences and no operation is performed in HH regions during main operations of the algorithm. In case, if any HH sequence violates the self-avoiding criteria then another HH sequence from the optimized HH population is tested. If no HH sequence passes for a certain number of times, the calculations are performed without considering HH sequences. In the present version of the tool, the HH sequence is applicable only for GA, ACO, and PGAs.

Same coordinate Optimization (SCO) is similar to the local optimization introduced by Bui and Sundarraj [159]. This property can be added as an additional step to any algorithm. This step works to increase the score of an individual by making new connections using the same set of coordinates of that individual. Bui and Sundarraj called it local optimization. They defined two possibilities in the 2-dimensional case when the connection between residues can be changed. However, in the three-dimensional cubic lattice we have implemented only one condition, *i.e.*, if we encounter a 4-point configuration such as ABCD as shown in Figure 3.6, then we can exchange parallel connections CD and EF by CE and DF, respectively, as shown. Now, if the new conformation has a higher fitness score, it can be used instead of the previous one. It is possible that the new conformation may form a loop (e.g., if AB and BD are connected in place of AB and CD) that separate the sequence into two parts. In such a situation, the algorithm

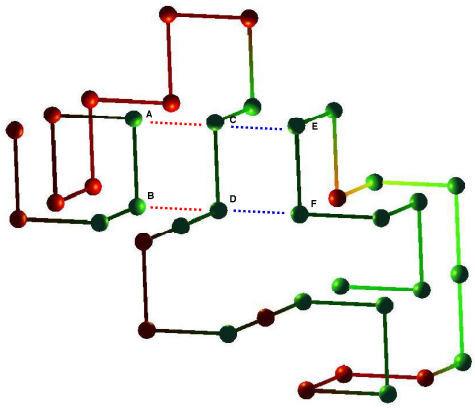

Figure 3.6: Same Coordinate Optimization; a) formation of loop when connecting AC and BD, and removing AB and CD and b) no loop formation when connecting CE and DF and removing CD and EF bonds.

searches for another edge of the loop where it can connect to the second part. If no such edge is found, the conformation is discarded. Thus, the best conformation out of all possible conformations of SCO is selected for further calculations.

Twin removal is another characteristic which has been tested by some authors for GA [119]. It is clear from the name that this feature removes the duplicated (twins) structure from the population. Thus, if this feature is selected, all the individuals in the population will have distinct structures. The property is also helpful in preventing the algorithm from getting stuck in local minima/maxima. It is mainly helpful when the whole population starts moving towards only one structure. We have also implemented the symmetry characteristic to the individuals. Symmetry is an important property in different molecules and in globular proteins. Therefore, in this initial stage of constructed software, we have only introduced Mirror Symmetry. By adding the Mirror Symmetry property to the calculations, the algorithm will allow only those individuals which have Mirror Symmetry in their structures. In future versions of the code, we will work to add other types of symmetry.

Chapter 4

Design and Implementation of Hybrid Optimization Tool for Protein Structure Prediction (HOT-PSP)

4.1 Architectural Plan

Having a software architectural plan is an important step before the development of a software. It gives the higher level structure of the system components, their relationships to each other, and to the environment. The software architecture of the developed application for simplified protein structure prediction is shown in Figure 4.1. We named the present application HOT-PSP which stands for Hybrid Optimization Tool for Protein Structure Prediction. Figure 4.1 represents the architecture of the HOT-PSP. The diagram also explains the workflow between the different components of the front-end and back-end parts. Details about these components are discussed below:

- **Graphical User Interface (GUI)** : The GUI can be divided into two parts: an Input unit and a Display unit. The Input unit takes the input structure of a protein in the form of a sequence of residues present in that protein. The sequence can be entered in the

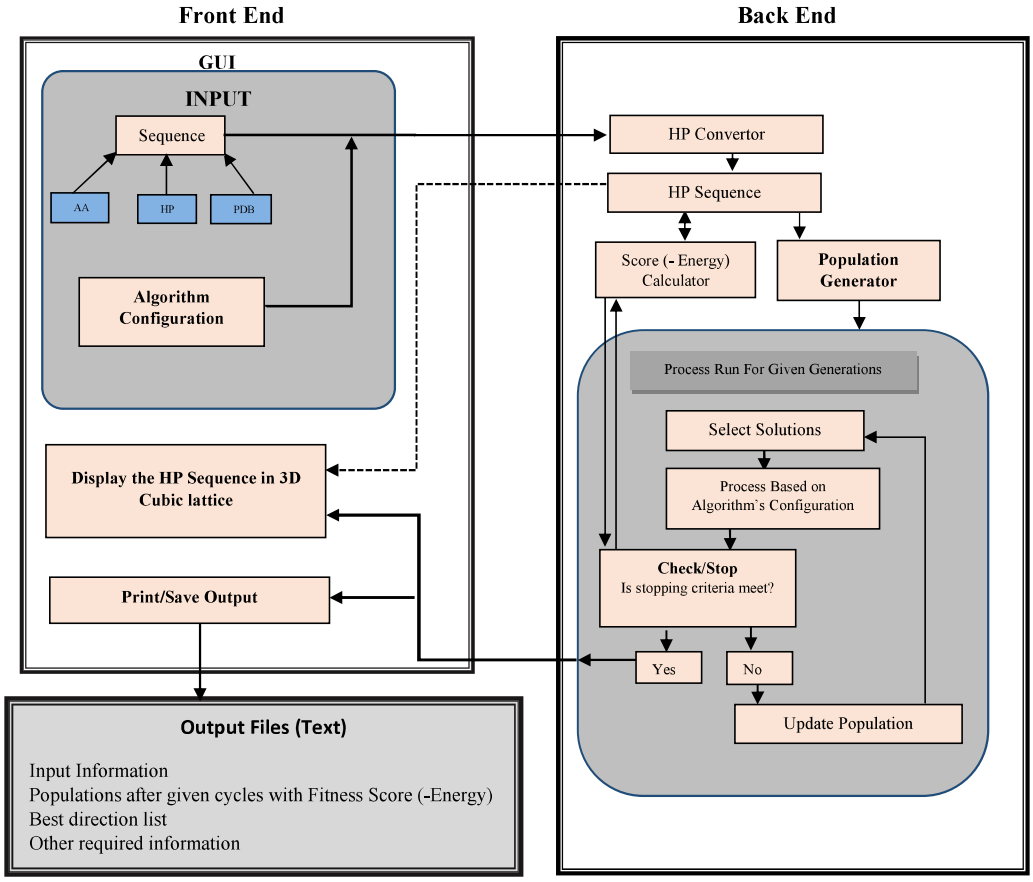

Figure 4.1: Architecture of the developed HOT-PSP tool.

form of H(h) and P(p), where letter H corresponds to hydrophobic residues and P for polar or hydrophilic residues. The sequence can be directly entered from a .pdb file. The input unit automatically takes the residues' sequence from the file and convert them into H and P forms. The three letter or one letter names of the residues can also be entered as the input sequence. The GUI also consists of input panels (Algorithmic Configuration) for setting the algorithm and other parameters which are required for calculations, structure display, and output results.

- **HP Converter** : HP model-based calculations are performed only on HP sequences. Therefore, if the input sequence is not in the form of HP sequence, HP convertor converts that into the HP sequence.

- **Score Calculator** : The Score Calculator calculates the fitness score (-energy) of the sequence under consideration based on equation 3.1. The main use of the Score Calculator is during optimization calculations, however, it can also be used to calculate the fitness score of any given sequence with given direction list.

- **Population Generator** : The Population Generator generates an initial population. The population is a set of different possible solutions (in our case structures in 3D cubic lattice) which can be generated based on the user's selected method.

- **Solution Selector** : In most of the implemented algorithms, the algorithmic operations are performed only on selected individuals. The Solution Selector selects these given number of individuals during each generation.

- **Process Solutions** : This is the main step where the selected individuals are modified based on the selected procedure. This procedure may include a single algorithm, a single step or a combination of steps from different algorithms.

- **Check/Stop** : Normal calculation process stops after the user specified number of generations, however, the calculations also stop in case the algorithm is stuck in a local minima. The **Check/Stop** method stops the calculation process if no increase in fitness score has been recorded for a certain number of generations, otherwise the next step of population update is followed.

- **Population update** : The Population Update component updates the population based on a user defined procedure, however if the individuals obtained after algorithmic operations do not meet the required conditions, the next generation cycle starts without updating the population.

- **Output Files** : This component creates the text files that record the output as well as input information. The input information includes the name of the algorithm or names of a combination of steps from different algorithms and other input parameters. The output section of the file includes the optimized structure and final population. The list of resultant individuals (population) after a given number of generations is also saved. Other optional information about the interpretation of results can also be saved into different text files (.cube and .log).

4.2 Development and Implementation

The software architecture discussed in the last section suggests the upper-level components of the software. The development of the software is completed by the joint implementation of these components. The implementation done using the Java programming language and the 3D animation view of the protein structure is displayed using JOGL (Java-based Open GL version 2). Java has many advantages such as object-oriented programming, robustness, security, ease of distributed computing, etc. Specifically, we prefer Java because of platform independent and multi-threading characteristics. Moreover, the JOGL library is capable of rendering the required animation view of the 3D structure. We divided the developed software into three main panels; an input unit, the display unit, and calculation setup unit. The following discussion highlights the input parameters/fields, and their use and purposes present in these three units.

4.2.1 Input Unit

The input unit is used to enter the residues' sequence of the protein to be optimized. The input can be entered either with .pdb file or manually using **Input Sequence** panel. The .pdb file can be selected by choosing **Open** option of **File** tab of the menu bar. This opens up a selection window to choose the .pdb file. However, the manual input can be entered using

Input Sequence panel which can be selected from the **New** option of the **File** tab. The **Input Sequence** window appears as shown in Figure 4.2.

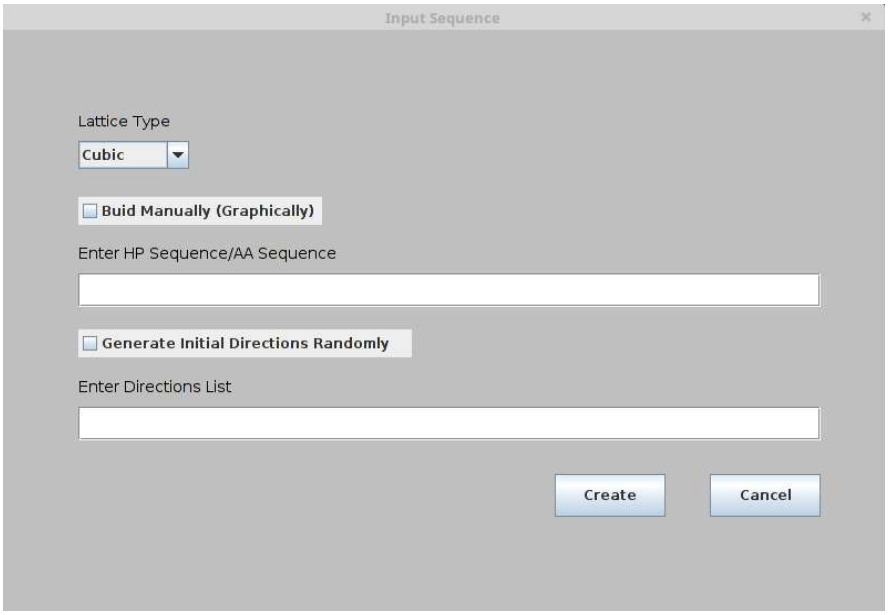

Figure 4.2: Dialog box for input sequence.

In the **Input Sequence** window the residues' sequence can be entered in the text field present under the label **Enter HP Sequence/AA Sequence**. The sequence can be entered as HP (hp) letters or using residues names. The **Create** button can be selected to take the sequence as input. After selecting the input sequence, a 3-dimensional structure displays in the display panel. If the entered sequence is not in accordance with any of the naming convention (three letters/one letter/HP(hp)), an error message is displayed to enter the correct sequence. Since the checkbox labeled with **Generate Initial Directions Randomly** is checked by default, the structure is displayed using random directions. If the user wants to enter his/her own directions (structure), the checkbox needs to be un-check. This opens an access to enter directions in the field under the label **Enter Direction List**. The **Create** button can also pop-up an error message if the directions do not follow the self-avoiding criteria. The **Cancel** button just closes the window and does not have any effect on the present structure displayed in the display panel.

An option to do the calculation using a different lattice (**lattice type**) is also present in the panel, however, in the present version of the tool only cubic lattice is available. Below this option, a checkbox is available for building the structure manually using a graphical interface, but this feature is also not finished in the present work.

4.2.2 Display Unit

The Display unit displays the 3D structure of the user-specified sequence of residues(Figure 4.3). This visual unit displays: the structure when the user initially enters the sequence, the calculated structure during the optimization process after a user-specified number of generations, and the final optimized structure.

The structure is constructed from the sequence of directions. The bonds between residues are represented with six possible directions. These six directions, *left (l), right (r), in (i), out (o), up (u) and down (d)*, respectively, represent the connections between residues in the -X, +X, out of screen, towards the screen, upward and downward directions of the monitor screen. Red is used for hydrophobic residues and green is used for hydrophilic ones. The residues position in the sequences can be expressed with the help of numbers. The user can hide/display the numbering using the **N** button from the keyboard. Similarly, the **B** button can be used to hide/display the residues' spheres.

