ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

ARTIFICIAL INTELLIGENCE BASED METHODS FOR THE SOLUTION OF PROTEIN FOLDING PROBLEM BY USING COARSE-GRAINED LATTICE AND OFF-LATTICE MODELS

Ph.D. THESIS

Berat DOĞAN

Department of Electronics and Communication Engineering

Electronics Engineering Programme

JUNE 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

ARTIFICIAL INTELLIGENCE BASED METHODS FOR THE SOLUTION OF PROTEIN FOLDING PROBLEM BY USING COARSE-GRAINED LATTICE AND OFF-LATTICE MODELS

Ph.D. THESIS

Berat DOĞAN
(504092203)

Department of Electronics and Communication Engineering

Electronics Engineering Programme

Thesis Advisor: Prof. Dr. Tamer ÖLMEZ

JUNE 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

PROTEİN KATLANMA PROBLEMİNİN ÇÖZÜMÜ İÇİN KABA-TANELİ KAFES VE KAFES-DIŞI MODELLERİ KULLANAN YAPAY ZEKA TABANLI YÖNTEMLER

DOKTORA TEZİ

Berat DOĞAN
(504092203)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Tamer ÖLMEZ

HAZİRAN 2015

Berat Doğan, a **Ph.D.** student of ITU **Graduate School of Science Engineering and Technology** student ID 504092203, successfully defended the **thesis** entitled "**ARTIFICIAL INTELLIGENCE BASED METHODS FOR THE SOLUTION OF PROTEIN FOLDING PROBLEM BY USING COARSE-GRAINED LATTICE AND OFF-LATTICE MODELS**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Tamer ÖLMEZ** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Nizamettin AYDIN** Yıldız Technical University	.....
	Prof. Dr. Mehmet KORÜREK İstanbul Technical University	.....
	Prof. Dr. Ahmet Hamdi KAYRAN İstanbul Technical University	.....
	Assist. Prof. Dr. Gökhan BİLGİN Yıldız Technical University	.....

Date of Submission : 16 March 2015
Date of Defense : 11 June 2015

To my beloved parents,

FOREWORD

I would like to thank my advisor, Tamer ÖLMEZ and his wife Zümray DOKUR ÖLMEZ, who were the first that invited me to the academic life when I was working at Netaş. I would like to thank them for their support and encouragement throughout the course of my studies and research.
I would like to thank Nizamettin AYDIN and Mehmet KORÜREK for beign in my thesis committee and providing their valueable comments for my research.
I am indebted to my mother and my father for their unconditional love and support. This work would have not been possible without their presence.
Last but not least, I would like to express my sincere thanks to my wife Hülya for her endless support at every moment. I could not have finished this work without her motivation and support. My daughter Melis, you have brightened our life with your coming.
I would also like to thank the Scientific and Technical Research Council of Turkey (TÜBİTAK) for their financial support during both in my MSc and PhD. This thesis is also supported by İTÜ BAPSO.

March 2015

Berat DOĞAN
(Electronics Engineer, MSc)

TABLE OF CONTENTS

Page

ABBREVIATIONS

ABC	**:** Artificial Bee Colony
ACO	**:** Ant Colony Optimization
AMBER	**:** Assisted Model Building with Energy Refinement
ASSRS	**:** Adaptive Step Size Random Search
CHARMM	**:** Chemistry at Hardvard Macromolecular Mechanics
DE	**:** Differential Evolution
ECEPP	**:** Empirical Conformation Energy Program for Peptides
EDA	**:** Estimation of Distribution Algorithm
ELP	**:** Energy Landscape Paving
EMC	**:** Evolutionary Monte Carlo
EN	**:** Elastic Net
FA	**:** Firefly Algorithm
FCC	**:** Face Centered Cubic
FI	**:** Farthest Insertion
GA	**:** Genetic Algorithm
GAMC	**:** Genetic Algorithm based on Matrix Coding
GAOSS	**:** Genetic Algorithm based on Optimal Secondary Structure
GROMOS	**:** Groningen Molecular Simulation
HHGA	**:** Hybrid of Hill-climbing and Genetic Algorithm
HP	**:** Hydrophobic Polar
HTS	**:** Heuristic Tabu Search
IF-ABC	**:** Internal Feedback Artificial Bee Colony
ILS	**:** Iterated Local Search
NMR	**:** Nuclear Magnetic Resonance
ORSSRS	**:** Optimized Relative Step Size Random Search
OSSRS	**:** Optimum Step Size Random Search
PDB	**:** Protein DataBank
PERM	**:** Pruned Enriched Rosenbluth Method
PS	**:** Pattern Search
PSO	**:** Particle Swarm Optimization
PSO2011	**:** Particle Swarm Optimization 2011
RS	**:** Random Search
REMC	**:** Replica Exchange Monte Carlo
SA	**:** Simulated Annealing
SMMP	**:** Simple Molecular Mechanics for Proteins
SOM	**:** Self Organizing Maps
TS	**:** Tabu Search
UniProt	**:** Universal Protein Resource
VS	**:** Vortex Search

LIST OF TABLES

Page

LIST OF FIGURES

Page

Page

ARTIFICIAL INTELLIGENCE BASED METHODS FOR THE SOLUTION OF PROTEIN FOLDING PROBLEM BY USING COARSE-GRAINED LATTICE AND OFF-LATTICE MODELS

SUMMARY

The protein folding problem is one of the most widely studied problem within the bioinformatics community. Computational methods proposed for the solution of this problem can be categorized into two main groups: Comparative modeling, and ab initio methods. Comparative modeling utilizes existing databases of experimentally determined protein structures to determine the three-dimensional structure of proteins. However, in ab initio methods three-dimensional structure of proteins are determined from solely their amino acid sequences. In the ab initio methods, a number of potential energy functions with different resolutions (including the simple coarse-grained methods and the detailed all-atom models) are proposed to model the interactions that occur among the amino acid molecules of the proteins. A search method is then used to thoroughly explore the energy landscape of the defined potential energy function to find the optimum fold of a protein.

In this thesis, new possibilities are searched to find an effective way of improving the search abilities for ab initio methods. Within this scope, both the coarse-grained and all-atom models are studied to determine the protein structures.

Coarse-grained methods studied in this thesis include the simplified lattice and off-lattice models. For the hydrophobic polar (HP) lattice model, a new state-space representation of the protein folding problem is proposed for the use of reinforcement learning methods. The proposed state-space representation reduces the dependency of the size of the state-action space to the amino acid sequence length. The proposed method also introduces the concept of "learning" for the protein folding problem in two-dimensional HP model. Thus, at the end of a learning process optimum fold of any sequence of a particular length can be found which is not the case in the existing methods. Moreover, by utilizing a swarm based reinforcement method (Ant-Q algorithm) the optimal fold is found rapidly when compared to the most widely used reinforcement learning algorithm, the Q-learning algorithm.

For the off-lattice AB model, a new optimization algorithm, the Vortex Search (VS) algorithm, is proposed to minimize the energy function of this model. The proposed VS algorithm tested on a benchmark numerical function set and it is shown that it performs quite well when compared to the well known optimization algorithms. Another contribution of the thesis presented for the off-lattice AB model deals with the energy function of this model. The energy landscape of the off-lattice AB model leads the algorithms to easily trap into local minimum points. In literature, to escape from local minimum points, usually a combination of the well known optimization algorithms or some extensions of these algorithms are proposed. However, in this thesis rather than an algorithmic improvement, a more smoothed energy landscape is

provided for the algorithms by modifying the energy function of the off-lattice AB model.

The all-atom model studied in the thesis is based on the ECEPP force field which is combined to the VS algorithm in conjuction with the SMMP software package. A number of proteins are selected from the PDB database to evaluate the performance of the proposed method results of which indicate that the proposed method is comparable to the existing methods.

PROTEİN KATLANMA PROBLEMİNİN ÇÖZÜMÜ İÇİN KABA-TANELİ KAFES VE KAFES-DIŞI MODELLERİ KULLANAN YAPAY ZEKA TABANLI YÖNTEMLER

ÖZET

Proteinler organizmadaki bütün biyolojik süreçlerde çok önemli işlevler üstlenmektedir. Genetik bilgiden hareketle, proteinlerin bu işlevsel yapılarının nasıl sentezlendiği uzun yıllardır bilinmesine rağmen, sentezlenme işlemi sonucunda proteinlerin kendilerine özgü üç boyutlu fonksiyonel yapılarının nasıl oluştuğu hala bilinmemektedir. Uzun yıllardır cevabı aranan bu probleme literatürde "protein katlanma problemi" adı verilmektedir.

Protein katlanma problemi ilk kez Levinthal tarafından 1960'lı yıllarda ortaya atılmıştır. Levinthal'ın çalışmasından önce, proteinlerin bir takım rastgele yapılardan geçerek doğal yapılarına ulaştıkları düşünülmekteydi. Levinthal ise çalışmasında proteinlerin çok daha sistematik bir yapıda katlandığını belirtmiştir. Çünkü ona göre rastgele yapılardan hareketle proteinlerin katlanabilmesi için pratikte mümkün olamayacak kadar çok olasılığın denenmesi gerekmekteydi. Bu basit çıkarım, sonraları bilim insanlarının protein katlanma problemine başka bir açıdan bakmalarına sebep olmuştur.

Protein katlanma problemi ile ilgili bir diğer önemli gelişme, Anfinsen'in bir proteinin üç boyutlu yapısının aminoasit dizilimiyle belirlendiğini deneysel olarak göstermesidir. Anfinsen'in bu çalışmasından hareketle proteinin üç boyutlu doğal yapısının minimum serbest enerjili yapı olduğu belirtilmektedir.

Protein katlanma problemi üzerinde bu kadar çok uğraşılmasının şüphesiz önemli nedenleri bulunmaktadır. Bir proteinin biyolojik olarak aktif veya fonksiyonel olabilmesi için mutlaka doğal yapısına katlanması gerekmektedir. Örneğin bazı mutasyonlar proteinlerin doğal yapılarına katlanmasını engelleyebilmektedir. Böyle bir durumda proteinler doğru bir şekilde katlanamamaktadır ve bu ise beraberinde bazı hastalıkların oluşmasına neden olmaktadır. Bazı durumlarda ise mutasyon olmaksızın proteinler yanlış katlanabilmektedir. Örneğin insan vücudunda bulunan amyloid-β proteinin yanlış katlanması Alzheimer hastalığının klinik belirtilerine neden olmaktadır. Benzer şekilde, Huntingdon ve Parkinson hastalıkları da proteinlerin yanlış katlanması sonucu oluşan hastalıklardır. Protein katlanma probleminin çözülmesi bu gibi hastalıkların tedavisine yönelik hedef ilaçların geliştirilmesi açısından oldukça önemlidir.