4.2.3 Calculation Setup

The calculation setup is the main component where all the calculation parameters are specified by the user. This is the next step of calculations when the user enters the sequence. The pop-up menu from the **Calculate** tab present in the menu bar displays option for **Native Structure** and **Fitness score**. The **Fitness score** option is to calculate the fitness score of the displayed structure. This **Native Structure** option is used to setup parameters for structure optimization. This button opens a new **Native Structure** window. The window consists

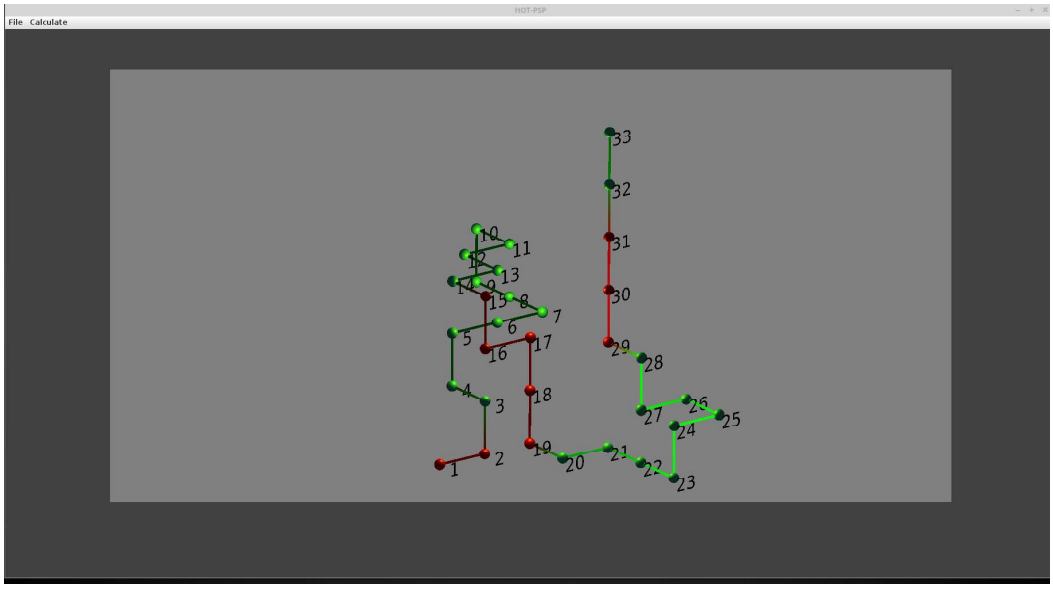

Figure 4.3: An example to display the protein structure in 3D cubic lattice.

of three tabs, **Algorithmic Configuration, Population, and Output Setup**, which contain different input fields required to calculate optimized structure. These tabs and their different fields are discussed below:

4.2.4 Algorithmic Configuration Tab

The **Algorithmic Configuration Tab** is used to setup and choose the algorithm/steps for calculations. The different available options in this tab are shown in Figure 4.4. The **Base Algorithm** option is the default option to select the base algorithm for calculations. The **Algorithm** drop down box gives the option to select an algorithm out of *"GA", "PGA-IM", "PGA-GM", "IA", "ACO", "PSO"*. The *"GA", "PGA-IM", "PGA-GM", "IA", "ACO"*, and *"PSO"*, which are, respectively, represent genetic algorithm, parallel genetic algorithms-island model, parallel genetic algorithms-grid model, immune algorithm, ant-colony algorithm, and particle swarm algorithms. *"PGA-IM"* and *"PGA-GM"* options require the user to enter the number of threads for parallel calculations using a checkbox field present at the bottom of the window. Otherwise, using default option of a single thread, these algorithms

work the same as *"GA"*. The optional **Add. Step** adds an additional step for calculations into the base algorithm steps. In the current version of the software only **" SCO"** (discussed in 3.7) option is available. Beside **Add. Step** option, two other fields are present to enter information related to the individuals used in algorithmic operations. The first field is to enter the number of these individuals or no. of parents, and the **Method** drop down box is used to select the individuals' selection method.

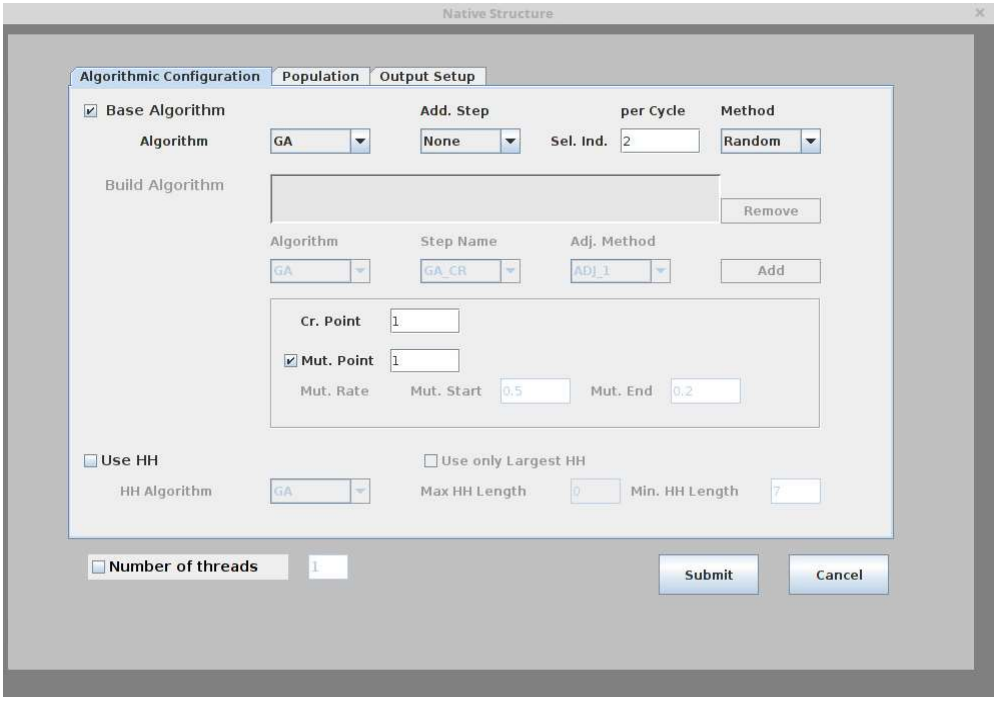

Figure 4.4: Native Structure window with Algorithm Configuration tab.

The user can also construct an algorithm using the **Build Algorithm** option. To do this the user needs to uncheck the **Base Algorithm** checkbox, which gives access to the three drop down boxes. These boxes are for algorithm name (**Algorithm**), step name (**Step Name**) and adjustment method (**Adj. Method**). The first two boxes are used to select a step from corresponding algorithms. The adjustment method is required to fit the structure into self-avoiding criteria. In the current version, only one adjustment method is present. The user can enter the adjust method after each step, however, if no adjust method is chosen, the tool

automatically add the default method after last step. To construct an algorithm (sequence of steps) the user needs to add the chosen steps into the sequence using the **Add** button. The user can also remove any step with the help of the **Remove** button. One can examine the selected steps from the rectangular space available on the top of the three drop-down boxes. All steps can be used any number of times, however, the sequence of steps should contains independent steps or have all dependent steps present at proper positions. In the GA's, *Crossover* and *Mutation* operations are independent steps. The operators, *hyper-mutation* and *hyper-macro-mutation*, suggested for immune algorithm are also independent. However, in certain situations, all the user entered steps are not possible to perform. In that case the **Submit** button gives an error and the user has to correct the sequence of steps in order to perform calculations. The **Use HH** checkbox can be used for separate hydrophobic-hydrophobic (HH) calculations for HH sequences present in the main HP sequence (discussed in section 3.2). The option is available only for *"GA","PGA-IM","PGA-GM"* and *"ACO"* algorithms. The user can select the algorithm and the maximum and minimum lengths of the HH sequence to be considered for HH calculations.

4.2.5 Population Tab

In the **Population Tab** the user can enter information about the individuals and constraints on the individuals' structure. This tab is clearly separated into two sections of fields (shown in Figure 4.5). The first section contains fields for the individual generation method and total population. Two options are possible for generation method "Random" and "Ant-Colony". Both options are available for all algorithms except "ACO", where only "Ant-Colony" option can work. The next two fields are for population size and number of generations labeled under **Population** and **Generations** respectively. There is also a drop down box for the type of **Symmetry** followed by the individuals. In the present work, only "Mirror" symmetry option is added. The "None" option is the default case which means the individuals are not required to obey any symmetry property. The **Allow Twins** checkbox can be unchecked to

make sure all the individuals in the population have distinct structures. These **Symmetry** and **Allow Twins** constraints are for initial populations. In order to put these constraints during the calculation process, the user also needs to select them from the second half of tab space. The same options are also available for HH calculations if the **Use HH** option is selected from the **Algorithmic Configuration Tab**.

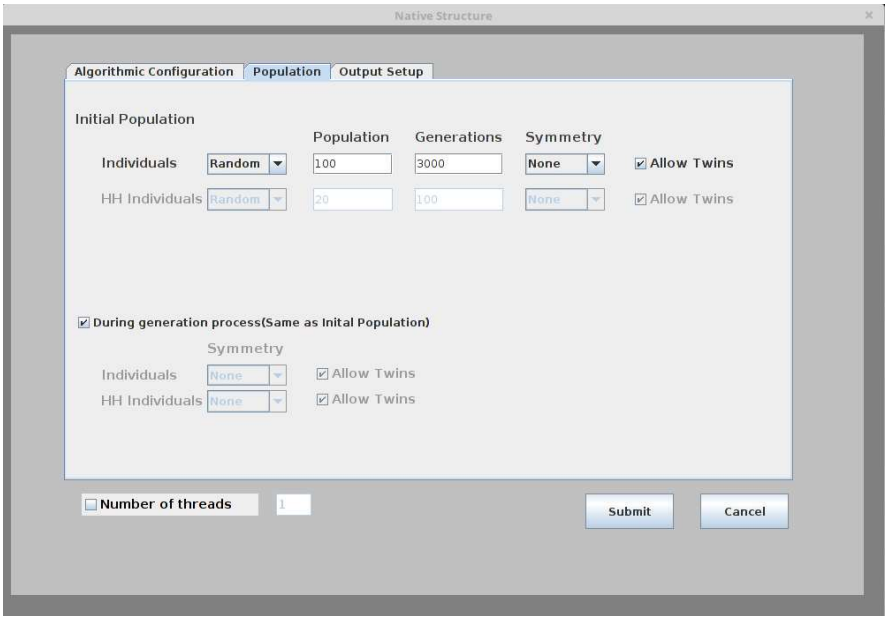

Figure 4.5: Native Structure window with population tab.

4.2.6 OutputSetup Tab

The **OutputSetup Tab** contains the fields required to save the output information. This tab is shown in Figure 4.6. The space labeled with **Directory Name** is to enter the folder's name to save the resultant files. Since the calculations are in the cubic lattice, the output files are automatically saved with the extension *.cube*. The second field, **File Name**, is used to enter the main parts of the file name. The software automatically adds the number at the end of the entered file name based on the number of files entered in the **No. of Files** field. The user can also specify the number of generations after which the populations can be saved in the .cube files. One needs to enter these generations in the field labeled with **Save every**.

The **.log** checkbox used to saves the optimization structures and corresponding fitness scores obtained from all the .cube file. This information is saved in a .log file.

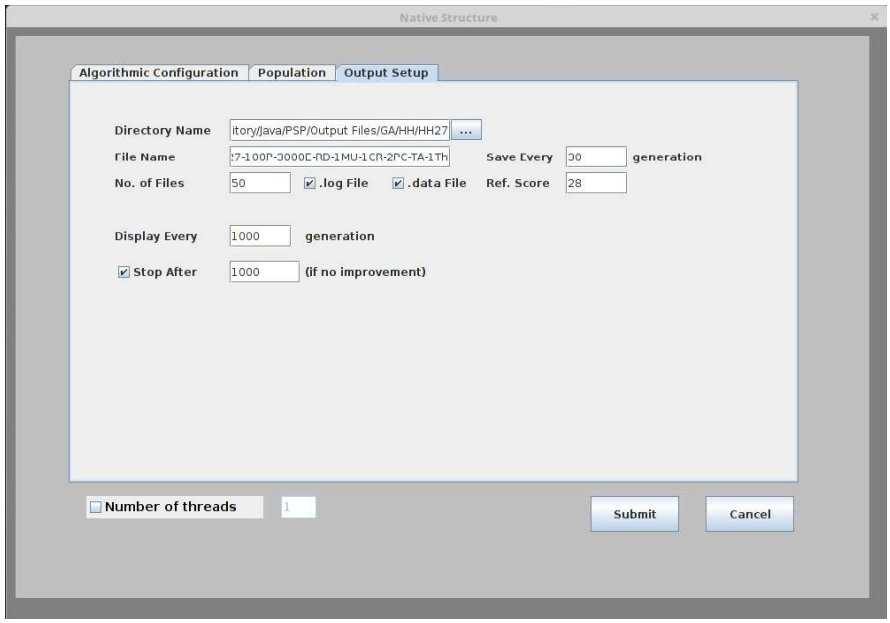

Figure 4.6: Native Structure window with output tab.

An analysis file can be created if the user selects the **.data file** check box. Since this option require a reference score to compare the resultant fitness score, the user needs to enter the reference score in the field titled with the **Ref. Score**. The *.data* file saves the final populations information in different contexts; percentage of population with a highest fitness score (HS), percentage of population with a reference fitness score (PPRS), lowest generation numbers when the fitness score becomes equal to reference score (FRS), lowest generation numbers when half population's fitness scores become equal to the reference fitness score (HPRS) and lowest generation numbers when the whole population's fitness scores become equal to reference fitness score (FPRS). The value entered in the **Display Every** field displays the optimized structure after user-specified number of generations. The **Stop After** checkbox can be used to stop the calculation if no improvement in fitness score has been observed after a certain number of generations.

Chapter 5

Simulation Results

The acceptance of a scientific work can be confirmed by the reproducibility of the reported results within the margins of experimental error. This chapter deals with the simulated results of the developed tool for optimization of protein structures. The results support the correct implementation of the genetic algorithm (GA), parallel genetic algorithms with island model (PGA-IM) and grid model (PGA-GM), immune algorithm (IA), particle swarm optimization algorithm (PSO), and ant-colony optimization algorithm (ACO). The use of the developed application for hybridization has also been confirmed by testing some possible combinations of steps among different algorithms. Further, the analysis of the performance of GA, PGA-IM, and PGA-GM has been analyzed with the number of parent individuals.

5.1 Experimental Details

The experiments to examine the proper working of the developed tool was confirmed by comparing the fitness scores of previously tested sequences with their reported values (Table 5.1). These tests were performed for five implemented algorithms and eight hybrid procedures (C1-C8). C8 hybrid procedure is combination of Genetic Algorithm and Immune Algorithm. Intel(R) Core(TM) i7-7700 CPU @3.6 GHz processor were used for performing different calculations except S48-S1 and S48-S2 sequences in case of ACO and C8 algorithm

where Intel(R) Core(TM) i7-8550U CPU @1.80 GHz processor was used.

5.1.1 HP Sequences

Two sets of previously reported sequences were studied for reproducibility. These sets of sequences are listed in Table 5.1 along with their fitness score. The fitness scores in the Table 5.1 are the best known values reported in [1, 2]. The sequences of length of 27 residues

Table 5.1: The standard protein sequences and their HP model-based best known fitness score (-energy) in 3D cubic lattice used in our calculations [1, 2].

ID	Length	Sequence	Fitness Score
S27-1	27	PHPHPHHHPPHPHPPPPPPPPPPHHP	9
S27-2	27	PHHPPPPPPPPPPHHPPHHPPHPPHPH	10
S27-3	27	HHHHPPPPPHPPPPPHHHPPPPPPPPH	8
S27-4	27	HHHPPHHHHPPPHPHPPHHPPHPPPHH	15
S27-5	27	HHHHPPPPHPHHPPPHHPPPPPPPPPP	8
S27-6	27	HPPPPPPHPHHHPPHHPPPHPPPPHPH	11
S27-7	27	HPPHPHHPPPHPPPPPHPHHPHPHPHH	13
S27-8	27	HPPPPPPPPPPPHPHPPPPPPPPHPHH	4
S27-9	27	PPPPPPPHHHPPPHPHHPPPHPPHPPP	7
S27-10	27	PPPPPHHHPHPHPHPHPPHHPHHPHPPP	11
S48-1	48	HPHHPPHHHHPHHHPPHHPPHPHHHPHPHHPPHHPPPHPPPPPPPPHH	32
S48-2	48	HHHHPHHPHHHHHPPHPPHHPPHPPPPPPHPPHPPPHPPHHPPHHHPH	34
S48-3	48	PHPHHPHHHHHHPPHPHPPHPHHPHPHPPPHPPHHPPHHPPHPHPPHP	34
S48-4	48	PHPHHPPHPHHHPPHHPHHPPPHHHHHPPHPHHPHPHPPPPHPPHPHP	33
S48-5	48	PPHPPPHPHHHHPPHHHHPHHPHHHPPHPHPHPPHPPPPPPHHPHHPH	32
S48-6	48	HHHPPPHHPHPHHPHHPHHPHPPPPPPPHPHPPHPPPHPPHHHHHHPH	32
S48-7	48	PHPPPPHPHHHPHPHHHHPHHPHHPPPHPHPPPHHHPPHHPPHHPPPH	32
S48-8	48	PHHPHHHPHHHHPPHHHPPPPPPHPHHPPHHPHPPPHHPHPHPHHPPP	31
S48-9	48	PHPHPPPPHPHPHPPHPHHHHHHPPHHHPHPPHPHHPPHPHHHPPPPH	34
S48-10	48	PHHPPPPPPHHPPPHHHPHPPHPHHPPHPPHPHHPPHHHHHHHPPHH	33

(S27-1 to S27-10) were taken from Custódio and Barbosa's work [1]. The sequences have also been used by other authors [163]. The second set of sequences of 48 residues (S48-1 to S48-10) were chosen from a paper reported by Maher *et al.* [2]. Albrecht *et al.* have also employed the same set of sequences for their stochastic protein folding [164].

5.1.2 Calculation Parameters

The performance of the implemented algorithms was checked against each sequence with a fixed set of parameters. Both sets of sequences were tested with a population of 100 individuals. Populations in all tests were created using random directions except in case of the ACO algorithm. In the case of ACO, the population was produced with the default ACO method. The number of generations used to test sequences S27 and S48 were 5000 and 10000 respectively. This implies that the total number of algorithmic operations performed for S27 and S48 were 5000 and 10000 times respectively. In order to maintain the same number of algorithmic operations among PGA-IM, the number of generations were reduced to 250 and 500, respectively, for S27 and S48 sequences. Due to the same reason, the calculations for PGA-GM were employed for 1000 and 2000 generations in case of S27 and S48 respectively. The data of each sequence of both sets was collected over 50 runs. Other parameters employed in case of each algorithm are separately listed below:

- GA-Crossover = One-point, Mutation = One-point
- PGA-IM-Crossover = One-point, Mutation = One-point, Cycle/gen=20, Ind./gen. = 2, Island Method =Random, Threads=5.
- PGA-GM-Crossover = One-point, Mutation = One-point, Ind./gen. = 2, Threads=5.
- PSO-mF=100 generations, mB=10 generations,
- ACO- $\rho = 0.5, \tau = 0.5, \alpha = 0.5, \gamma = 0.5$, $\beta = 0.5$, Local Individuals = 10 .

- IA- Clones No. = 10, c = 0.5, Age Limit (τ_B) = 100, Age List Size = 500.

The seven hybrid procedures (C1-C7) among possible steps and a hybridization test of genetic algorithm and immune algorithm (C8) were analyzed. These combinations were tested for the S48-1 sequence using a population of 100 individuals and the statistics was analyzed over 50 runs. The steps and parameters employed in these combinations are given below and A population of 100 individuals were used except otherwise mentioned:

- C1-**Steps-** Crossover (GA) + Micro-Mutation (IA) + Hyper-Macro-Mutation (IA)
  Parameters-Generation = 10000, Ind./gen. = 2, Crossover point = 3, Hyper Mutation c = 0.5

- C2-**Steps**- Cloning (IA) + Crossover (GA) + Mutation(GA)
  Parameters-Generation = 10000, Ind./gen. = 2, Clones No. = 10, Crossover point = 2, Mutation point = 2

- C3-**Steps-** Crossover (GA) + Cloning (IA) +Aging Operation (IA)
  Parameters-Generation = 20000, Ind./gen. = 2, Crossover point = 2, Age of Individuals = 10, Clones No. = 10 Age List Size= 200

- C4-**Steps**- Crossover (GA) + Local Search (ACO) + Aging Operation (IA)
  Parameters-Generation = 20000, Ind./gen. = 2, Crossover point = 5, Local Individuals = 10, Age Limit = 10, Age List Size= 500

- C5-**Steps-** Cloning (IA) + Crossover (GA) + Local Search (ACO)
  Parameters-Generation = 20000, Ind./gen. = 10, Crossover point = 5, Clones No. = 10, Local Individuals = 10

- C6-**Steps-** Crossover (GA) + Hyper-Macro-Mutation (IA)
  Parameters-Generation = 20000, Ind./gen. = 2, Crossover point = 5, Hyper Mutation c = 0.5, Age Limit = 10, Age List Size= 500

- C7-**Steps**-Crossover (GA) + Local Search (ACO)

  Parameters-Generation = 20000, Ind./gen. = 2, Crossover point = 5, Local Individuals = 10

- C8-**Steps-**GA + IA

  Parameters-Generation = 40000, Population= 200, Ind./gen. = 2, Crossover point = 1, Mutation point = 1, Clones No. = 50, c = 0.5, Age Limit (τ_B) = 50, Age List Size = 500.

The analysis of the selected number of parent individuals per cycle in the case of GA, PGA-IM, and PGA-GM was performed using One-point Crossover and One-point Mutation. The study is conducted for the 2, 5, and 10 individuals, and all results were collected over 50 runs for a population of 100 individuals.

5.2 Results and Discussions

5.2.1 Algorithm Analysis

The analysis of the implemented base algorithms has been done by optimizing both sets of sequences and comparing their fitness scores w.r.t the previously reported best fitness scores. Both sets of sequences have also been utilized to compare different algorithms in terms of average fitness scores (AS) and the average time for some fixed number of generations (AT). Different comparisons of each sets of sequences (S27 and S48) are illustrated with the help of bar graphs.

The best fitness scores (BS) of all S27 sequences calculated for 50 independent runs over 5000 generations are shown in Figure 5.1. It can be observed from the graph that all the algorithms are able to achieve the previously reported best fitness scores. However, ACO and PSO algorithms in case of S27-7 sequence are not able to achieve the fitness score as reported by the Patton *et al.* [163]. This might be because these implemented algorithms are

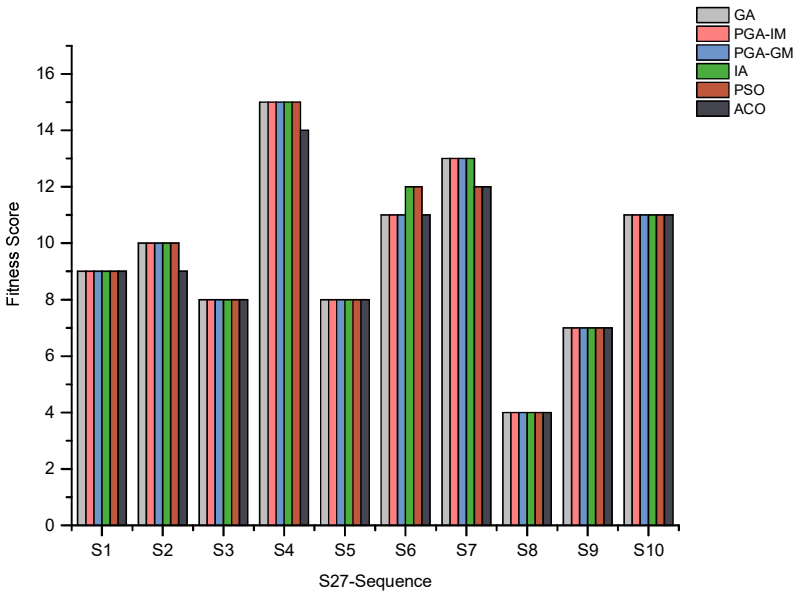

Figure 5.1: The best fitness scores of S27 sequences out of 50 runs for implemented GA, PGA-IM, PGA-GM, IA, PSO, and ACO algorithms.

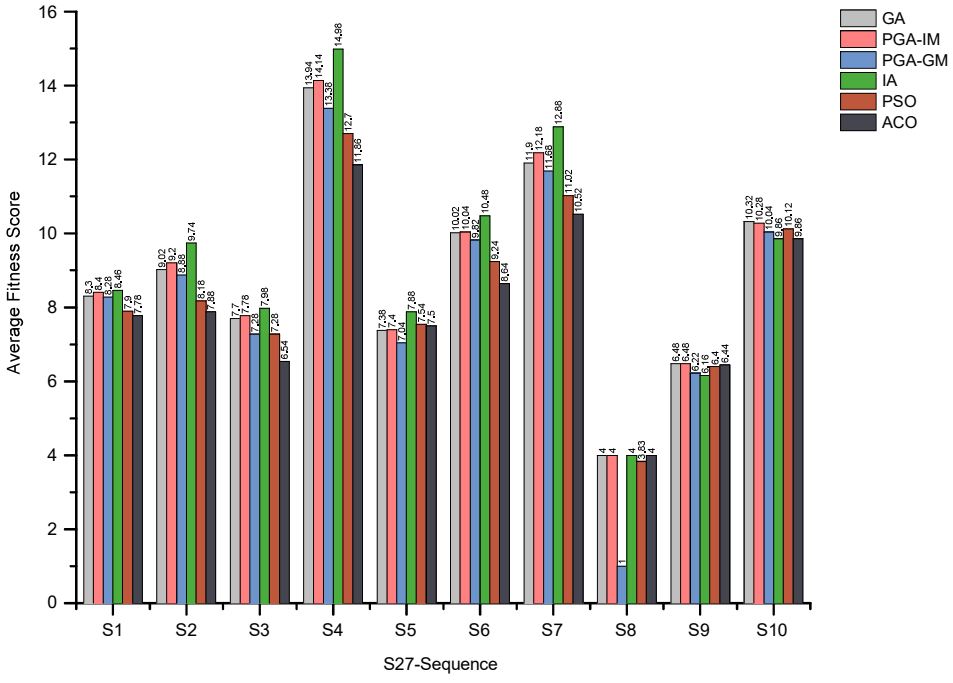

Figure 5.2: The average scores of S27 sequences calculated over 50 runs for implemented GA, PGA-IM, PGA-GM, IA, PSO, and ACO algorithms.