Günümüzde proteinlerin üç boyutlu doğal yapıları NMR (nükleer manyetik rezonans) ve X-Işını kristolografisi gibi teknolojiler kullanılarak tespit edilebilmektedir. Fakat bu yöntemler oldukça zaman alıcı ve pahalı yöntemlerdir. Dahası, X-Işını kristolografisi ile proteinlerin üç boyutlu yapısını tespit edebilmek için proteinlerin düzgün sıralanmış kristaller oluşturması gerekmektedir ki bu bütün proteinlerin sahip olduğu bir özellik değildir. NMR teknolojisi ile proteinlerin üç boyutlu yapısını tespit edebilmek için ise, proteinlerin çözülebilir olması

gerekmektedir ve bu yöntemle büyük proteinlerin yapısı çoğunlukla tespit edilememektedir. Deneysel yöntemlerdeki mevcut zorluklardan dolayı, aminoasit dizilimi belirlenmiş protein sayısı ile üç boyutlu yapıları deneysel olarak belirlenmiş protein sayısı arasındaki uçurum her geçen gün artmaktadır. Bu farkı kapatmak için deneysel yöntemlere alternatif olarak bir takım yöntemlere ihtiyaç duyulduğu aşikardır. Bilim insanları bu gerçekten yola çıkarak, hesapsal yöntemlerle bir proteinin aminoasit diziliminden üç boyutlu doğal yapısını belirlemeye yönelik yöntemler öne sürmüşlerdir.

Literatürdeki mevcut hesapsal yöntemler, "Karşılaştırmalı Modelleme" ve "Ab Initio (herhangi bir bilgi olmadan başlama)" olmak üzere iki ana grup altında incelenebilir. Karşılaştırmalı modelleme yöntemleri proteinlerin üç boyutlu yapılarını tespit etmek için yapısı deneysel olarak belirlenmiş proteinlerden faydalanır. Karşılaştırmalı modelleme yöntemlerinden olan homoloji modellemede, benzer aminoasit dizilimine sahip proteinlerin yapılarının da benzer olacağı kabulünden hareketle yola çıkılır. Bu amaçla, yapısı belirlenmek istenen bir proteine, yapısı deneysel olarak belirlenmiş proteinler içerisinden aminoasit dizilimleri en çok benzeyenler (ilgili proteinin homologu olanlar) bulunur. Buradan hareketle ilgili proteinin yapısı tahmin edilir. Benzer şekilde bir diğer karşılaştırmalı modelleme yöntemi olan iş parçası modeli (threading) yönteminde, yapısı bilinen proteinlerin sahip olduğu birtakım ortak üç boyutlu yapılardan (fold) hareketle herhangi bir proteinin üç boyutlu yapısı bulunmaya çalışılır. Bu ortak üç boyutlu yapıların aminoasit dizileri ile yapısı bulunmaya çalışılan proteinin aminoasit diziliminin örtüştüğü yerler tespit edilir ve buradan hareketle ilgili proteinin üç boyutlu yapısı bulunmaya çalışılır. Karşılaştırmalı modelleme yöntemleri iyi sonuçlar vermesine rağmen, birçok proteinin bir homolog proteine sahip olmaması ve aminoasit dizilimleri benzemesine rağmen proteinlerin farklı üç boyutlu yapılara sahip olabilmelerinden ötürü çoğu zaman bu yöntemler yetersiz kalmaktadır. Ab initio yöntemlerinde ise yapısı deneysel olarak bulunmuş proteinlerden faydalanılmaz ve herhangi bir proteinin üç boyutlu yapısı yalnızca aminoasit diziliminden hareketle bulunmaya çalışılır. Ab initio yöntemleri bu anlamda karşılaştırmalı modelleme yöntemlerinden ayrılır.

Ab initio yöntemlerinde, proteinlerin üç boyutlu doğal yapısının minimum serbest enerjili yapı olduğu kabulünden hareketle, birtakım enerji fonksiyonları türetilmekte ve protein katlanma süreci bu enerji fonksiyonları yardımıyla modellenmeye çalışılmaktadır. Literatürde bu amaçla geliştirilen modeller kaba-taneli (coarse-grained veya düşük çözünürlüklü) ve tüm-atom modelleri olmak üzere iki ana grup altında incelenebilir. Kaba-taneli modellerde bir proteine ait herbir aminoasit sadece tek bir atommuş gibi düşünülerek problem çözülmeye çalışılmaktadır. Bu modeller, tüm-atom modellerine göre daha yaklaşık modeller olmasına rağmen hesapsal açıdan hızlı oldukları için kullanılmaktadırlar. Tüm-atom modelleri, adından da anlaşılacağı üzere proteine ait aminoasitlerin bütün atomlarını göz önünde bulunduran modellerdir. Bu modeller, kaba-taneli modellere göre daha gerçekçi olmalarına rağmen hesapsal açıdan dezavantajlıdır. Öyle ki, bir proteinin tüm-atom modelleri ile üç boyutlu yapısının bulunması işlemi günler, hatta aylar boyunca sürebilmektedir.

Bu tezin ana çerçevesi kaba-taneli yöntemleri içermekle birlikte tezde tüm-atom modellerine ilişkin çalışmalar da yapılmıştır. Tez kapsamında kaba-taneli modellerden, literatürde çok bilinen kafes HP modeli ve kafes-dışı AB model çalışılmıştır. Tüm-atom modeli olarak ise ECEPP kuvvet alanını gerçekleyen model çalışılmıştır.

Kafes HP modeli hidrofobik etkinin protein katlanmasında büyük rol üstlendiği gerçeğinden hareketle önerilmiştir. Bu nedenle bu modelde aminoasitler, suyu sevmeyen (hidrofobik) ve suyu seven (polar) aminoasitler olmak üzere ikiye ayrılmıştır. Hidrofobik aminoasitlerin globüler proteinlerin üç boyutlu yapılarında çoğunlukla iç bölgelerde bulunma eğiliminde oldukları bilinmektedir. Bu bilgiden hareketle HP-model, suyu sevmeyen aminoasitleri protein iç bölgesine, suyu seven aminoasitleri dış bölgeye hareket etmeye zorlayan bir model olarak karşımıza çıkmaktadır.

Kafes-dışı AB-modeli, kafes HP modeline oldukça benzemekle birlikte farklı olarak bu modelde aminoasitler arası açı değerleri [-180, 180] aralığında değerler alabilmektedir. Yani kafes HP modelinden farklı olarak, bu modelde sürekli uzayda çalışılmaktadır. Bu ise protein yapısının daha doğru bir şekilde bulunmasına imkan tanımaktadır.

ECEPP kuvvet alanı, literatürdeki büyük ölçekli kuvvet alanlarına kıyasla daha basit bir kuvvet alanıdır. Kuvvet alanları, bir sistemin benzetimini yaparken enerji fonksiyonunu türetmede kullanılan parametrelerin ve eşitliklerin bütünü olarak düşünülebilir. ECEPP kuvvet alanında, moleküllerin sahip olduğu kovalent bağ uzunlukları ve bağ açıları dengedeki değerlerinde sabit kabul edilip sadece dihedral açıları bulunmaya çalışılmaktadır.

Tez kapsamında, kafes HP modelini kullanarak protein katlanma probleminin çözümüne yönelik takviyeli öğrenmeye dayalı bir yöntem önerilmiştir. Literatürde bir çok farklı yöntemle kafes HP modeli kullanılarak protein katlanma problemi çözülmeye çalışılmıştır. Fakat takviyeli öğrenmeye dayalı yöntemlerin kullanımı oldukça yenidir. Literatürde bu problemin çözümüne yönelik önerilen takviyeli öğrenme yöntemlerinin bazı sakıncaları vardır. Bu tez çalışmasında önerilen yeni bir durum uzayı sayesinde bu sakıncalar giderilmiştir. Ayrıca sürü zekasına dayalı bir takviyeli öğrenme yöntemi (Ant-Q) kullanılarak, literatürde önerilen yönteme kıyasla çok daha hızlı bir şekilde sonuca ulaşılmaktadır.

Tez kapsamında, kafes-dışı AB model ile kullanılmak üzere, yeni bir sürekli optimizasyon algoritması geliştirilmiştir. Önerilen yeni optimizasyon algoritması Girdap Arama algoritması adıyla literatüre kazandırılmıştır. Girdap Arama algoritması zengin bir matematiksel fonksiyon kümesi üzerinde denenmiş ve oldukça başarılı sonuçlar alınmıştır. Aynı algoritmadan kafes-dışı AB model ile birlikte protein katlanma probleminin çözümü için de faydalanılmıştır. Tez kapsamında kafes-dışı AB model için önerilen bir diğer yenilik, bu algoritmanın enerji fonksiyonu ile ilgilidir. Kafes-dışı AB modelin mevcut enerji fonksiyonu çok fazla yerel minimum noktaya sahip olduğundan algoritmalar bu yerel minimum noktalara kolayca takılabilmektedir. Tez kapsamında mevcut enerji fonksiyonuna yapılan bir modifikasyonla bu problemin önüne geçilmeye çalışılmıştır.

Tüm-atom modelinde kullanılan ECEPP kuvvet alanı da sürekli bir enerji fonksiyonuna sahip olduğundan, yine Girdap Arama algoritması kullanılarak proteinlerin üç boyutlu yapıları bulunmaya çalışılmıştır. Bu amaçla PDB veri tabanından elde edilen peptidlerin üç boyutlu yapıları aminoasit dizilimlerinden hareketle bulunmaya çalışılmıştır. Elde edilen sonuçlar, deneysel olarak elde edilen yapılarla karşılaştırılmış ve sonuçların mevcut hesapsal yöntemlerle kıyaslanabilir düzeyde olduğu gözlemlenmiştir.

1. INTRODUCTION

1.1 The Protein Folding Problem

Proteins are among the most important macromolecules in all living organisms. They play a vital role in most of the activities within cells of living organisms some of which are listed below [1]:

- Proteins are passive building blocks of many biological structures, such as the coats of viruses, the cellular cytoskeleton, the keratin in our skin or the collagen in our bones and cartilages;
- They transport and store other species, from oxygen or electrons to macromolecules;
- They act as hormones, transmit information and signals between cells and organs;
- They act as antibodies, defend the organism against intruders;
- They are the essential component of muscles, converting chemical energy into mechanical one, and allowing the animals to move and interact with the environment;
- They control the passage of species through the membranes of cells and organelles, they are doorkeepers;
- They control gene expression;
- They are the essential agents in the transcription of the genetic information into more protein;
- As chaperones, they protect other proteins to help them to acquire their functional three-dimensional (3D) structure via the folding process.

Proteins are basically sequences of amino acids that chain together via peptide bonds. Therefore, proteins are also known as polypeptides. Once synthesized, all proteins fold into a unique three-dimensional structure which enables them to perform some biological tasks as exampled above. It is known that, the resulting folds (three-dimensional structures) of the proteins are the minimum free energy conformations.