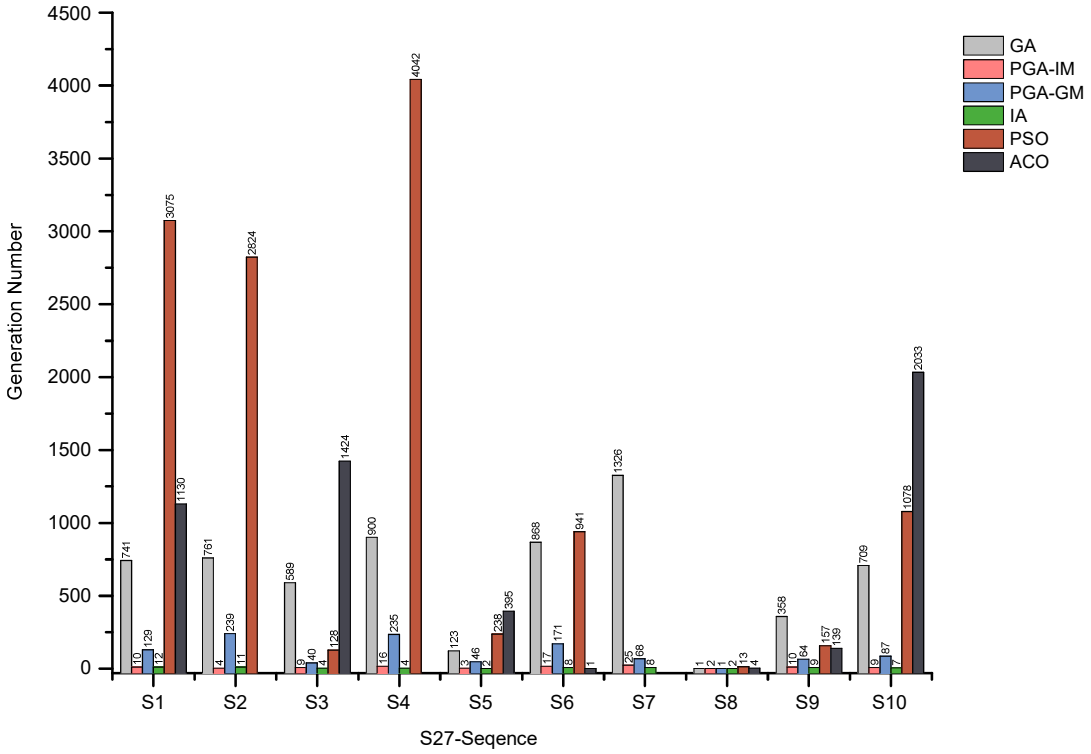

Figure 5.3: Comparison of generation number for best scores of S27 sequences calculated over 50 runs for implemented base algorithms. In some sequences ACO is not able to reach the reported optimal values.

not efficient enough to reach standard value within 5000 generations. Figure 5.2 compares the AS of the optimized structures obtained for different algorithms of S27 sequences. The plot clearly indicates that the AS in most sequences are higher for IA and lower in case of ACO and PSO algorithms. The plot also reveals that the AS of S27-1, S27-5, S27-8, S27-9 and S27-10 are almost similar in all algorithms.

A comparison of the lowest generation number to achieve the previously reported best fitness scores (LG) among all algorithms has also been conducted and Figure 5.3 exhibits the LG values for S27 sequences. It can be seen that the LG values are lower in case of IA and highest in PSO and ACO algorithms. The LG values for S27-3, S27-5, S27-8, S27-9 from most algorithms are lower than S27-1, S27-2, S27-4, S27-6, S27-10, and S27-8. This can

be attributed to the number of hydrophobic residues, *i.e.*, possible number of hydrophobic contacts, and also to the internal arrangement of the positions of hydrophobic residues. It can also be observed that LG values from GA are higher than PGA-IM and PGA-GM. In the case of PGA-IM and PGA-GM, these values can be further decreased by using greater number of threads. In our calculations, we have used 5 threads which means that each generation of PGA-GM is equivalent to five GA generations. Similarly, in case of PGA-IM, every generation has done the same work as 100 GA generations. The LG values for IA and PGA algorithms are very close to each other compared to other algorithms.

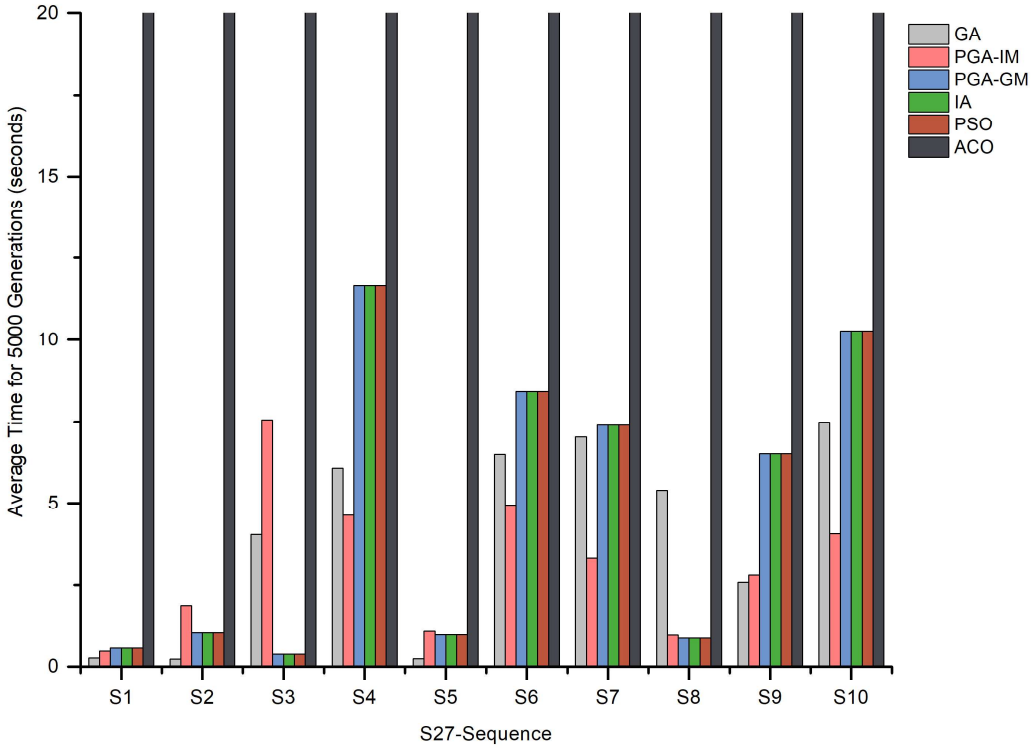

Figure 5.4: The average time taken by S27 sequences to complete 5000 generations for all the implemented base algorithms. The average is calculated over 50 runs. The average times in case ACO algorithm are much higher and are not fully covered in the graph. The exact average times taken is listed in Supplementary Tables A.1. Average times of 5000 genetic operations in case of PGA-IM and PGA-GM are taken for 250 and 1000 generations, respectively.

Figure 5.4 illustrates the bar graph of the average times of calculations (AT) of all the S27 sequences for all implemented algorithms. The graph shows that AT values of S27-1, S27-2, S27-3, S27-5, and S27-8 sequences are lower than the rest of the sequences. This observation is neither in accordance with the number of hydrophobic residues present in the sequence, nor due to the number of possible hydrophobic-hydrophobic contacts. Also, since except for the ACO algorithm, the individuals have been generated using a random procedure, the reason cannot be associated with the positions of the residues in the sequence. Performance wise, the fitness score graph reveals that the IA is better than other algorithms, however in most cases, the average time of calculations of IA is higher as compared to GA and PGA-IM. The average time of calculations of all the S27 sequences from PGA-GM, PSO and IA have almost the same values. The ACO algorithm seems much more expensive in the context of average time.

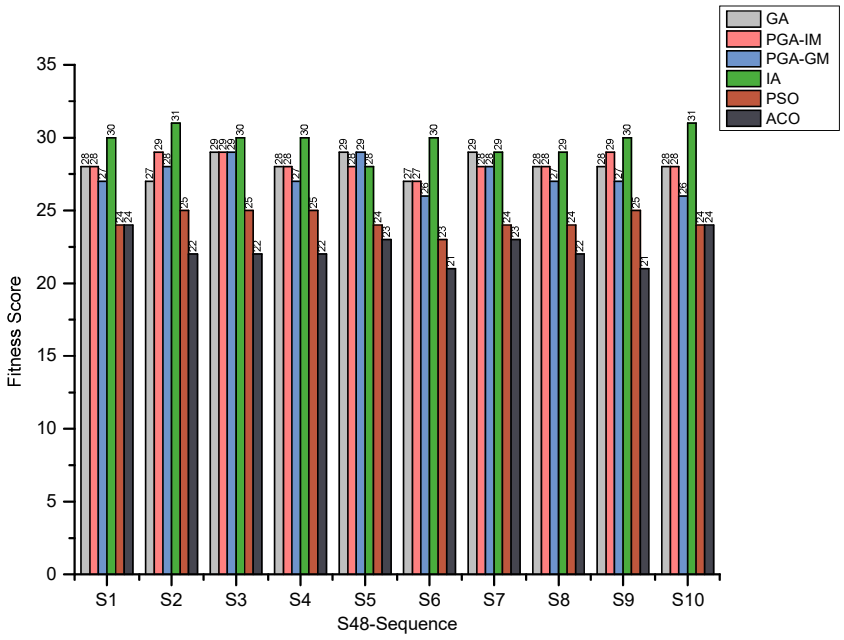

Figure 5.5: The highest scores of S48 sequences out of 50 runs for implemented GA, PGA-IM, PGA-GM, IA, PSO, and ACO algorithms.

The results of the S48 sequences are illustrated in Figures 5.5, 5.6 and 5.7. The analysis is

done for the highest fitness score (HS), average score (AS) and average time (AT) calculated for 10000 generations over 50 runs. Figure 5.5 shows the highest values of the fitness scores obtained by different algorithms within 10000 generations. The fitness score values are quite reasonable and close to the previously reported best fitness scores to predict the proper implementation of the algorithms. The graphs reveal that the IA also out-performed over all other algorithms as in case S27. Except for S48-2 sequence, the calculated HS values from PGA-GM are lower than those from the GA and PGA-IM. The graph between AS vs S48-sequences for different algorithms is plotted in Figure 5.6. The AS of different sequences of different algorithms increases in the following order: ACO, PSO, PGA-GM, GA, PGA-IM and IA. This order is similar to S27 (Figure 5.2), however, due to the larger length of the S48 sequences, the differences are higher.

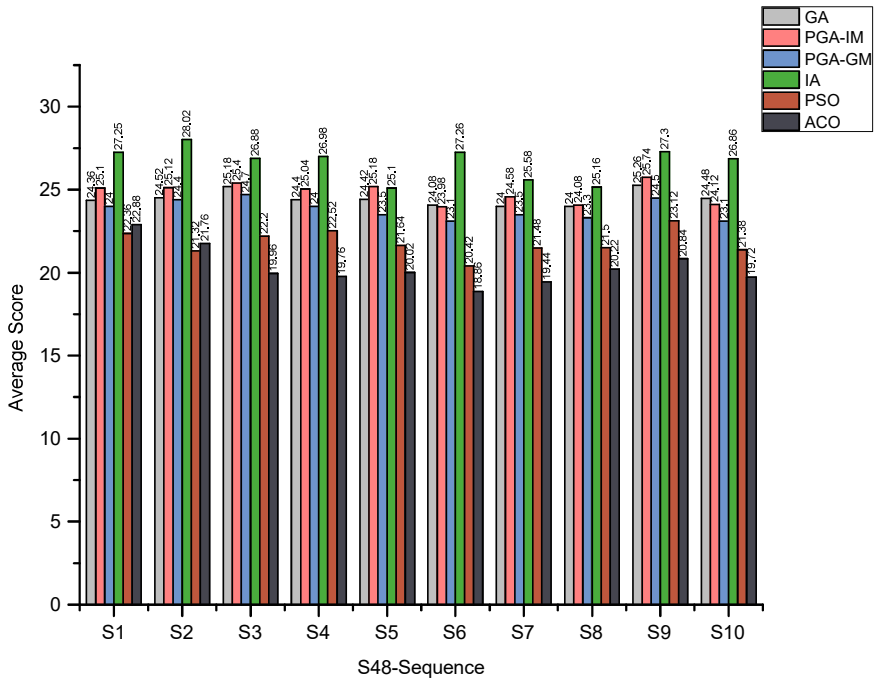

Figure 5.6: The average scores of S48 sequences calculated over 50 runs for implemented GA, PGA-IM, PGA-GM, IA, PSO, and ACO algorithms.

The average time graph for S48 sequences is illustrated in Figure 5.7. It represents that the ACO algorithm takes much higher time than the other algorithms. Average time taken

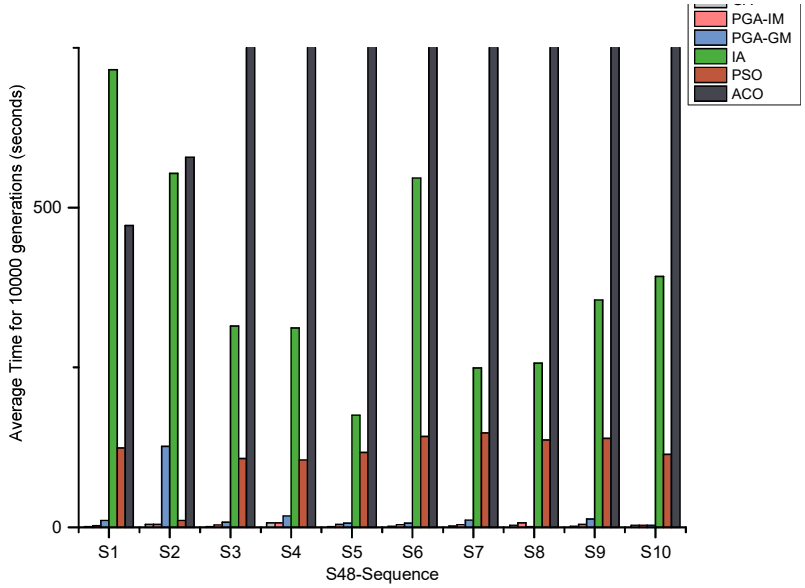

Figure 5.7: The average time taken by S48 sequences to complete 10000 generations for all the implemented base algorithms. The average is calculated over 50 runs. The average times in case the ACO algorithm are much higher and are not fully covered in the graph. The exact average times are listed in Supplementary Table A.2. Average times for 10000 genetic operations in case of PGA-IM and PGA-GM are taken for 500 and 2000 generations respectively.