However, it is not known how a protein can choose the minimum energy fold among all possible folds. This process is known as the protein folding process (or problem) and it is one of the most widely studied problem within the bioinformatics community.

Genomic projects are providing us with the linear amino acid sequence of hundreds of thousands of proteins. If only we could learn how each and every one of these folds in three-dimensions we would have the complete part list of an organism and could face the challenge of understanding how these parts assemble in a cell. This is not only an intellectual challenge but it has also enormous practical implications [2]. For example, most of the drugs interact with faulty or foreign proteins to prevent them performing their functions. Faulty proteins are those which are not folded correctly. These misfolded proteins can have serious effects, including many well known diseases, such as Alzheimer's, Mad Cow (BSE), and Parkinson's disease. The drugs that we use to treat these diseases might not be aimed at the best target. Some other biologically relevant proteins can be better targets for a certain disease. Thus, a better understanding of the protein structures can provide valuable information for us to design exact drugs theoretically on a computer without a great deal of experimentation.

Experimental methods such as X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy are currently used to determine three-dimensional structures of the proteins. However, these methods are not only time consuming but also expensive and labor intensive [3]. Moreover, these methods have some restrictions. For example, X-ray crystallography requires the protein or the protein complex under study to form a reasonably well ordered crystal, a feature that is not universally shared by proteins. NMR spectroscopy needs proteins to be soluble and there is a limit to the size of protein that can be studied [2]. Some proteins (like membrane proteins) are not easily accessible. This situation further complicates the crystallization or solvation process. As a result of these restrictions, the gap between the experimentally resolved number of protein structures to the known protein sequences dramatically increases. In Figure 1.1, a comparison of number of known protein sequences stored in UniProt database to the number of known protein structures stored in PDB database is given for the last a few years. From this figure it is clear that, to fill up this gap some alternative to experimental methods are required.

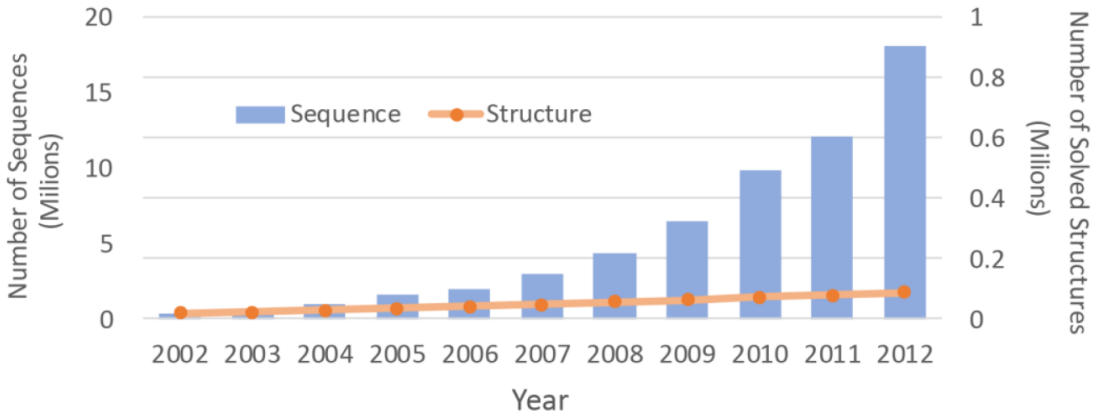

Figure 1.1 : A comparison of the number of solved protein structures to the number of known protein sequences [4].

1.2 Computational Solution Methods for the Protein Folding Problem

Computational solution methods can be categorized into two main groups: Comparative modeling, and ab initio methods. Comparative modeling utilizes existing databases of experimentally determined protein structures. This group can be further split into two main subgroups: Homology modeling, and Threading [5]. In Homology modeling it is assumed that, if two proteins have similar amino acid sequences they will also have similar 3D structures. Thus, for a given amino acid sequence, a similar sequence of an experimentally determined structure is searched. The structure of the best matching sequence is then optimized to predict the 3D structure of the corresponding amino acid sequence. Similarly, threading scans the amino acid sequence of the unknown structure against a database of experimental structures. A scoring function is evaluated for each comparison to assess the compatibility of the sequence to the structure, thereby producing plausible 3D models [5].

Comparative modeling is highly studied and it has proven to be quite efficient and applicable for a majority of proteins [6]. However, there are three main reasons that makes the ab initio methods still interesting. First of all, there still exists a number of proteins which do not show any homology with proteins of known structure. Second, comparative modeling does not offer any insight as to why a protein adopts a certain structure; and third, although some proteins show high resemblance to other proteins they still adopt different structures, which in principle means that predictions made by comparative modeling are never fully reliable [7].

The ab initio (means "from the beginning" or "to start without knowledge") or de novo method is proposed to determine the structure of the proteins from solely their amino acid sequences. This method models physical interactions of amino acids in a polypeptide chain to determine the structure. In some of these models, interactions with the surrounding solvent are also included. In the ab initio models, a potential energy function is used to model these physical interactions. The potential energy function must be accurate enough to capture the important interactions yet simple enough, so that calculations can be performed with today's computational power in real time [5]. For this purpose a number of force fields with different resolutions (including the simple coarse-grained methods and the detailed all-atom models) are proposed. All force-fields have its own energy function to be optimized to determine the native structure of a given amino acid sequence. It is accepted that, the native structure of an amino acid sequence is the configuration that minimizes the given energy function.

One of the most important development on the solution of the protein folding problem is the Anfinsen's study in which it was shown that, the information for protein folding is resided entirely within the amino acid sequence of the protein. To show this, Anfinsen first denatured the 3D structure of ribonuclease A by using the denaturant urea plus 2ME (2-mercaptoethanol). The denaturant broke the disulfide bonds of the protein and thus, the protein unfolded to a non-native structure. But once the denaturant was removed, the protein simultaneously refolded to its native structure.

Around the same period, Levinthal also focused on the protein folding problem. According to Levinthal, it was impossible for a protein to visit all possible conformations during the folding process. Because, a protein could fold very quickly and there was no time for a protein to visit all possible conformations during this limited period of time. For example, for a 150 amino acid length protein, when the protein backbone considered having three degrees of freedom, there are 3^{150} different structures to reach the global minimum. If we consider 10^{12} structures are tried in a second, a total time of $7x10^{53}$ years are still needed to try all of the structures [9]. As a result, Levinthal inferred that, a protein must follow a pathway to its native structure during the folding process.

The pathway followed by the proteins during the folding process can be considered as folding funnels in energy landscapes defined by ab initio methods. In Figure 1.2, a representative energy landscape is given. In ab initio methods, usually a search method is used to thoroughly explore the energy landscape to find the native fold within reasonable amount of time.

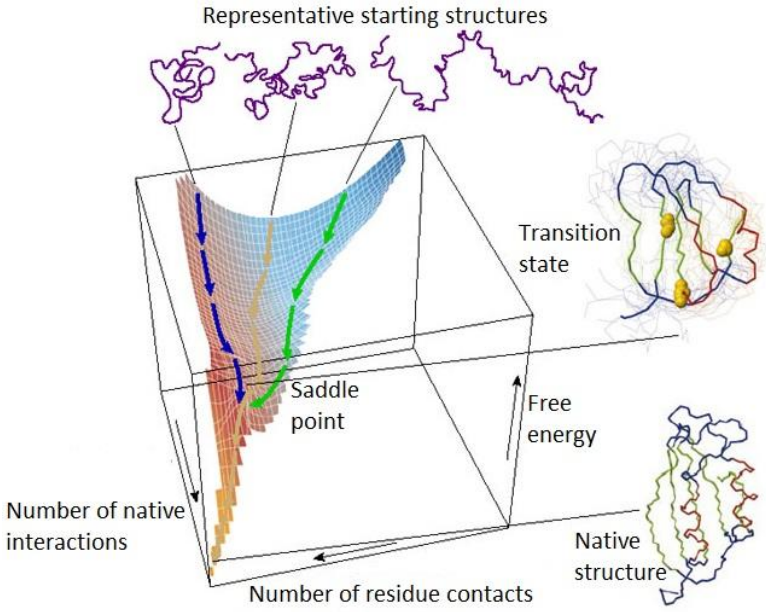

Figure 1.2 : Folding pathway of a protein on the energy landscape forms a funnel [8].

In this thesis, new possibilities are searched to find an effective way of improving the search abilities for ab initio methods. Within this scope, both the coarse-grained and all-atom models are studied to determine the protein structures.

1.2.1 Coarse-grained lattice and off-lattice models

Coarse-grained methods studied in this thesis include the simplified lattice and off-lattice models. In these models, each amino acid of a protein is represented in a binary form. Perhaps, the most widely studied model is the so called HP model [10], in which each amino acid in a protein sequence is considered as hydrophobic or polar. In the HP model, high resolution lattice models are used to accurately model the protein structure and retain the computational efficiency of lattice models as well [11]. In lattice models, each amino-acid is mapped to a particular lattice point to form a continuous and self-avoiding amino acid sequence with fixed bond lengths between successive amino acid pairs. The lattice models benefits greatly from the discretization of protein phase space; however, it also suffers from this strategy. The

discrete nature of the model surely affects the folding behaviors, especially the dynamics of the system [11]. To overcome this problem off-lattice model (or toy model) was proposed [12]. In the off-lattice model each amino acid in a protein chain is considered either A (hydrophobic) or B (polar or hydrophilic) as in HP model. In this model, again the amino acids are linked up with a fixed bond length, but different from the HP model the backbone can continuously bend between any pair of successive links. Additionally, in this model nonconsecutive amino-acids interact through a modified Leonard-Jones potential and there is an energy contribution from each bond angle between successive bonds. Therefore, when compared to the HP model, the off-lattice AB model is much more realistic.

1.2.2 All-atom models

In the all-atom models, all atomic details of a protein along with the physical interactions such as bond angle, torsion angle, van-der Waals forces, electrostatic interactions, charge transfer etc. are considered. These models are usually computationally expensive. In literature, there exist a number of well-known force fields such as AMBER [13], CHARMM [14], GROMOS [15] and ECEPP [16] etc. proposed for ab-initio protein structure prediction. In this thesis, the ECEPP force-field is utilized to determine the three-dimensional structure of the proteins from their primary amino-acid sequence. The ECEPP force-field is chosen because, it is computationally less expensive than the others and it is much more simple for us to integrate this force-field to our methods.

1.3 Contribution of the Thesis

For the HP lattice model, a new state-space representation of the protein folding problem is proposed for the use of reinforcement learning methods [17]. The proposed state-space representation reduces the dependency of the size of the state-action space to the amino acid sequence length. The proposed method also introduces the concept of "learning" for the protein folding problem in two-dimensional HP model. Thus, at the end of a learning process optimum fold of any sequence of a particular length can be found which is not the case in the existing methods. Moreover, by utilizing a swarm based reinforcement method (Ant-Q algorithm) the optimal fold is found rapidly when compared to the traditional Q-learning algorithm.

For the off-lattice AB model, a new optimization algorithm, the Vortex Search (VS) algorithm [18], is proposed to minimize the energy function of this model. The proposed VS algorithm is tested on a benchmark numerical function set and it is shown that it performs quite well when compared to the well-known optimization algorithms. Another contribution of the thesis presented for the off-lattice AB model deals with the energy function of this model. The energy landscape of the off-lattice AB model leads the algorithms to easily trap into local minimum points. In literature, to escape from local minimum points, usually a combination of the well-known optimization algorithms or some extensions of these algorithms are proposed. However, in this thesis rather than an algorithmic improvement, a more smoothed energy landscape is provided for the algorithms by modifying the energy function of the off-lattice AB model [19].

For the all atom model, the ECEPP force field used in the experiments is implemented by the VS algorithm in conjuction with the SMMP software package. A number of proteins are selected from the PDB database to evaluate the performance of the proposed method. It is shown that the proposed method is comparable to the existing methods.

1.4 Organization of the Thesis

Organization of the thesis is as follows. In Chapter-2, some basic information about the amino acids, proteins and protein structures are given. Then, the experimental methods used to determine the three-dimensional structures of the proteins are detailed and finally, the database for the experimentally resolved protein structures (the PDB database) is introduced.

In Chapter-3, first the HP lattice model is introduced and then, the newly proposed reinforcement learning based method for the solution of protein folding problem in HP lattice model is detailed.

In Chapter-4, the off-lattice AB model and the newly proposed optimization algorithm, the Vortex Search (VS) algorithm, are introduced. This chapter is concluded with the details of the modified-energy function proposed for the off-lattice AB model.

In Chapter-5, all-atom models for the protein folding problem is introduced and the details of the ECEPP force-field used within this thesis is given. Finally, the method used to determine the three-dimensional structures of the proteins by using the ECEPP force field concludes this chapter.

Chapter-6, mainly covers the experimental results of the proposed methods introduced in the previous chapters. First, the results for the proposed reinforcement based model for the HP lattice model is given. Then, the performance of the VS algorithm on the benchmark numerical function set and on the off-lattice AB model is given. The performance of the modified-energy function for the off-lattice AB model is also studied in this section. Computational results for the all-atom (ECEPP force field) model is given along with the three-dimensional structures determined for the provided protein set.

Finally, Chapter-7 concludes the thesis with a short discussion on possible future studies.

2. PROTEIN STRUCTURES

2.1 Protein Structures

Proteins are one of the most essential building blocks of living organisms. In living cells, most of the functions take place with the help of proteins. This functional diversity provided by the proteins is achieved by various combinations of 20 basic amino acids forming the proteins. Each protein has its unique amino acid sequence. Once the proteins are synthesized (or the sequence of the amino acids is formed), they fold into a unique three-dimensional structure that makes them functional or biologically active. Thus, it can be inferred that, the unique three-dimensional structure of a protein is determined by its unique amino acid sequence [20].

The structures of the 20 basic amino acids are shown in Figure 2.1. Each amino acid is represented by a three-letter or one-letter abbreviation. In Table 2.1, these abbreviations are listed. From Figure 2.1, it can be shown that, except the Proline, all of the remaining 19 amino acids share a common structure. This common structure is shown in Figure 2.2 and it consist of an amino group and a carboxyl group which are bonded to the alpha carbon (α carbon), a hydrogen atom and a side-chain (R chain). Different properties among the amino acids arise from the variations in the structures of different R groups.

Amino acids have different physicochemical properties, some of which are common for certain group of amino acids. These properties are mainly determined by the side-chains of amino acids. For the coarse-grained methods only the hydrophobicity properties of amino acids are interested which are listed in Table 2.1 along with the charge properties.

The structures of the proteins are mainly formed by the peptide and disulfide bonds. A peptide bond is formed between two amino acid molecules when the carboxyl group of one molecule reacts with the amino group of other molecule, releasing a molecule of water (H_2O). In Figure 2.3, peptide bond formation is shown.

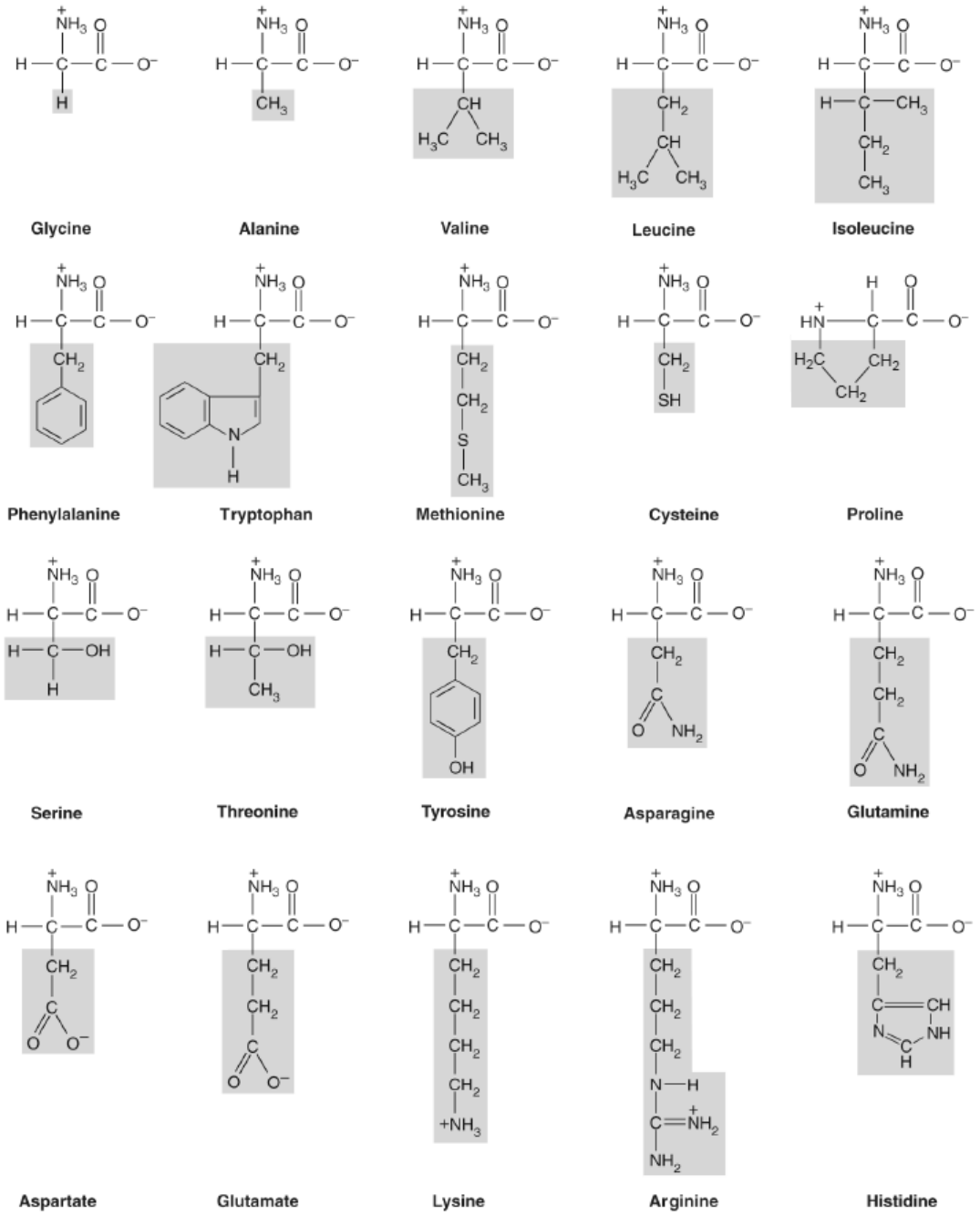

Figure 2.1 : 20 basic amino acids [20].

In Figure 2.3, only two amino-acid molecules react with each other. Thus, the resulting molecule is named as a dipeptide. With the addition of new amino acid molecules, the dipeptide chain gets longer and a polypeptide chain is formed. Proteins are composed of one or more polypeptide chains. Therefore, proteins are also named as polypeptides. One side of a polypeptide chain has an amino group which is named as N-terminal, and the other side of a polypeptide chain has a carboxyl group which is named as C-terminal. The polypeptide chain then folds into

a unique three-dimensional structure to form the protein structure as mentioned before. Since the peptide bonds are very rigid bonds, during the folding process the three-dimensional structure of the protein is formed by the turns around the C_α-C and C_α-N bonds for which a representative sketch is shown in Figure 2.4.

Table 2.1 : Basic amino acids and their three-letter and one-letter abbreviations.

Amino acid	Three-letter code	One-letter code	Hydrophobicity	Charge
Alanine	Ala	A	hydrophobic	neutral
Arginine	Arg	R	hydrophilic	+
Asparagine	Asn	N	hydrophilic	neutral
Aspartic acid	Asp	D	hydrophilic	-
Cysteine	Cys	C	moderate	neutral
Glutamic acid	Glu	E	hydrophilic	-
Glutamine	Gln	Q	hydrophilic	neutral
Glycine	Gly	G	hydrophobic	neutral
Histidine	His	H	moderate	+
Isoleucine	Ile	I	hydrophobic	neutral
Leucine	Leu	L	hydrophobic	neutral
Lysine	Lys	K	hydrophilic	+
Methionine	Met	M	moderate	neutral
Phenlyalanine	Phe	F	hydrophobic	neutral
Proline	Pro	P	hydrophobic	neutral
Serine	Ser	S	hydrophilic	neutral
Threonine	Thr	T	hydrophilic	neutral
Tryptophan	Trp	W	hydrophobic	neutral
Tyrosine	Tyr	Y	hydrophobic	neutral
Valine	Val	V	hydrophobic	neutral

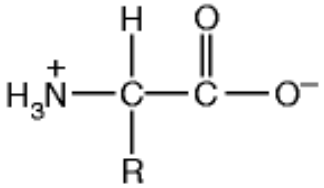

Figure 2.2 : General structure of an amino acid.

There are four basic structures of proteins. The primary structure of proteins are determined by the amino acid sequence that is encoded by genes. The secondary structure of proteins are defined by the local structures of the folded three-dimensional structure which is also known as the tertiary structure. Structures those include two or more tertiary structures are known as quaternary structures. In Figure

2.5, four basic structures of proteins are shown. A more detailed information is also provided below for these four basic structures.