Table 5.2: Average score (AS) and Best score (BS) for different set of hybrid procedures.

Cal. Set	AS	BS
C1	23.58	28
C2	25.00	25
C3	22.24	27
C4	25.08	29
C5	18.88	22
C6	28.98	29
C7	26.26	28
C8	24.74	27

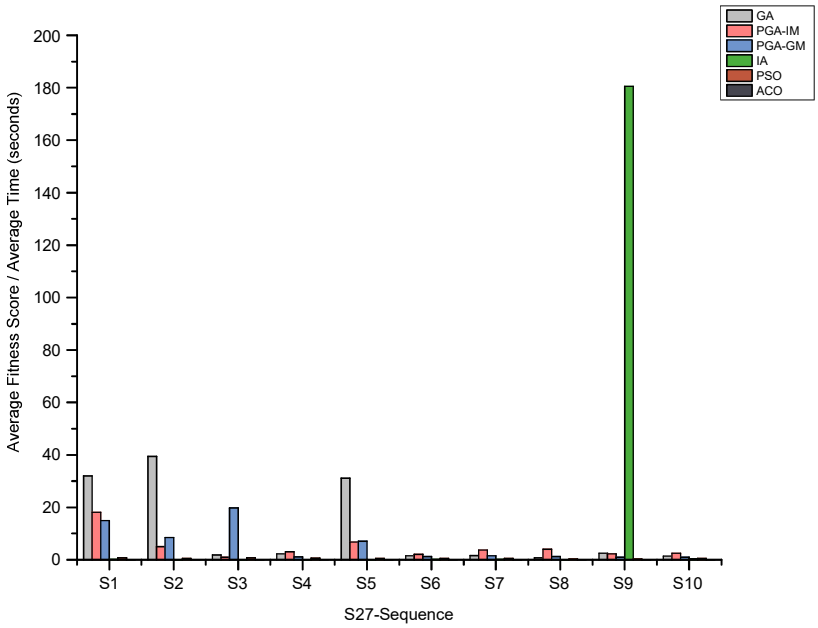

Figure 5.8: The average scores of S27 sequences calculated over 50 runs for implemented GA, PGA-IM, PGA-GM, IA, PSO, and ACO algorithms.

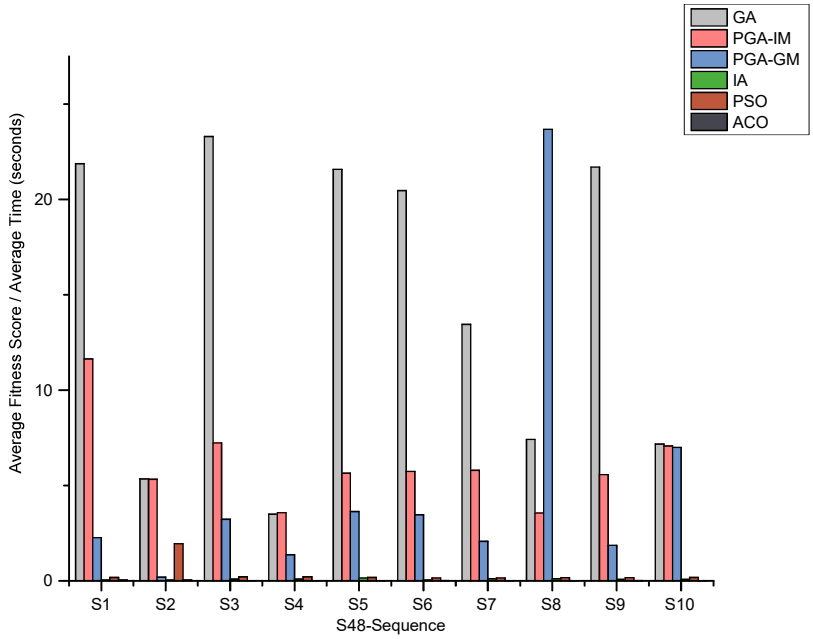

Figure 5.9: The average scores of S48 sequences calculated over 50 runs for implemented GA, PGA-IM, PGA-GM, IA, PSO, and ACO algorithms.

by GA, PGA-IM, and PGA-GM are almost same for all the sequences. The average time taken by PSO algorithm is higher than GA but lower than IA. The similar trend among different algorithms for each sequence clearly suggests that the reason of the differences in average times of calculations cannot be attributed to the internal arrangement of residues in the sequences which could be one possibility in case of S27.

The performance of different algorithms from both the sets of sequences are compared using average fitness score/ average time (shown in Figures 5.8 and 5.9). Both the graphs show that in most of the cases the genetic algorithm and parallel genetic algorithm out-perform than other algorithms. The use of the software for hybridization has been tested for eight hybrid procedures (C1-C8). Only the S48-1 sequence has been analyzed for this work. The obtained results of the best and aver-

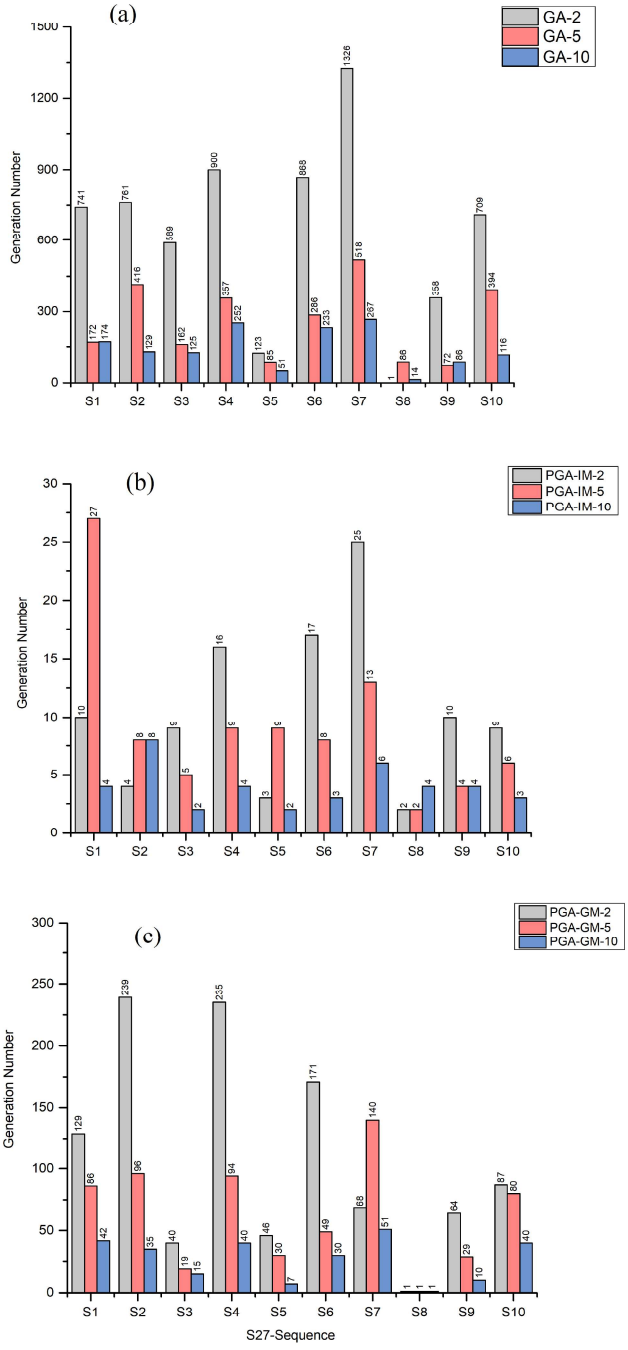

Figure 5.10: The best generation number over 50 runs vs S27 sequences for GA, PGA-IM, PGA-GM, in case of 2, 5, 10 individuals used to perform the algorithmic operation.

age fitness scores are illustrated in Table 5.2. The values are quite reasonable and confirm that the tool can be utilized to search and analyze new hybrid procedures.

The above results ascertain the proper working of the developed tool. We further demonstrated the use of the software for algorithms' analysis based on the number of selected parent individuals. The study has been conducted to analyze the performance of GA, PGA-IM, and PGA-GM. The work has been accomplished by testing both sets of sequences (S27 and S48). The lowest generation number (LG) to achieve the previously reported best fitness scores of S27 sequences are shown in Figure 5.10. It can be seen that as the number of individuals increases, the LG values decreases. The reason can be attributed to the higher number of individuals involved in each step. The details of the calculated data for S27 and S48 sequences are, respectively, listed in Supplementary Tables A.3 and A.4.

The Figure 5.10 shows that the average scores in both cases, S27 and S48, increases with number of parent individuals. These observations suggest that the average time (AT) should also increase with more individuals, however, in our calculations, we have not observed this trend. In the case of GA and PGA-IM, the AT does not increase with the increase of individuals. Similarly, for S48 sequences the PGA-GM does not follow the expected pattern. This indicates that there is more randomness involved in time calculations. It might be due to other system processes which were running during calculation time.

5.3 Conclusions

The results obtained for different base algorithms and hybridization of possible algorithmic steps, illustrate that the developed tool (HOT-PSP) is working as purposed. Since this is a comprehensive tool to study simplified protein structure, the detailed analysis of different algorithms and hybrid procedures can be performed in a simple manner. Our simulated analysis of the base algorithms shows that the Immune algorithm performs better than other algorithms, however, it takes more time for the same number of generations as compared to PGA-IM and PGA-GM. The performance of the implemented ACO and PSO to calculate

the fitness scores is not the same as stated in the literature. The use of different values of input parameters might raise the fitness scores in these algorithms. The hybrid procedures tests suggest that the software can be used to search new optimization procedures. The hybrid of GA and IA algorithm (C8) is tested and the C8's average score is higher than GA's average score, however, reduces the performance to reach the optimal score. The user can test other hybrid procedures or can use other parameters in order to search for better algorithm. The calculation time in case of PGA-IM and PGA-GM can be further decreased by having access to the more threads. The comparison of average fitness scores among GA, PGA-IM and PGA-GM suggests that the fitness scores in case of PGA-IM's is higher than GA for both sets of sequences. The GA, PGA-IM and PGA-GM analysis of the number of parents indicates that it is better to use a higher number of individuals per generation.

Chapter 6

Summary and Future Plan

Protein structure prediction (PSP) is one of the important optimization problems. The importance of PSP can be advocated from the significance of proteins and their functional dependency on their structure. The HP model is a simple model which is widely used for simplified protein structure prediction. This thesis presents an automation and visualization application for simplified protein structure prediction using the HP model. The optimized structure can be obtained from meta-heuristic algorithms or using a possible combination of different algorithmic steps. The working of the tool is tested with the help of two different sets of standard benchmark sequences. The results of these benchmark sequences are consistent with the reported studies. The simulated analysis further reveals that Immune algorithm performed better than other implemented algorithms, however, the average time for the same number of generations were much lower in the case of PGA-IM and PGA-GM. These results indicate the successful accomplishment of the main objective to develop an optimization tool for simplified protein structure prediction. For efficient and effective working of the developed software, we incorporated it with some important characteristics which are summarized as below:

- The software consists of an interactive GUI which is a user friendly environment for calculation setup as well as to display the output results based on a user-defined number of generations.

- The software provides the facility to display the HP model-based protein structure in three-dimensional cubic lattice.
- The fitness score (or energy) of a given sequence can be calculated for randomly generated directions. The fitness score can also be calculated for the user-specified directions (structure).
- Hybridization among possible steps of implemented algorithms can be performed on demand bases, and software itself handles (warning) unacceptable algorithmic configurations.
- Constraints such as mirror symmetry and twin removal can be introduced on demand basis.
- Separate calculations can be performed for hydrophobic-hydrophobic (HH) sequences present inside the main HP sequence. Parallelism can be performed on some computationally expensive steps using a higher number of threads.
- The development work has been done using the Java programming language and Java based OpenGL. Java has many advantages such as object-oriented, cross-platform, robust, secure, easy for distributed computing.
- The user can set or define save options for output results. Multiple output files can be generated for a given set of calculation parameters and data obtained from all these files can be saved in a single text file.

6.0 Future Goals

We have developed the software for simplified protein structure prediction using a few algorithms and some additional features. However, there are different directions in which the present tool can be extended in the future:

- Symmetry is an important feature of globular proteins and many other molecules. In the present version we have only introduced the mirror symmetry constraint, however, we would like to add options for other types of symmetry constraints

- The code can be further extended to compare other optimization algorithms and/or with some other features which can be helpful to optimize the proteins closer to the real picture.

- We would also like to add features for manual designing and editing of protein structure from the graphical display. This feature will be useful for easy construction of the initial structure and also for editing the structure during the calculation process.

- The designed software will be more useful if it can compare the calculated structure with the experimentally obtained structure. Introducing features such as root mean square differences (RMSD), standard deviation, curve fitting, etc. will be helpful for this purpose.