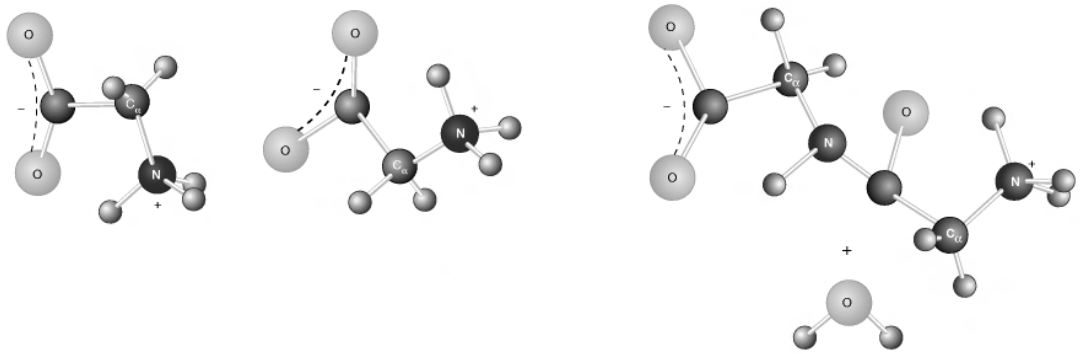

Figure 2.3 : Peptide bond formation between two amino acid molecules [20].

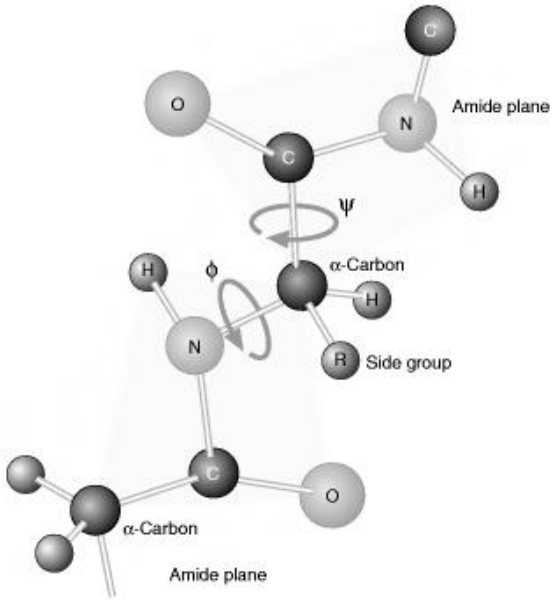

Figure 2.4 : Turns around the C_α-C and C_α-N bonds during the folding process [20].

2.1.1 Primary structure of proteins

As mentioned before, all proteins have their unique amino acid sequences. Here, the primary structure of proteins are determined by this unique amino acid sequence resided between the N-terminal and C-terminal.

Proteins those have similar primary structures are known as "homolog" proteins. The studies performed on the primary structures of proteins are mainly focused on the sequence similarity of different species to infer some genetic relationships among these species. For example, the myglobin protein, which is common for most of the species, has 153 identical amino acids for human and whale species [21].

2.1.2 Secondary structures of proteins

Secondary structures of proteins are formed by the local changes. These local changes occur as a results of the interactions between amino acids which are close to each other in the primary structure. In the globular proteins, basic units of the secondary structures can be classified as, alpha helix (α-helix), beta sheet (β-sheet) and turns.

Perhaps the most well-known and easily recognizable structures are the α-helices which are common in most of the protein structures. It is known that, 30% amino acids of the globular proteins are in the α-helical form [21]. In Figure 2.5, the structure of an α-helix is shown. As it can be shown from this figure, α-helix has a spiral like structure which is stabilized by the hydrogen bonds parallel to the helix axis. Each turn of a helix has 3.6 amino acids and the linear distance between the starting point and ending point of a turn (pitch) is 5.4 angstrom [22].

2.1.3 Tertiary structures of proteins

Tertiary structure is formed by further folding of secondary structures in three-dimensional space with the help of disulfide bonds, hydrophobic effects, and van der-waals forces etc. The tertiary structure of a protein is accepted as the stable minimum free energy conformation.

In a tertiary structure, α-helix and β-sheet structures can be found alone or together. There are also some combinations of α-helices and β-sheets connected through turns, that form patterns which are present in many different protein structures. This type of structures are named as super-secondary structures or motifs. Some example of motifs are alpha-alpha (two α-helices linked by a turn), beta-beta (two β-strands linked by a turn), beta-alpha-beta (β-strand linked to an α-helix that is also linked to another β-strands by turns). There are also some more complex motif structures like the Greek-key and the beta-barrel.

Another hierarchical level of protein tertiary structure is known as "domain". Domains are independently folding and functional structural units of a protein that are formed by the segments of the polypeptide chain. Proteins can have multiple structural domains and a particular domain can be found in different proteins. Domains of different proteins can come together to form new functional protein complexes which is known as domain-domain interaction.

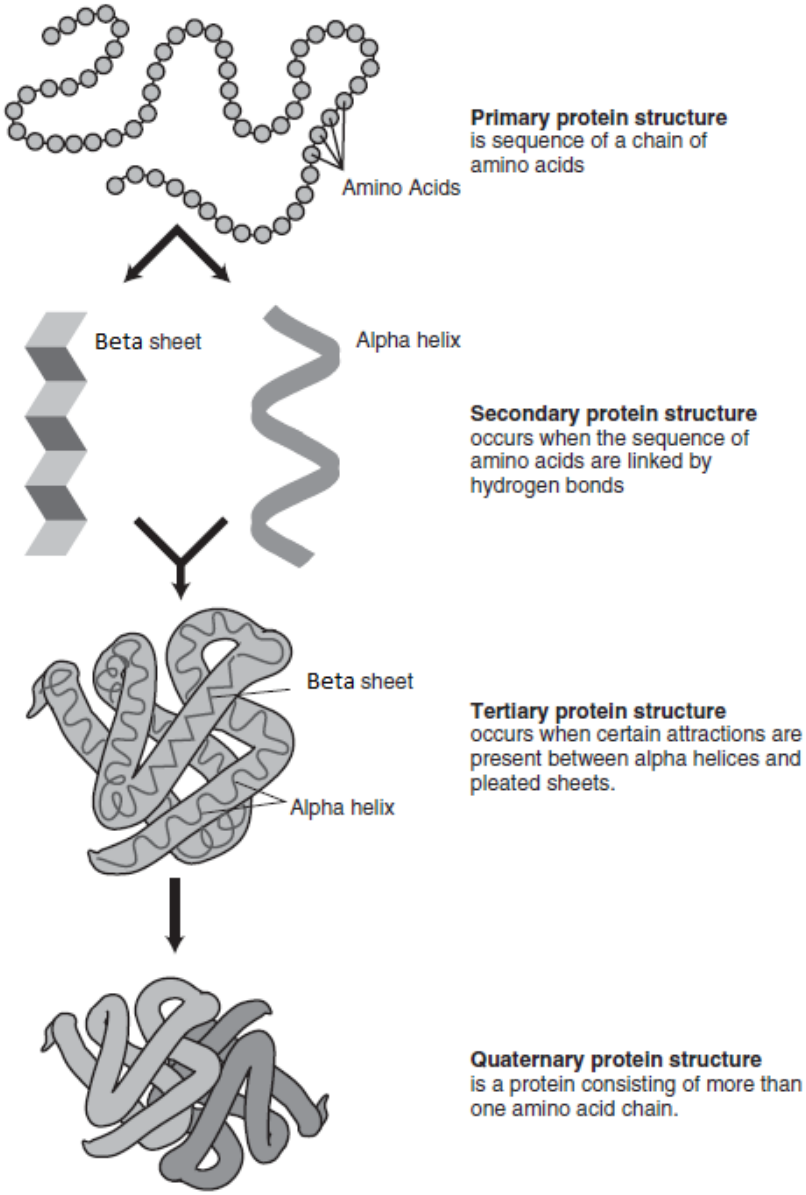

Figure 2.5 : Primary, secondary, tertiary and quaternary structures of the proteins [23].

2.1.4 Quetarnary structures of proteins

Many proteins consist of more than one polypeptide chain. Here, the quaternary structure of proteins is formed by the interactions of these polypeptide chains constituting the protein. The interactions forming the quaternary structure of proteins are totally same with the interactions those forming the tertiary structure. But different from the tertiary structure, in the quaternary structure these interactions occur among the polypeptide chains. In quaternary structures, polypeptide chains are usually called as the sub-units. In Figure 2.5, a representative sketch of the protein quaternary structure is shown.

2.2 Experimental Methods Used in Protein Tertiary Structure Determination

There are two main experimental methods used for protein tertiary structure determination. X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy.

2.2.1 X-ray crystallography

X-ray crystallography requires protein crystals, which are formed by vapor diffusion from purified protein solutions under optimal conditions [24]. Crystallization of a protein is a laboring process which may take months or even years to grow a crystal large enough. The growth crystal of the protein is subjected to an X-ray beam and a diffraction of the beam occurs. The resulting diffraction pattern is recorded on a film that is sensitive to X-ray radiation or an area detector is used. The rules for diffraction are given by Bragg's law [25]. By using the Bragg's law and the amplitude and phase data of the diffracted beams, the electron density maps are calculated. The corresponding protein tertiary structure is then obtained by fitting the amino acid sequence to the electron density maps. This process is also a labor intensive process and requires experienced scientist to interpret and to determine the correct coordinates of atoms constituting the protein structure.

2.2.2 Nuclear magnetic resonance spectroscopy

NMR spectroscopy does not require the crystallization of the protein, but the NMR is performed in a solution. In the NMR spectroscopy, the magnetic moment property of the nuclei atoms such as hydrogen, carbon and nitrogen, is utilized in order to determine distances between atoms in a molecule. This is done by exposing the protein solution to an external magnetic field and high frequency pulses. Then, the emitted radiation from the nuclei of the sample is recorded. It is possible to distinguish different emitted frequencies for different types of atom groups. A problem associated with NMR methods is that of ambiguity. Often a number of possible structures are generated, each equally good according to the method.

2.3 The PDB Database

The three-dimensional structures of proteins resolved by the experimental methods, X-ray crystallography and NMR spectroscopy, are deposited in the PDB database.

The PDB database is the only database that records the three-dimensional structure of molecules such as, proteins, nucleic acids and some other complex molecules. Thus, the PDB database is quite important for the scientist researching in biomedical and agricultural sciences.

As of February 2015, there are 106293 molecule structures deposited in the PDB database. 98770 of these structures are protein structures and 88517 of these protein structures are determined by X-ray crystallography, 9495 by NMR spectroscopy, 529 by electron microscopy, 68 of them by hybrid methods and 161 of them are determined by using some other methods [26].

In the PDB database each entry is uniquely identified by a four-letter code. In the first part of a PDB entry there are the name of the molecule, the biological source, some bibliographic references, and the R-value and R-free factors. R-value is the measure of the quality of the atomic model obtained from the crystallographic data. When solving the structure of a protein, the researcher first builds an atomic model and then calculates a simulated diffraction pattern based on that model. The R-value measures how well the simulated diffraction pattern matches the experimentally-observed diffraction pattern. A totally random set of atoms will give an R-value of about 0.63, whereas a perfect fit would have a value of 0. Typical values are about 0.20 [26].

In Figure 2.6, a sample PDB file format is shown for the protein 1A3I. A brief explanation of the parts in Figure 2.6 is provided below.

HEADER, TITLE, EXPDATA and AUTHOR: This part provides information about the researchers who defined the protein structure and the experimental method that is used to determine this structure.

REMARK: This part contains free-form annotation.

SEQRES: This part provides the sequence information for the corresponding protein structure. Each chain of a protein is identified by a letter. If a protein that consists of three polypeptide chains as in Figure 2.6, the chains are identified as A, B and C.

ATOM: In this part of the file, coordinate information of each atom constituting the protein structure is provided. For example, in Figure 2.6 the first atom is the nitrogen (N) atom of the amino acid proline (PRO) of the chain A. The xyz coordinates for this atom is (8.316, 21.206, 21.530). In the remaining three columns of a line

provided for an ATOM part, the occupancy information, the temperature factor and the element symbol are provided, respectively.

HETATM: This part describes the coordinate information of het-atoms, that is those atoms which are not part of the protein molecule.

```
HEADER    EXTRACELLULAR MATRIX                    22-JAN-98   1A3I
TITLE     X-RAY CRYSTALLOGRAPHIC DETERMINATION OF A COLLAGEN-LIKE
TITLE    2 PEPTIDE WITH THE REPEATING SEQUENCE (PRO-PRO-GLY)
...
EXPDTA    X-RAY DIFFRACTION
AUTHOR    R.Z.KRAMER,L.VITAGLIANO,J.BELLA,R.BERISIO,L.MAZZARELLA,
AUTHOR   2 B.BRODSKY,A.ZAGARI,H.M.BERMAN
...
REMARK 350 BIOMOLECULE: 1
REMARK 350 APPLY THE FOLLOWING TO CHAINS: A, B, C
REMARK 350   BIOMT1   1  1.000000  0.000000  0.000000        0.00000
REMARK 350   BIOMT2   1  0.000000  1.000000  0.000000        0.00000
...
SEQRES   1 A    9  PRO PRO GLY PRO PRO GLY PRO PRO GLY
SEQRES   1 B    6  PRO PRO GLY PRO PRO GLY
SEQRES   1 C    6  PRO PRO GLY PRO PRO GLY
...
ATOM      1  N   PRO A   1       8.316  21.206  21.530  1.00 17.44           N
ATOM      2  CA  PRO A   1       7.608  20.729  20.336  1.00 17.44           C
ATOM      3  C   PRO A   1       8.487  20.707  19.092  1.00 17.44           C
ATOM      4  O   PRO A   1       9.466  21.457  19.005  1.00 17.44           O
ATOM      5  CB  PRO A   1       6.460  21.723  20.211  1.00 22.26           C
...
HETATM  130  C   ACY   401       3.682  22.541  11.236  1.00 21.19           C
HETATM  131  O   ACY   401       2.807  23.097  10.553  1.00 21.19           O
HETATM  132  OXT ACY   401       4.306  23.101  12.291  1.00 21.19           O
...
```

Figure 2.6 : PDB file format [27].

3. LATTICE MODELS FOR THE SOLUTION OF PROTEIN FOLDING PROBLEM

3.1 Lattice Models

Lattice models are approximate models which are proposed for fast exploration of the huge search space of the protein folding problem. In literature, these models are also named as simplified low resolution or coarse-grained models.

In the lattice models, each amino acid in the chain is treated in a binary form based on their hydrophobicity property (hydrophobic (H) or hydrophilic, polar (P)) and represented as a single bead in a lattice structure. The lattice structures can be in different forms with varying numbers of neighboring amino acids either in two-dimensional or three-dimensional (2D or 3D), such as square, cubic, triangular, face-centered-cube (FCC) or any of the Bravais Lattices [3].

In literature, there are two main lattice models, the HP model [10] and the Gō model [28]. When compared to the Gō model, the HP lattice model is a very well-known and highly studied model and thus, it is usually chosen as the base model for the comparison of different algorithms proposed for the protein folding problem in lattice models.

In the well-known HP lattice model, as mentioned before, each amino acid in a chain is treated either hydrophobic (H) or polar (P) and occupies a lattice position in a 2D or 3D lattice structure. The energy of a conformation is computed according to the number of neighboring H-H contacts in the lattice structure which are not consecutive in the amino acid chain. This model is based on the fact that, the hydrophobic force is one of the most effective forces in the protein folding dynamics. In the three-dimensional structure of the proteins, the hydrophobic amino acids are usually occur in the core of the proteins, whereas the hydrophilic (or polar) ones occur in the surface of the proteins. Thus, by promoting the number of neighboring H-H contacts a hydrophobic core is implicitly formed within the lattice structure.

Let us, define the primary structure of a protein consists of n amino acid as $\mathcal{P}$. In 2D-HP lattice model this protein can be mathematically defined as below;

$$\mathcal{P} = p_1 p_2 p_3 .... p_n, \quad p_i \in \{H, P\}, \forall 1 \le i \le n \tag{3.1}$$

Here, $p_i \in \{H, P\}$ represents each amino acid in the chain which are either hydrophobic or hydrophilic (polar). A valid protein structure is defined with a function C, such that each residue of the amino acid chain is mapped to the lattice points in Cartesian coordinates by this function. This can be mathematically defined as in (3.2).

$$\begin{gathered} \mathcal{B} = \{\mathcal{P} = p_1 p_2 p_3 .... p_n \mid p_i \in \{H, P\}, \forall 1 \le i \le n, n \in N\} \\ \mathcal{G} = \{G = (x_i, y_i) \mid x_i, y_i \in \Re, 1 \le i \le n\} \\ C : \mathcal{B} \to \mathcal{G} \end{gathered} \tag{3.2}$$

Here, $C : \mathcal{B} \to \mathcal{G}$ represents the mapping process of an amino acid $p_i \in \{H, P\}$ to a lattice point (x_i, y_i) in Cartesian coordinates. After this mapping process, for $\forall 1 \le i, j \le n,$ with $|i - j| \ge 2$ the energy of the resulting protein structure in 2D-HP lattice model is defined as in (3.3).

$$E(C) = \sum_{i,j} I(i, j)$$

$$I(i, j) = \begin{cases} -1, & \text{if } p_i = p_j = H \text{ and } |x_i - x_j| + |y_i - y_j| = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

where (x_i, y_i) represents the position of the amino acid $p_i \in \{H, P\}$ and (x_j, y_j) represents the position of the amino acid $p_j \in \{H, P\}$ in Cartesian coordinates. More clearly, the energy function is decreased by 1 for each two amino acids that are mapped by C on neighboring positions in the lattice, but that are not consecutive in the primary structure $\mathcal{P}$. Such two amino acids are called as topological neighbors. In Figure 3.1, a sample configuration with energy -9 for the protein $\mathcal{P}$ = HPHPPHHPHPPHPHHPPHPH is given.

In literature, a number of optimization methods (including Monte Carlo methods, Evolutionary Algorithms, Tabu Search and hybrid approaches) have been proposed for the solution of the protein folding problem by using HP lattice model. In the following subsection, a review of the existing studies can be found.

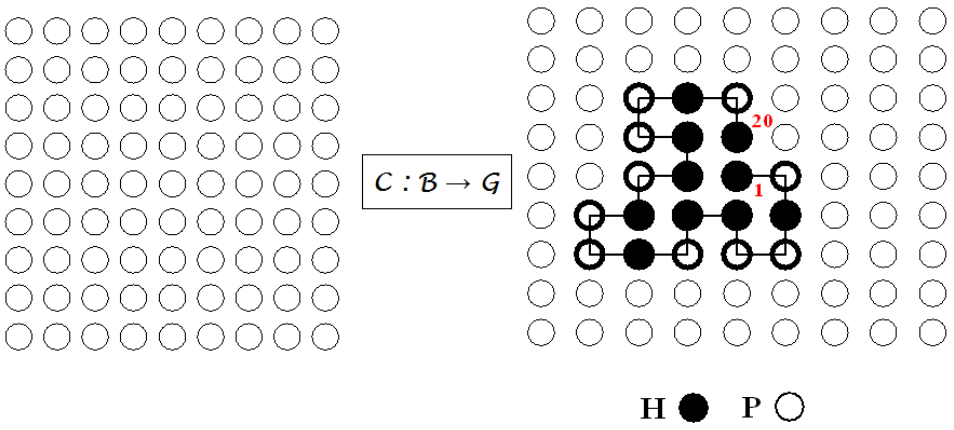

Figure 3.1 : A sample configuration with energy -9 after mapping process of the protein $\mathcal{P}$ = HPHPPHHPHPPHPHHPPHPH in 2D HP lattice model.

3.1.1 Literature review for the HP lattice model

In [29], authors proposed a method which is based on an improved ant colony optimization (ACO) approach (called ACO-HPPFP-3) for the solution of protein folding problem both in 2D and 3D HP lattice models. They showed that, the proposed method performs better than the previous state-of-the-art algorithms (such as the Genetic Algorithm (GA), Evolutionary Monte Carlo (EMC) and Pruned Enriched Rosenbluth Method (PERM)) on sequences whose native conformations do not contain structural nuclei (parts of the native fold that predominantly consist of local interactions) at the ends, but rather in the middle of the sequence, and that it generally finds a more diverse set of native conformations. Their experimental results also indicated that, the proposed ACO based approach scales worse with sequence length but usually finds a more diverse ensemble of native states. In [30], authors implemented and evaluated the replica exchange Monte Carlo (REMC) method, which had been applied very successfully to more complex protein models and other optimization problems with complex energy landscapes, in combination with the highly effective pull move neighborhood in two widely studied Hydrophobic Polar (HP) lattice models. They demonstrated that REMC utilizing the pull move neighborhood significantly outperforms current state-of-the-art methods (PERM and ACO-HPPFP-3) for protein structure prediction in the HP model on 2D and 3D lattices. In [31], based on the evolutionary Monte Carlo (EMC) algorithm, authors made four points of ameliorations and proposed a so-called genetic algorithm based on optimal secondary structure (GAOSS) method to efficiently predict the protein folding conformations in the two-dimensional