Bibliography

[1] Fábio L Custódio, Hélio JC Barbosa, and Laurent E Dardenne. Investigation of the three-dimensional lattice hp protein folding model using a genetic algorithm. *Genetics and Molecular Biology*, 27(4):611–615, 2004.

[2] Brian Maher, Andreas A Albrecht, Martin Loomes, Xin-She Yang, and Kathleen Steinhöfel. A firefly-inspired method for protein structure prediction in lattice models. *Biomolecules*, 4(1):56–75, 2014.

[3] Albert Y Zomaya. *Parallel computing for bioinformatics and computational biology: models, enabling technologies, and case studies*, volume 55. John Wiley & Sons, 2006.

[4] Rui Zhang, Shiji Song, and Cheng Wu. A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem. *Knowledge-Based Systems*, 27:393–406, 2012.

[5] Yin-Yann Chen, Chen-Yang Cheng, Li-Chih Wang, and Tzu-Li Chen. A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems—A case study for solar cell industry. *International Journal of Production Economics*, 141(1):66–78, 2013.

[6] Vincent Kelner, Florin Capitanescu, Olivier Léonard, and Louis Wehenkel. A hybrid optimization technique coupling an evolutionary and a local search algorithm. *Journal of Computational and Applied Mathematics*, 215(2):448–456, 2008.

[7] Xiaojia Liu, Haizhong An, Lijun Wang, and Xiaoliang Jia. An integrated approach to optimize moving average rules in the eua futures market based on particle swarm optimization and genetic algorithms. *Applied Energy*, 185:1778–1787, 2017.

[8] Xu Wang, Zi-Yu Li, and Jia-Yu Zhong. Construction of quantitative transaction strategy based on lasso and neural network. *Applied Economics and Finance*, 4(4):134–144, 2017.

[9] Anh-Tuan Nguyen, Sigrid Reiter, and Philippe Rigo. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058, 2014.

[10] Petr Dostál. The use of optimization methods in business and public services. In *Handbook of Optimization*, pages 717–777. Springer, 2013.

[11] Carl J Walters and Ray Hilborn. Ecological optimization and adaptive management. *Annual review of Ecology and Systematics*, 9(1):157–188, 1978.

[12] RA Ynchausti, LB Hales, and KS Gritton. Knowledgescaping: on-line, adaptive optimizing control methods. In *Dynamic Modeling Control Applications for Industry Workshop, 1997., IEEE Industry Applications Society*, pages 62–67. IEEE, 1997.

[13] Viliam Slodicak. How to combine stochastic programming and mathematical theory of programming. In *Applied Machine Intelligence and Informatics, 2008. SAMI 2008. 6th International Symposium on*, pages 267–272. IEEE, 2008.

[14] American Association for the Advancement of Science et al. So much more to know.... *Science*, 309(5731):78–102, 2005.

[15] Ken A Dill. Dominant forces in protein folding. *Biochemistry*, 29(31):7133–7155, 1990.

[16] Peter E Wright. 50 years of protein structure determination. *Journal of Molecular Biology*, 1(392):1, 2009.

[17] Cyrus Levinthal. Are there pathways for protein folding? *Journal de chimie physique*, 65:44–45, 1968.

[18] Immo E Scheffler, Elliot L Elson, and Robert L Baldwin. Helix formation by d (ta) oligomers: Ii. analysis of the helix-coil transitions of linear and circular oligomers. *Journal of molecular biology*, 48(1):145–171, 1970.

[19] Jean-Renaud Garel and Robert L Baldwin. Both the fast and slow refolding reactions of ribonuclease a yield native enzyme. *Proceedings of the National Academy of Sciences*, 70(12):3347–3351, 1973.

[20] Frederick M Hughson, Doug Barrick, and Robert L Baldwin. Probing the stability of a partly folded apomyoglobin intermediate by site-directed mutagenesis. *Biochemistry*, 30(17):4113–4118, 1991.

[21] Philipp Neudecker, Paul Robustelli, Andrea Cavalli, Patrick Walsh, Patrik Lundström, Arash Zarrine-Afsar, Simon Sharpe, Michele Vendruscolo, and Lewis E Kay. Structure of an intermediate state in protein folding and aggregation. *Science*, 336(6079):362–366, 2012.

[22] Donald B Wetlaufer. Nucleation, rapid folding, and globular intrachain regions in proteins. *Proceedings of the National Academy of Sciences*, 70(3):697–701, 1973.

[23] S Reich and J Yonath. Study of collagen denaturation kinetics by dynamic mechanical methods. *Biopolymers*, 6(7):997–1000, 1968.

[24] Fabrizio Chiti and Christopher M Dobson. Amyloid formation by globular proteins under native conditions. *Nature chemical biology*, 5(1):15–22, 2009.

[25] Kit Fun Lau and Ken A Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10):3986–3997, 1989.

[26] William E Hart and Sorin Istrail. Robust proofs of np-hardness for protein folding: general lattices and energy potentials. *Journal of Computational Biology*, 4(1):1–22, 1997.

[27] Ron Unger and John Moult. Finding the lowest free energy conformation of a protein is an np-hard problem: proof and implications. *Bulletin of mathematical biology*, 55(6):1183–1198, 1993.

[28] Bonnie Berger and Tom Leighton. Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete. *Journal of Computational Biology*, 5(1):27–40, 1998.

[29] Aviezri S Fraenkel. Complexity of protein folding. *Bulletin of mathematical biology*, 55(6):1199–1210, 1993.

[30] James DeGregori, Gustavo Leone, Alexander Miron, Laszlo Jakoi, and Joseph R Nevins. Distinct roles for e2f proteins in cell growth control and apoptosis. *Proceedings of the National Academy of Sciences*, 94(14):7245–7250, 1997.

[31] Haim Garty and Steven JD Karlish. Role of fxyd proteins in ion transport. *Annu. Rev. Physiol.*, 68:431–459, 2006.

[32] James C Whisstock and Arthur M Lesk. Prediction of protein function from protein sequence and structure. *Quarterly reviews of biophysics*, 36(03):307–340, 2003.

[33] Roy D Sleator and Paul Walsh. An overview of in silico protein function prediction. *Archives of microbiology*, 192(3):151–155, 2010.

[34] Julie S Valastyan and Susan Lindquist. Mechanisms of protein-folding diseases at a glance. *Disease models & mechanisms*, 7(1):9–14, 2014.

[35] Fred E Cohen and Jeffery W Kelly. Therapeutic approaches to protein-misfolding diseases. *Nature*, 426(6968):905–909, 2003.

[36] Enrique Reynaud. Protein misfolding and degenerative diseases. *Nature Education*, 3(9):28, 2010.

[37] Vernon M Ingram et al. Gene mutations in human haemoglobin: the chemical difference between normal and sickle cell haemoglobin. *Nature*, 180(4581):326–328, 1957.

[38] Claudio Soto. Unfolding the role of protein misfolding in neurodegenerative diseases. *Nature Reviews Neuroscience*, 4(1):49, 2003.

[39] Christopher M Dobson. Protein folding and misfolding. *Nature*, 426(6968):884, 2003.

[40] Anthony L Fink. Compact intermediate states in protein folding. *Annual review of biophysics and biomolecular structure*, 24(1):495–522, 1995.

[41] GN t Ramachandran and V Sasisekharan. Conformation of polypeptides and proteins. *Advances in protein chemistry*, 23:283–437, 1968.

[42] Sven Hovmöller, Tuping Zhou, and Tomas Ohlson. Conformations of amino acids in proteins. *Acta Crystallographica Section D: Biological Crystallography*, 58(5):768–776, 2002.

[43] Haipeng Gong and George D Rose. Does secondary structure determine tertiary structure in proteins? *Proteins: Structure, Function, and Bioinformatics*, 61(2):338–343, 2005.

[44] Lewis E Kay. Nmr studies of protein structure and dynamics. *Journal of Magnetic Resonance*, 173(2):193–207, 2005.

[45] Teppei Ikeya, Tomomi Hanashima, Saori Hosoya, Manato Shimazaki, Shiro Ikeda, Masaki Mishima, Peter Güntert, and Yutaka Ito. Improved in-cell structure determination of proteins at near-physiological concentration. *Scientific Reports*, 6:38312, 2016.

[46] Andrea Cavalli, Xavier Salvatella, Christopher M Dobson, and Michele Vendruscolo. Protein structure determination from nmr chemical shifts. *Proceedings of the National Academy of Sciences*, 104(23):9615–9620, 2007.

[47] K Linderstrom-Lang and H Holter. On the ionisation of proteins, 1924.

[48] Charles Tanford and John G Kirkwood. Theory of protein titration curves. i. general equations for impenetrable spheres. *Journal of the American Chemical Society*, 79(20):5333–5339, 1957.

[49] James B Matthew and Frederic M Richards. Anion binding and ph-dependent electrostatic effects in ribonuclease. *Biochemistry*, 21(20):4989–4999, 1982.

[50] James B Matthew and Frank RN Gurd. Calculation of electrostatic interactions in proteins. *Methods in enzymology*, 130:413–436, 1986.

[51] CF Jacobsen and K Linderstrøm-Lang. Salt linkages in proteins. *Nature*, 164(4166):411–412, 1949.

[52] Harold Edelhoch and James C Osborne. The thermodynamic basis of the stability of proteins, nucleic acids, and membranes. *Advances in protein chemistry*, 30:183–250, 1976.

[53] Sun Dao-pin, Eskil Söderlind, Walt A Baase, Joan A Wozniak, Uwe Sauer, and Brian W Matthews. Cumulative site-directed charge-change replacements in bacteriophage t4 lysozyme suggest that long-range electrostatic interactions contribute little to protein stability. *Journal of molecular biology*, 221(3):873–887, 1991.

[54] Alan R Fersht. Conformational equilibria in α-and δ-chymotrypsin: the energetics and importance of the salt bridge. *Journal of molecular biology*, 64(2):497–509, 1972.

[55] MF Perutz and H Raidt. Stereochemical basis of heat stability in bacterial ferredoxins and in haemoglobin a2. *Nature*, 255(5505):256–259, 1975.

[56] David J Barlow and JM Thornton. Ion-pairs in proteins. *Journal of molecular biology*, 168(4):867–885, 1983.

[57] Ken A Dill, Sarina Bromberg, Kaizhi Yue, Hue Sun Chan, Klaus M Ftebig, David P Yee, and Paul D Thomas. Principles of protein folding—a perspective from simple exact models. *Protein Science*, 4(4):561–602, 1995.

[58] Alfred E Mirsky and Linus Pauling. On the structure of native, denatured, and coagulated proteins. *Proceedings of the National Academy of sciences*, 22(7):439–447, 1936.

[59] John C Kendrew, G Bodo, Howard M Dintzis, RG Parrish, Harold Wyckoff, and David C Phillips. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, 181(4610):662–666, 1958.

[60] Paul Doty and Jen Tsi Yang. Polypeptides. vii. poly-γ-benzyl-l-glutamate: The helix-coil transition in solution1. *Journal of the American Chemical Society*, 78(2):498–500, 1956.

[61] Paul Doty, AM Holtzer, JH Bradbury, and ER Blout. Polypeptides. ii. the configuration of polymers of γ-benzyl-l-glutamate in solution1. *Journal of the American Chemical Society*, 76(17):4493–4494, 1954.

[62] P Doty, K Imahori, and E Klemperer. The solution properties and configurations of a polyampholytic polypeptide: Copoly-l-lysine-l-glutamic acid. *Proceedings of the National Academy of Sciences*, 44(5):424–431, 1958.

[63] Walter Kauzmann. *Denaturation of proteins and enzymes*. Johns Hopkins Press: Baltimore, 1954.

[64] Kaizhi Yue and Ken A Dill. Folding proteins with a simple energy function and extensive conformational searching. *Protein Science*, 5(2):254–261, 1996.

[65] C Nick Pace, J Martin Scholtz, and Gerald R Grimsley. Forces stabilizing proteins. *FEBS letters*, 588(14):2177–2184, 2014.

[66] Walter Kauzmann. *The Mechanism of Enzyme Action*, volume 78. Johns Hopkins University Press, Baltimore, 1954.

[67] Walter Kauzmann. Some factors in the interpretation of protein denaturation. *Advances in protein chemistry*, 14:1–63, 1959.

[68] Frank F Schmidt. *CliffsQuickReview Biochemistry II*. Houghton Mifflin Harcourt, 2000.

[69] Susanne Moelbert, Eldon Emberly, and Chao Tang. Correlation between sequence hydrophobicity and surface-exposure pattern of database proteins. *Protein Science*, 13(3):752–762, 2004.

[70] Hue Sun Chan, Sarina Bromberg, and Ken A Dill. Models of cooperativity in protein folding. *Philosophical Transactions: Biological Sciences*, pages 61–70, 1995.

[71] Charles Tanford. Contribution of hydrophobic interactions to the stability of the globular conformation of proteins. *Journal of the American Chemical Society*, 84(22):4240–4247, 1962.

[72] John F Brandts. The thermodynamics of protein denaturation. i. the denaturation of chymotrypsinogen. *Journal of the American Chemical Society*, 86(20):4291–4301, 1964.

[73] Ken A Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

[74] C Nick Pace, Hailong Fu, Katrina Fryar, John Landua, Saul R Trevino, David Schell, Richard L Thurlkill, Satoshi Imura, J Martin Scholtz, Ketan Gajiwala, et al. Contribution of hydrogen bonds to protein stability. *Protein Science*, 23(5):652–661, 2014.

[75] Themis Lazaridis, Georgios Archontis, and Martin Karplus. Enthalpic contribution to protein stability: insights from atom-based calculations and statistical mechanics. In *Advances in protein chemistry*, volume 47, pages 231–306. Elsevier, 1995.

[76] Jairo R Montoya-Torres. Designing supply chains using optimization. In *Encyclopedia of Business Analytics and Optimization*, pages 726–736. IGI Global, 2014.

[77] Lin Tang, Hui Cao, Li Zheng, and Ningjian Huang. Uncertainty-aware rfid network planning for target detection and target location. *Journal of Network and Computer Applications*, 74:21–30, 2016.

[78] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[79] Changjun Zhou, Caixia Hou, Qiang Zhang, and Xiaopeng Wei. Enhanced hybrid search algorithm for protein structure prediction using the 3D-HP lattice model. *Journal of Molecular Modeling*, 19(9):3883–3891, 2013.

[80] Cheng-Jian Lin and Shih-Chieh Su. Protein 3D HP Model Folding Simulation Using a Hybrid of Genetic Algorithm and Particle Swarm Optimization. *International Journal of Fuzzy Systems*, 13(2):140–147, 2011.

[81] Swakkhar Shatabda, M A Hakim Newton, and Abdul Sattar. Mixed Heuristic Local Search for Protein Structure Prediction. *AAAI Conference on Artificial Intelligence*, pages 876–882, 2013.

[82] Xiaolong Zhang, Ting Wang, Huiping Luo, Jack Y Yang, Youping Deng, Jinshan Tang, and Mary Qu Yang. 3d protein structure prediction with genetic tabu search algorithm. *BMC systems biology*, 4(1):S6, 2010.

[83] Abu Dayem Ullah and Kathleen Steinhöfel. A hybrid approach to protein folding problem integrating constraint programming with local search. *BMC Bioinformatics*, 11 Suppl 1:S39, 2010.

[84] Sanzo Miyazawa and Robert L Jernigan. Residue–residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading. *Journal of molecular biology*, 256(3):623–644, 1996.

[85] Sanzo Miyazawa and Robert L Jernigan. Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, 18(3):534–552, 1985.

[86] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[87] William R Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the smith-waterman and fasta algorithms. *Genomics*, 11(3):635–650, 1991.

[88] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[89] Bhusan K Kuntal, Polamarasetty Aparoy, and Pallu Reddanna. Easymodeller: A graphical interface to modeller. *BMC research notes*, 3(1):226, 2010.

[90] Torsten Schwede, Jurgen Kopp, Nicolas Guex, and Manuel C Peitsch. Swiss-model: an automated protein homology-modeling server. *Nucleic acids research*, 31(13):3381–3385, 2003.

[91] Joost Schymkowitz, Jesper Borg, Francois Stricher, Robby Nys, Frederic Rousseau, and Luis Serrano. The foldx web server: an online force field. *Nucleic acids research*, 33(suppl 2):W382–W388, 2005.

[92] Johannes Söding, Andreas Biegert, and Andrei N Lupas. The hhpred interactive server for protein homology detection and structure prediction. *Nucleic acids research*, 33(suppl 2):W244–W248, 2005.

[93] Sitao Wu and Yang Zhang. Lomets: a local meta-threading-server for protein structure prediction. *Nucleic acids research*, 35(10):3375–3382, 2007.

[94] Sitao Wu, Jeffrey Skolnick, and Yang Zhang. Ab initio modeling of small proteins by iterative tasser simulations. *BMC biology*, 5(1):17, 2007.

[95] Jianyi Yang and Yang Zhang. I-tasser server: new development for protein structure and function predictions. *Nucleic acids research*, 43(W1):W174–W181, 2015.

[96] Ambrish Roy, Alper Kucukural, and Yang Zhang. I-tasser: a unified platform for automated protein structure and function prediction. *Nature protocols*, 5(4):725–738, 2010.

[97] Lawrence A Kelley, Stefans Mezulis, Christopher M Yates, Mark N Wass, and Michael JE Sternberg. The phyre2 web portal for protein modeling, prediction and analysis. *Nature protocols*, 10(6):845–858, 2015.

[98] Morten Källberg, Haipeng Wang, Sheng Wang, Jian Peng, Zhiyong Wang, Hui Lu, and Jinbo Xu. Template-based protein structure modeling using the raptorx web server. *Nature protocols*, 7(8):1511–1522, 2012.

[99] Sitao Wu and Yang Zhang. Muster: improving protein sequence profile–profile alignments by using multiple sources of structure information. *Proteins: Structure, Function, and Bioinformatics*, 72(2):547–556, 2008.

[100] Kristian W Kaufmann, Gordon H Lemmon, Samuel L DeLuca, Jonathan H Sheehan, and Jens Meiler. Practically useful: what the rosetta protein modeling suite can do for you. *Biochemistry*, 49(14):2987–2998, 2010.

[101] Yang Zhang. I-tasser server for protein 3d structure prediction. *BMC bioinformatics*, 9(1):40, 2008.

[102] Chao Wang, Haicang Zhang, Wei-Mou Zheng, Dong Xu, Jianwei Zhu, Bing Wang, Kang Ning, Shiwei Sun, Shuai Cheng Li, and Dongbo Bu. Falcon@ home: a high-throughput protein structure prediction server based on remote homologue recognition. *Bioinformatics*, 32(3):462–464, 2015.

[103] Martin Karplus and David L Weaver. Protein folding dynamics: The diffusion-collision model and experimental data. *Protein Science*, 3(4):650–668, 1994.

[104] Ron Unger and John Moult. Genetic algorithms for protein folding simulations. *Journal of molecular biology*, 231(1):75–81, 1993.

[105] Shaojian Sun. Reduced representation model of protein structure prediction: statistical potential and genetic algorithms. *Protein Science*, 2(5):762–785, 1993.

[106] James U Bowie and David Eisenberg. An evolutionary approach to folding small alpha-helical proteins that uses sequence information and an empirical guiding fitness function. *Proceedings of the National Academy of Sciences*, 91(10):4436–4440, 1994.

[107] Thomas Dandekar and Patrick Argos. Potential of genetic algorithms in protein folding and protein engineering simulations. *Protein Engineering*, 5(7):637–645, 1992.

[108] Thomas Dandekar and Patrick Argos. Folding the main chain of small proteins with the genetic algorithm. *Journal of Molecular Biology*, 236(3):844–861, 1994.

[109] Thomas Dandekar and Patrick Argos. Identifying the tertiary fold of small proteins with different topologies from sequence and secondary structure using the genetic algorithm and extended criteria specific for strand regions. *Journal of Molecular Biology*, 256(3):645–660, 1996.

[110] Shaojian Sun, Paul D Thomas, and Ken A Dill. A simple protein folding algorithm using a binary code and secondary structure constraints. *Protein Engineering*, 8(8):769–778, 1995.

[111] Jan T Pedersen and John Moult. Ab initio structure prediction for small polypeptides and protein fragments using genetic algorithms. *PROTEINS: Structure, Function, and Bioinformatics*, 23(3):454–460, 1995.

[112] Igor Berenboym and Mireille Avigal. Genetic algorithms with local search optimization for protein structure prediction problem. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1097–1098. ACM, 2008.

[113] Neal Lesh, Michael Mitzenmacher, and Sue Whitesides. A complete and effective move set for simplified protein folding. In *Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 188–195. ACM, 2003.

[114] Heikki Maaranen, Kaisa Miettinen, and Marko M Mäkelä. Quasi-random initial population for genetic algorithms. *Computers & Mathematics with Applications*, 47(12):1885–1895, 2004.

[115] Gunnar W Klau, Neal Lesh, Joe Marks, and Michael Mitzenmacher. Human-guided tabu search. In *AAAI/IAAI*, pages 41–47, 2002.

[116] Hans-Joachim Böckenhauer, Abu Zafer M Dayem Ullah, Leonidas Kapsokalivas, and Kathleen Steinhöfel. A local move set for protein folding in triangular lattice models. In *International Workshop on Algorithms in Bioinformatics*, pages 369–381. Springer, 2008.

[117] Ivan Kondov and Rüdiger Berlich. Protein structure prediction using particle swarm optimization and a distributed parallel approach. In *Proceedings of the 3rd workshop on Biologically inspired algorithms for distributed systems*, pages 35–42. ACM, 2011.

[118] Nashat Mansour, Fatima Kanj, and Hassan Khachfe. Particle swarm optimization approach for protein structure prediction in the 3d hp model. *Interdisciplinary sciences, computational life sciences*, 4(3):190, 2012.

[119] Md Tamjidul Hoque, Madhu Chetty, Andrew Lewis, and Abdul Sattar. Twin removal in genetic algorithms for protein structure prediction using low-resolution model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(1):234–245, 2011.

[120] Trent Benjamin Higgs, Bela Stantic, Md Tamjidul Hoque, and Abdul Sattar. Applying feature-based resampling to protein structure prediction. In *Proc. BiCoB*, pages 239–244, 2012.

[121] Shih-Chieh Su, Cheng-Jian Lin, and Chuan-Kang Ting. An effective hybrid of hill climbing and genetic algorithm for 2d triangular protein structure prediction. *Proteome science*, 9(1):S19, 2011.

[122] Manuel Cebrián, Ivan Dotú, Pascal Van Hentenryck, and Peter Clote. Protein structure prediction on the face centered cubic lattice by local search. In *AAAI*, volume 8, pages 241–246, 2008.

[123] Swakkhar Shatabda, MA Newton, Duc Nghia Pham, and Abdul Sattar. Memory-based local search for simplified protein structure prediction. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 345–352. ACM, 2012.

[124] Eyal Halm and Niels Bohrweg. Genetic Algorithm for Predicting Protein Folding in the 2D HP Model A Parameter Tuning Case Study. *Population (English Edition)*, 2007.

[125] Jan T Pedersen and John Moult. Genetic algorithms for protein structure prediction. *Current Opinion in Structural Biology*, 6:227–231, 1996.

[126] J T Pedersen and J Moult. Protein folding simulations with genetic algorithms and a detailed molecular description. *Journal of molecular biology*, 269(2):240–59, 1997.

[127] Md Tamjidul Hoque, Madhu Chetty, and Laurence S Dooley. A new guided genetic algorithm for 2d hydrophobic-hydrophilic model to predict protein folding. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 259–266. IEEE, 2005.

[128] T. Hoque, M. Chetty, and L.S. Dooley. A Guided Genetic Algorithm for Protein Folding Prediction Using 3D Hydrophobic-Hydrophilic Model. *2006 IEEE International Conference on Evolutionary Computation*, pages 2339–2346, 2006.

[129] Yi-Yao Huang. *Protein Folding Prediction with Genetic Algorithms*. PhD thesis, National Sun Yat-sen University, 2004.

[130] Swakkhar Shatabda, MA Hakim Newton, Mahmood A Rashid, and Abdul Sattar. An efficient encoding for simplified protein structure prediction using genetic algorithms. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1217–1224. IEEE, 2013.

[131] Mahmood A Rashid, MA Hakim Newton, Md Tamjidul Hoque, Swakkhar Shatabda, Duc Nghia Pham, and Abdul Sattar. Spiral search: a hydrophobic-core directed local search for simplified psp on 3d fcc lattice. *BMC bioinformatics*, 14(2):S16, 2013.

[132] JACEK BŁAŻEWICZ, KEN DILL, PIOTR ŁUKASIAK, and MACIEJ MIŁOSTAN. A tabu search strategy for finding low energy structures of proteins in hp-model. *Computational Methods in Science and Technology*, 10(1):7–19, 2004.

[133] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[134] Alena Shmygelska, Rosalia Aguirre-Hernandez, and Holger H Hoos. An ant colony optimization algorithm for the 2d hp protein folding problem. In *International Workshop on Ant Algorithms*, pages 40–52. Springer, 2002.

[135] Md Kamrul Islam. *Memetic approach for prediction of low resolution protein structures using lattice models*. PhD thesis, Monash University. Faculty of Information Technology. Gippsland School of Information Technology, 2011.

[136] Md Kamrul Islam and Madhu Chetty. Novel memetic algorithm for protein structure prediction. In *Australasian Joint Conference on Artificial Intelligence*, pages 412–421. Springer, 2009.

[137] Md Islam, Madhu Chetty, AZMD Ullah, and Kathleen Steinhöfel. A memetic approach to protein structure prediction in triangular lattices. In *Neural Information Processing*, pages 625–635. Springer, 2011.

[138] Md Kamrul Islam and Madhu Chetty. Clustered memetic algorithm for protein structure prediction. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[139] Md Kamrul Islam and Madhu Chetty. Clustered memetic algorithm with local heuristics for ab initio protein structure prediction. *IEEE Transactions on Evolutionary Computation*, 17(4):558–576, 2013.

[140] Jyh-Jong Tsay and Shih-Chieh Su. An effective evolutionary algorithm for protein folding on 3d fcc hp model by lattice rotation and generalized move sets. *Proteome science*, 11(1):S19, 2013.

[141] Mahmood A Rashid, MA Newton, Md Tamjidul Hoque, and Abdul Sattar. Mixing energy models in genetic algorithms for on-lattice protein structure prediction. *BioMed research international*, 2013, 2013.

[142] Colin Kern and Li Liao. Lattice models with asymmetric propensity matrices for locationally informed protein structure prediction. In *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*, pages 90–93. IEEE, 2013.

[143] A Dayem Ullah, Leonidas Kapsokalivas, Martin Mann, and Kathleen Steinhöfel. Protein folding simulation by two-stage optimization. In *International Symposium on Intelligence Computation and Applications*, pages 138–145. Springer, 2009.