hydrophobic–hydrophilic (2D HP) model. Nine benchmarks were tested to verify the effectiveness of the proposed approach and the results showed that for the listed benchmarks GAOSS can find the best solutions so far. In [32], a filter-and-fan approach is was proposed for the solution of the protein folding problem in 2D HP lattice model. The method augments a simple neighborhood structure in a dynamic and adaptive fashion using a truncated form of tree search. By means of the filter-and-fan approach, authors pursued the approach of seeking an effective guidance strategy within a simpler neighborhood by extending the so-called pull-move neighborhood. The algorithm is equipped with a tabu search short-term memory aimed at enhancing the local search and providing the tree search with appropriate legitimacy restrictions to drive the search and keep the method from generating duplicated conformations. Computational results for standard sets of benchmark problems showed that the F&F algorithm is highly competitive with the existing leading algorithms, requiring only a single solution trial to obtain best known solutions to all problems tested, in contrast to a hundred or more trials required in the typical case to evaluate the performance of the best of the alternative methods. In [33], authors proposed a new branch and bound method which can be successfully applied to the protein folding problem under the 2D-HP model. The proposed method utilizes the dynamic threshold and r-step checking techniques which can reduce the search space and thus make the algorithm more efficient. In [34] authors proposed a hybrid of hill-climbing and genetic algorithm (HHGA) based on elite-based reproduction strategy for protein structure prediction on the 2D triangular lattice. In the proposed study, two local search operators are also used to improve the solutions during the search process. First, given the current solution, local search I chooses its neighbor residues, which are generated in a way similar to mutation operation: i.e., randomly changing its direction. Consequently, if the fitness value of a neighbor is better than the current solution, this neighbor residue will be accepted to replace the current one. In local search II, the neighbor residues are generated in a way similar to crossover operation. That is, five neighbors are created by changing the direction of the second segment after the crossover point, where rotation angles are 60°, 120°, 180°, 240° and 300°, respectively. If any of the five folding directions leads to a superior fitness to the original direction, this neighbor will replace the current solution. The simulation results showed that the proposed HHGA can successfully deal with the protein structure prediction problems. Specifically, HHGA

significantly outperforms conventional genetic algorithms and is comparable to the state-of-the-art method in terms of free energy. In [35], by incorporating the generation of an initial conformation based on a greedy strategy, the conformation update mechanism based on pull moves, and some heuristic off-trap strategies into an improved energy landscape paving (ELP) method, authors proposed an alternative version of the ELP method, called the ELP–pull-move method. The proposed method was tested on both 2D and 3D hydrophobic-hydrophilic protein-folding models. Experimental results indicated that, for ten 2D benchmark sequences of length ranging from 20 to 100, the proposed algorithm can find the lowest energies found so far. Obtained results showed that, the algorithm converges more rapidly and efficiently than previous methods. For all ten 3D sequences with a length of 64, the ELP–pull move method finds lower energies within comparable computational times. In another study [36], authors used the improved ELP method mainly for the 3D-HP model by using different length of sequences in their experiments. The algorithm was tested on three sets of 3D HP benchmark instances consisting 31 sequences. For eleven sequences with 27 monomers, the proposed method explores the conformation surfaces more efficiently than other methods, and finds new lower energies in several cases. For ten 48-monomer sequences, authors found the lowest energies so far. Obtained results showed that, the algorithm converges rapidly and efficiently. For all ten 64-monomer sequences, the algorithm finds lower energies within comparable computation times than previous methods. In [37], authors proposed an improved evolutionary algorithm for the protein folding problem on 3D FCC HP lattice model. They also combined three different local search methods, including lattice rotation for crossover, K-site move for mutation, and generalized pull move; to improve previous EA-based approaches. Experimental results showed that the proposed approach is able to find optimal conformations which were not found by previous EA-based approaches. In [38], authors proposed a novel approximation algorithm to solve the protein folding problem in HP model. Authors claimed that the proposed algorithm is polynomial in terms of the length of the given HP string. Hence the algorithm is expected to perform very well for larger HP strings. In [39], authors introduced a simplified energy function for the 2D-HP model. Different from the original function (which is discrete), the simplified energy function searches the optimum protein structures in a continuous space. The simplified energy function totals the distance between all pairs of hydrophobic amino

acids. To optimize the simplified energy function, they introduced one of the recently proposed swarm intelligence algorithm, the firefly algorithm (FA). In the experiments, 14 sequences of different chain lengths from 18 to 100 were used as the dataset. The obtained results of FA were compared to the standard genetic algorithm and immune genetic algorithm. Each algorithm ran 20 times. The averaged energy convergence results showed that FA achieves the lowest values. A point should be noted for this study is, in this study authors used their own artificial sequences. The widely used benchmark sequences are not used for comparison. Thus, it is an open question whether the simplified energy function will work well on these benchmark sequences which are relatively difficult than the other ones. In [40], authors proposed a new Spiral Search algorithm based on tabu-guided local search. The proposed algorithm uses a novel H-core directed guidance heuristic that squeezes the structure around a dynamic hydrophobic-core centre. In this method, random walks were applied to break premature H-cores and thus to avoid early convergence. A novel relay-restart technique was also used to handle stagnation. The proposed algorithm was tested on a set of benchmark protein sequences. The experimental results showed that the proposed spiral search algorithm outperforms the state-of-the-art local search algorithms for simplified protein structure prediction.

Above mentioned studies along with the some newly published ones [41-43], mostly use a two-step search process to find the optimum fold of the sequences in 2D or 3D HP lattice model. In the first step usually an optimization algorithm such as ACO, EMC, GA, FA etc. is used to find a local conformation and then in the second step a local search method such us pull-moves, K-site move, lattice rotation, local structural motifs etc. is used to further improve this local conformation. Since the energy landscape of the protein folding problem in HP lattice model is highly complex the algorithms are usually trapped into a local minimum point. Thus, the use of a local search method is usually required to avoid from these local points.

In this thesis, a reinforcement learning based approach is proposed for the solution of the protein folding problem in 2D HP lattice model. The use of reinforcement learning methods is relatively new when compared to the existing ones. In [44-47], authors proposed some reinforcement learning based methods for the solution of the protein folding problem in 2D HP lattice model. Details of these methods are given

in comparison to our newly proposed reinforcement learning based method in the following sub-section.

3.2 The Proposed Reinforcement Learning Based Method for the Solution of Protein Folding Problem in Two-Dimensional HP Lattice Model

The benefits of using lattice structures in protein folding problem are twofold. Firstly, lattice structures discretize the search space and thus reduces the size of the search space when compared to the continuous one. Secondly, the grid like structure of lattices guides the algorithm during the search process to form self-avoiding protein configurations, in which each amino acid in the sequence is mapped to only a particular point on the grid. This mapping process is usually handled in two different ways. In the first one, the amino acid sequence is considered as a constant chain and folding is performed by iteratively modifying the positions of each amino acid on the grid without breaking this chain. While in the second one, each amino acid in the sequence is consecutively added to form continuous and self-avoiding amino acid chains on the grid which can be considered as a navigation problem or a robot path planning problem.

It is shown that, reinforcement learning methods perform well on the solution of the robot path planning problems [48-49]. Thus, in this thesis the reinforcement learning methods are used for the solution of the protein-folding problem in two dimensional lattice model. There exist many studies in literature that proposed different methods for the solution of this problem. A review of the existing studies is given in the previous section. However, the use of reinforcement learning methods are quite new. In [44-47], authors used the Q-learning algorithm to solve the protein folding problem in two dimensional hydrophobic-polar (2D-HP) model.

In order to use the reinforcement learning methods for the solution of the protein folding problem in 2D-HP model, first a state-action space should be defined properly. Thus, each move of the agent on the grid could be easily mapped into the defined state-action space.

In the existing studies [44-47], a state-action space is defined for this purpose. However, in these studies the size of the defined state-action space is highly affected by the amino acid sequence length. As the amino acid sequence length increases, the

size of the proposed state-action space also increases, dramatically. So, even for the small sized amino acid sequences it is not computationally possible to create the state-action space at the beginning of the algorithm. The only way is to create the state-action space dynamically during the learning process, which is not desirable. Moreover, in these studies the state-action space is created for all amino acid sequences individually. So, all amino acid sequences have a unique state-action space and the algorithm must learn all of these state-action spaces separately. By this way, after a learning process, the proposed method could not be able to find the optimal fold of another amino acid sequence, which conflicts with the philosophy of the term “learning”.

In this thesis, to overcome above mentioned drawbacks a new state-action space is proposed. The proposed state-action space allows the agent to find the optimal fold of any amino acid sequence (protein) with a particular length. This is achieved by incorporating the "learning" concept to the 2D HP protein folding problem by using the newly proposed state-action space. Moreover, by utilizing a swarm based reinforcement method (Ant-Q algorithm) the optimal fold is found rapidly when compared to the traditional Q-learning algorithm.

3.2.1 State space representations

A valid protein configuration forms a self avoiding path which means, the mapped positions of two different amino acids must not be same in the 2D grid. By considering the resulting self avoiding path, a solution can be represented by an $n-1$ length sequence $\pi = \pi_1\pi_2\pi_3....\pi_{n-1}, \quad \pi_i \in \{L, R, U, D\}, \ \forall 1 \le i \le n-1$ of directions, which encodes the relative positions of the current amino acid to the previous one. Let us consider the configuration given in Figure 3.1. The resulting sequence for this protein then can be given as $\pi = RDDLULDLLURURULURRD$.

In order to study the protein folding problem in 2D-HP lattice model with reinforcement learning methods, first a state-action space that encodes the above mentioned sequence of directions should be proposed. In the following sub-sections, the state-action space defined in the existing studies [44-47] and the state-action space defined in this thesis is given, respectively.

3.2.1.1 The existing state space representation

The state space S proposed by Czibula et.al. [44-47] consists of $\frac{4^n-1}{3}$ states i.e. $S=\left\{s_1,s_2,...,s_{\frac{4^n-1}{3}}\right\}$ which is given in Figure 3.2. At the beginning, the agent is at state s_1. A state $s_{i_k}\in S\left(i_k\in\left[1,\frac{4^n-1}{3}\right]\right)$ is reached by the agent at a given moment after it has visited states $s_1,s_{i_1},s_{i_2},...s_{i_{k-1}}$ is a terminal state if the number of states visited by the agent in the current sequence is $n-1$ i.e. $k=n-2$. A path from the initial to the final state forms a configuration for the protein $\mathcal{P}$. In Table 3.1 resulting states at each step are given as an example for the protein sequence $\mathcal{P}$ = HPHPPH.

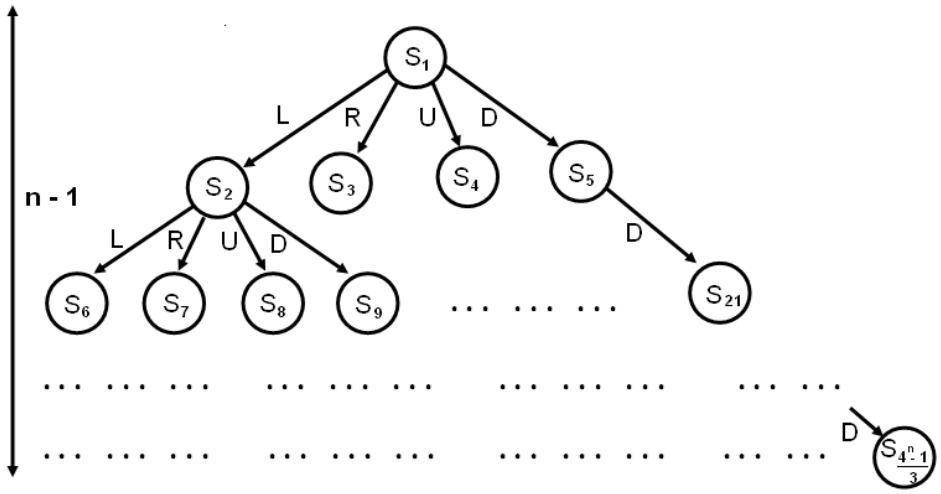

Figure 3.2 : State space for the reinforcement learning method given in [45].