[144] Natalio Krasnogor, BP Blackburne, Edmund K Burke, and Jonathan D Hirst. Multimeme algorithms for protein structure prediction. In *International Conference on Parallel Problem Solving from Nature*, pages 769–778. Springer, 2002.

[145] David A Pelta and Natalio Krasnogor. Multimeme algorithms using fuzzy logic based memes for protein structure prediction. In *Recent advances in memetic algorithms*, pages 49–64. Springer, 2005.

[146] Chris Thachuk, Alena Shmygelska, and Holger H Hoos. A replica exchange monte carlo algorithm for protein folding in the hp model. *BMC bioinformatics*, 8(1):342, 2007.

[147] Alexandru-Adrian Tantar, Nouredine Melab, and El-Ghazali Talbi. A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 12(12):1185–1198, 2008.

[148] Vincenzo Cutello, Giuseppe Nicosia, Mario Pavone, and Jonathan Timmis. An immune algorithm for protein structure prediction on lattice models. *IEEE transactions on evolutionary computation*, 11(1):101–117, 2007.

[149] Martin Mann, Sebastian Will, and Rolf Backofen. Cpsp-tools–exact and complete algorithms for high-throughput 3d lattice protein studies. *BMC bioinformatics*, 9(1):230, 2008.

[150] Ivan Dotu, Manuel Cebrian, Pascal Van Hentenryck, and Peter Clote. On lattice protein structure prediction revisited. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(6):1620–1632, 2011.

[151] Hsiao-Ping Hsu, Vishal Mehra, Walter Nadler, and Peter Grassberger. Growth algorithms for lattice heteropolymers at low temperatures. *The Journal of chemical physics*, 118(1):444–451, 2003.

[152] Rolf Backofen and Sebastian Will. A constraint-based approach to fast and exact structure prediction in three-dimensional protein models. *Constraints*, 11(1):5–30, 2006.

[153] Sandhya P Dubey, Gopalakrishna N Kini, Sathish M Kumar, S Balaji, Sumana MP Bhat, and Harshad R Kavathiyal. A novel conformation generation framework for de novo protein structure prediction using hydrophobic-polar model. *Asian Journal of Biochemistry*, 11(3), 2016.

[154] Swakkhar Shatabda, MA Hakim Newton, Mahmood A Rashid, Duc Nghia Pham, and Abdul Sattar. The road not taken: retreat and diverge in local search for simplified protein structure prediction. *BMC bioinformatics*, 14(2):S19, 2013.

[155] Sayantan Mandal and Nanda Dulal Jana. Protein structure prediction using 2d hp lattice model based on integer programming approach. In *Proceedings of 2012 International Congress on Informatics, Environment, Energy and Applications*, pages 17–18, 2012.

[156] Hyun-suk Yoon. *Optimization approaches to protein folding*. PhD thesis, Georgia Institute of Technology, 2006.

[157] Md Tamjidul Hoque, Madhu Chetty, Andrew Lewis, Abdul Sattar, and Vicky M Avery. Dfs-generated pathways in ga crossover for protein structure prediction. *Neurocomputing*, 73(13):2308–2316, 2010.

[158] Thomas C Beutler and Ken A Dill. A fast conformational search strategy for finding low energy structures of model proteins. *Protein Science*, 5(10):2037–2043, 1996.

[159] Thang N Bui and Gnanasekaran Sundarraj. An efficient genetic algorithm for predicting protein tertiary structures in the 2d hp model. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 385–392. ACM, 2005.

[160] Pasquale Salza, Filomena Ferrucci, and Federica Sarro. elephant56: Design and implementation of a parallel genetic algorithms framework on hadoop mapreduce. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 1315–1322. ACM, 2016.

[161] Sir Frank Macfarlane Burnet et al. The clonal selection theory of acquired immunity. 1959.

[162] Alena Shmygelska and Holger H Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC bioinformatics*, 6(1):30, 2005.

[163] Arnold L Patton, William F Punch III, and Erik D Goodman. A standard ga approach to native protein conformation prediction. In *ICGA*, pages 574–581, 1995.

[164] Andreas Alexander Albrecht, Alexandros Skaliotis, and Kathleen Steinhöfel. Stochastic protein folding simulation in the three-dimensional hp-model. *Computational Biology and Chemistry*, 32(4):248–255, 2008.

Appendix A

Supplementary Tables

Table A.1: Average time (AS) for 5000 genetic operation using different algorithms.

Sequence	GA	PGA-IM	PGA-GM	IA	PSO	ACO
S27-2	0.22872	1.84372	1.0483	1.0483	1.0483	306.7157
S27-3	4.05338	7.55668	0.36844	0.3684	0.36844	274.1049
S27-4	6.04402	4.62814	11.67556	11.676	11.67556	395.1094
S27-5	0.23732	1.09966	0.9981	0.9981	0.9981	330.4955
S27-6	6.50526	4.9391	8.4034	8.4034	8.4034	345.8957
S27-7	7.04414	3.29914	7.40184	7.4018	7.40184	333.2862
S27-8	5.38876	0.98846	0.84968	0.8497	0.84968	224.273
S27-9	2.58402	2.797	6.53052	6.5305	6.53052	328.9415
S27-10	7.49116	4.0719	10.27574	10.276	10.27574	381.5363

Table A.2: Average time (AS) for 10000 genetic operation using different algorithms.

Sequence	GA	PGA-IM	PGA-GM	IA	PSO	ACO
S48-1	1.11384	2.15876	10.61186	715.79	123.8914	472.093
S48-2	4.58684	4.72002	126.79692	553.47	10.8961	578.582
S48-3	1.0809	3.5095	7.66898	314.66	107.9051	2917.597
S48-4	6.96378	7.00798	17.60004	311.93	105.1294	3391.136
S48-5	1.13172	4.46694	6.4812	175.21	117.1708	3215.338
S48-6	1.17684	4.18524	6.67534	546.06	142.0506	1589.532
S48-7	1.78526	4.24546	11.31568	249.33	147.4582	3358.413
S48-8	3.239	6.7699	0.9844	256.57	136.2474	3308.206
S48-9	1.16476	4.62698	13.16068	355.5	139.0251	949.432
S48-10	3.40942	3.41216	3.30178	391.97	114.284	3447.318

Table A.3: Benchmark fitness score (F) of S27 sequences and comparison of the calculated best score (BS) and lowest generation number to achieve BS out of 50 runs, average score (AS) and average time (AT) for 5000 generations. The data is collected for GA (GA-2, GA-5, GA-10), PGA-IM (PGA-IM-2, PGA-IM-5, PGA-IM-10), PGA-GM (PGA-GM-2, PGA-GM-5, PGA-GM-10), in case of 2,5, 10 parent individuals.

Sr. No.	F	BS	BG	AS	AT	BS	BG	AS	AT	BS	BG	AS	AT
Sequence		GA-2				GA-5				GA-10			
S27-1	9	9	741	8.30	0.259	9	172	8.38	0.535	9	174	8.28	1.000
S27-2	10	10	761	9.02	0.229	10	416	9.22	0.494	10	129	9.26	0.945
S27-3	8	8	589	7.70	4.053	8	162	7.78	8.933	8	125	7.68	3.967

Continue on next page

S27-4	15	15	900	13.94	6.044	15	357	14.24	4.233	15	252	14.16	3.577
S27-5	8	8	123	7.38	0.237	8	85	7.40	0.531	8	51	7.26	0.940
S27-6	11	11	868	10.02	6.505	12	286	10.12	2.682	12	233	10.22	5.799
S27-7	13	13	1326	11.90	7.044	13	518	11.92	2.107	13	267	11.86	3.119
S27-8	4	4	1	4.00	5.389	4	86	4.00	3.249	4	14	4.00	1.142
S27-9	7	7	358	6.48	2.584	7	72	6.48	3.968	7	86	6.52	4.789
S27-10	11	11	709	10.32	7.491	11	394	10.24	6.221	11	116	10.52	11.300
Sequence		PGA-IM-2				PGA-IM-5				PGA-IM-10			
S27-1	9	9	10	8.40	0.462	9	27	8.34	0.562	9	4	8.38	1.623
S27-2	10	10	4	9.20	1.844	10	8	9.18	0.919	10	8	9.20	1.844
S27-3	8	8	9	7.78	7.557	8	5	7.20	5.039	8	2	7.90	5.659
S27-4	15	15	16	14.14	4.628	15	9	14.10	3.797	15	4	14.04	5.917
S27-5	8	8	3	7.40	1.100	8	9	7.32	0.463	8	2	7.52	1.872
S27-6	11	11	17	10.04	4.939	11	8	10.30	4.140	11	3	10.24	4.349
S27-7	13	13	25	12.18	3.299	13	13	12.04	2.918	13	6	12.20	3.875
S27-8	4	4	2	4.00	0.988	4	2	4.00	1.798	4	4	4.00	0.414
S27-9	7	7	10	6.48	2.797	7	4	6.58	2.277	7	4	6.70	3.047
S27-10	11	11	9	10.28	4.072	11	6	10.48	3.076	11	3	10.62	4.068
Sequence		PGA-GM-2				PGA-GM-5				PGA-GM-10			
S27-1	9	9	129	8.28	0.555	9	86	8.34	0.695	9	42	8.42	0.903
S27-2	10	10	239	8.88	1.048	10	96	8.98	1.199	10	35	9.36	1.669
S27-3	8	8	40	7.28	0.368	8	19	7.78	0.436	8	15	7.70	0.820
S27-4	15	15	235	13.38	11.676	15	94	13.92	11.284	15	40	14.12	11.830
S27-5	8	8	46	7.04	0.998	8	30	7.34	1.162	8	7	7.40	2.106
S27-6	11	11	171	9.82	8.403	11	49	10.10	9.523	11	30	10.24	9.769
S27-7	13	13	68	11.68	7.402	13	140	11.80	8.156	13	51	12.06	8.435

Continue on next page

S27-8	4	4	1	1.00	0.850	4	1	4.00	1.127	4	1	4.00	1.575
S27-9	7	7	64	6.22	6.531	7	29	6.48	11.227	7	10	6.74	13.093
S27-10	11	11	87	10.04	10.276	11	80	10.32	9.993	11	40	10.34	12.667

Table A.4: Benchmark fitness score (F) of S48 sequences and comparison of the calculated highest score (HS), average score (AS) and average time (AT) for 10000 generations. The data is collected for GA (GA-2, GA-5, GA-10), PGA-IM (PGA-IM-2, PGA-IM-5, PGA-IM-10), PGA-GM (PGA-GM-2, PGA-GM-5, PGA-GM-10), in case of 2,5, 10 parent individuals.

Sr. No.	F	BS	AS	AT	BS	AS	AT	BS	AS	AT
Sequence		GA-2			GA-5			GA-10		
S48-1	32	28	24.36	1.114	28	24.76	3.315	27	25.00	7.625
S48-2	34	27	24.52	4.587	30	24.76	7.844	28	25.40	12.418
S48-3	34	29	25.18	1.081	30	25.14	2.874	31	25.50	6.926
S48-4	33	28	24.40	6.964	28	24.96	11.571	28	24.90	11.455
S48-5	32	29	24.42	1.132	29	24.94	2.707	28	25.00	7.770
S48-6	32	27	24.08	1.177	28	23.94	3.159	28	24.50	8.502
S48-7	32	29	24.00	1.785	27	24.30	4.644	29	24.70	10.681
S48-8	31	28	24.00	3.239	27	24.60	5.261	28	24.10	8.392
S48-9	34	28	25.26	1.165	30	25.36	4.810	29	25.60	10.918
S48-10	33	28	24.48	3.409	29	24.42	5.852	28	25.00	9.119
Sequence		PGA-IM-2			PGA-IM-5			PGA-IM-10		
S48-1	32	28	25.10	2.159	29	25	0.562	29	24.80	13.994
S48-2	34	29	25.12	4.720	29	25	0.919	29	25.90	17.112

Continue on next page

S48-3	34	29	25.40	3.510	29	25	5.039	30	25.68	13.481
S48-4	33	28	25.04	7.008	30	25	3.797	28	25.20	18.909
S48-5	32	28	25.18	4.467	27	24	0.463	29	25.26	13.605
S48-6	32	27	23.98	4.185	27	24	4.140	27	24.36	15.859
S48-7	32	28	24.58	4.245	27	25	2.918	28	24.42	18.515
S48-8	31	28	24.08	6.770	27	25	1.798	28	24.68	13.408
S48-9	34	29	25.74	4.627	29	26	2.278	29	26.00	16.654
S48-10	33	28	24.12	3.412	30	25	3.076	28	24.60	14.287
Sequence		PGA-GM-2			PGA-GM-5			PGA-GM-10		
S48-1	32	27	24.00	10.612	29	25.10	11.677	28	25.08	8.821
S48-2	34	28	24.40	126.797	29	25.08	116.881	29	25.52	42.653
S48-3	34	29	24.70	7.669	29	25.04	9.9204	29	25.84	6.958
S48-4	33	27	24.00	17.600	28	24.54	2.327	28	24.86	3.629
S48-5	32	29	23.50	6.481	29	24.86	7.561	29	24.72	3.331
S48-6	32	26	23.10	6.675	26	24.10	12.205	29	24.28	8.619
S48-7	32	28	23.50	11.316	27	23.86	15.178	27	24.76	11.725
S48-8	31	27	23.30	0.984	28	24.18	3.7137	28	24.18	7.787
S48-9	34	27	24.50	13.161	30	25.30	16.499	29	25.68	11.592
S48-10	33	26	23.10	3.302	29	24.30	221256	28	25.02	5.627