The action space $\mathcal{A}$ consists of 4 actions *L (Left), R (Right), U (Up), D (Down)* which are the relative directions of the current position of the agent to the previous one. So, the action space can be given as $\mathcal{A}=\{a_1,a_2,a_3,a_4\}$, where $a_1=L$, $a_2=R$, $a_3=U$ and $a_4=D$.

At a given moment, from a state $s\in S$ agent can move in 4 successor states, by executing one of the 4 possible actions. Thus, the transition probability between a state s and each neighbor state s' of s is equal to 0.25.

In the existing method, the state space S (the agent's environment) consists of $\frac{4^n-1}{3}$ states i.e. $S=\left\{s_1,s_2,...,s_{\frac{4^n-1}{3}}\right\}$. Let us consider the sequence given in Figure 3.1 which has a total number of 20 amino acids. So, for this sequence the state space S consists of $\frac{4^{20}-1}{3}=3.665\times10^{11}$ states. Thus, even to create the state-action space for a sequence length of 20, the required computational time is dramatically huge and it is not possible to create such a big space by an average PC. The only way is to create the state space dynamically, but there is not any information provided about this point in the studies [44-47]. To sum up, in the existing method the size of the state space S strictly depends on the length of the amino acid sequence. As the length of the amino acid sequence *n* increases, the size of the state space also increases dramatically.

Table 3.1 : An example of the existing state space representation for the sequence $\mathcal{P}$= HPHPPH.

Agent in state	Can choose action	Resulting states
S1	L, R, U, D	S2 = L S3 = R S4 = U S5 = D
S2	L, R, U, D	S6 = LL S7 = LR S8 = LU S9 = LD
S3	L, R, U, D	S10 = RL S11 = RR S12 = RU S13 = RD
S4	L, R, U, D	S14 = UL S15 = UR S16 = UU S17 = UD
S5	L, R, U, D	S18 = DL S19 = DR S20 = DU S21 = DD
. . .	L, R, U, D	. . .
S1361	L, R, U, D	S1362 = DDDDDL S1363 = DDDDDR S1364 = DDDDDU S1365 = DDDDDD

Another point that should be commented on is the learning stage of the existing method. Let us again consider the amino acid sequence given in Figure 3.1. Since the state space S encodes the relative positions of the current amino acid to the previous one, after a learning process all we have is a sequence of directions where the optimal one is given as *RDDLULDLLURURULURRD* for the sequence given in Figure 3.1. Here, it should be noted that at the end of the learning process there is not

any information provided about the characteristics of the amino acids whether they are hydrophobic or hydrophilic. This information is totally lost at the end of the learning process. Thus, for an another amino acid sequence the state-action space must be re-initialized and the agent must learn the environment for this new sequence. This situation conflicts with the philosophy of the term "learning". Because, in a learning process the previous information must be conserved.

3.2.1.2 The proposed state space representation

In the previous sub-section the drawbacks of the existing reinforcement learning method are discussed. In this thesis, to overcome these drawbacks a new state space representation is proposed. Unlike the existing one, the proposed state-action space comprises the characteristics of the amino acids. By this way, the information stored in the state-action space is conserved. So, there is no need to re-initialize the state-action space for another amino acid sequence.

This section mainly covers the definition of the proposed state action space. To allow a comparison to the existing method the proposed state-action space is studied in two different scenarios.

Scenario 1: In the first scenario, the agent tries to find the optimal policy for only a particular amino acid sequence which is also the case in the above mentioned existing method. In Figure 3.3, the proposed state-action space is given for this case. As it can be shown in Figure 3.3, the new state-action space has a matrix like structure in which each column represents an element of the amino acid sequence that is hydrophobic or polar. Again there are four possible directions $\{L, R, U, D\}$ that agent can move when it is at a state *s*.

In this case, the total number of states S consists of only $[4\cdot(n-1)+1]$ states for a n length amino acid sequence. Let us again consider the amino acid sequence $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH given in Figure 3.1. The total number of states is $[4\cdot(20-1)+1]=77$ states, which is very small when compared to the existing one (3.665×10^{11}). For better understanding, in Table 3.2 all of the state action pairs are given for a short sequence $\mathcal{P}_2$ = HPHPPH.

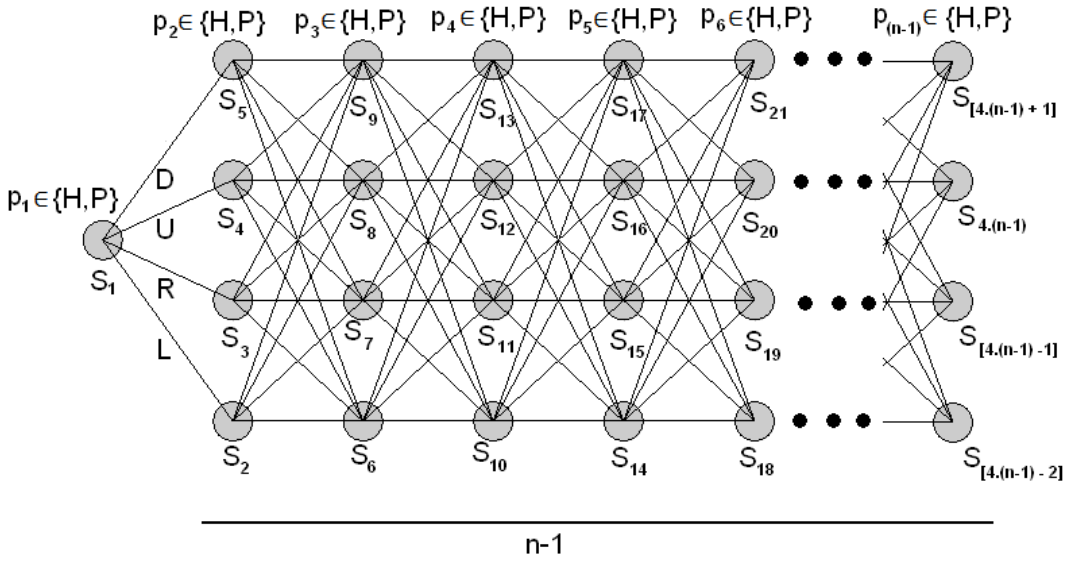

Figure 3.3 : The proposed state space for Scenario-1.

Note that, as in the existing method here also it is not possible to talk about an actual "learning". Because, the agent learns the space for only the corresponding amino acid sequence and thus, the Q-table only consists of the state-action pairs for this individual amino acid sequence. However, when compared to the existing method, the new state-action space still has advantages. First, the size of the state-action space is reduced dramatically which allows creating the Q-table at the beginning of the algorithm. For the second advantage let us consider the sequence $\mathcal{P}_2$ = HPHPPH. Since $\mathcal{P}_2 \subset \mathcal{P}_1$, there is no need to learn the space for the $\mathcal{P}_2$ again. The Q-table created for the sequence $\mathcal{P}_1$ already comprises the solution for the $\mathcal{P}_2$. But since the existing method only encodes the directions, it is not possible to deduce whether $\mathcal{P}_1$ comprises $\mathcal{P}_2$ or not.

Scenario 2: In the previous scenario, only a particular amino acid sequence is considered that is why the Q-table only consists of the state-action pairs for the corresponding sequence and hence the subsets of this sequence. However, to talk about an actual learning the state-action space should comprise all of the possible combinations of an *n* length amino acid sequence. In Figure 3.4 the proposed state-action space for this case is given. Since an amino acid could be either H or P, for a n length amino acid sequence there are 2^n different sequences. Thus, the state-action space should be designed to allow an agent to learn all of these sequences. So, after a learning process an agent can find the optimal fold of a sequence with the help

of resulting Q-table which covers all of the state-action pairs. In Table 3.3 the state-action space for $n = 3$ is given for better understanding.

Table 3.2 : State-action space for the sequence $\mathcal{P}_2$ = HPHPPH (Scenario-1).

Agent in state	Can choose action	Resulting states
S1 = H	L, R, U, D	S2 = HPL S3 = HPR S4 = HPU S5 = HPD
S2 = HPL S3 = HPR S4 = HPU S5 = HPD	L, R, U, D	S6 = HPHL S7 = HPHR S8 = HPHU S9 = HPHD
S6 = HPHL S7 = HPHR S8 = HPHU S9 = HPHD	L, R, U, D	S10 = HPHPL S11 = HPHPR S12 = HPHPU S13 = HPHPD
S10 = HPHPL S11 = HPHPR S12 = HPHPU S13 =HPHPD	L, R, U, D	S14 = HPHPPL S15 = HPHPPR S16 = HPHPPU S17 = HPHPPD
S14 = HPHPPL S15 =HPHPPR S16 = HPHPPU S17=HPHPPD	L, R, U, D	S18 = HPHPPHL S19 = HPHPPHR S20 = HPHPPHU S21 = HPHPPHD

Note that, in Table 3.3 the total number of states is 25. But this number must be doubled because the initial state S1 is considered as H but it could also be P. So, in total for $n = 3$ there are 50 states. In this case, the total number of states is given as $S = (2 + 4 \cdot \sum_{n=2}^{N} 2^n)$, where N represents the length of the amino acid sequence. When compared to the existing method (in which total number of states for $n = 3$ is $\frac{4^3 - 1}{3} = 21$) for $n = 3$ the size of the proposed state-action space is bigger. However, as n increases the total number of states becomes much smaller than the existing one. For example for $n = 20$ the size of the proposed state-action space is $S = (2 + 4 \cdot \sum_{n=2}^{20} 2^n) \approx 8.4 \times 10^6$ which is much smaller than the existing one (3.665×10^{11}).

As it can be shown, the newly proposed state-action space reduced the size of the state-action space dramatically for both scenarios. However, for Scenario-2 the size of the state-action space is still highly depends on the amino acid sequence length n. But it should be noted that, the proposed state action space allows the agent to predict the optimal fold for $n + a$ length sequences, where $a \geq 1$. If the agent learns the space for all of the n length sequences, it will be possible to extend the space

and predict the optimal fold for longer sequences. This is another important advantage of the proposed state-action space and could be studied separately.

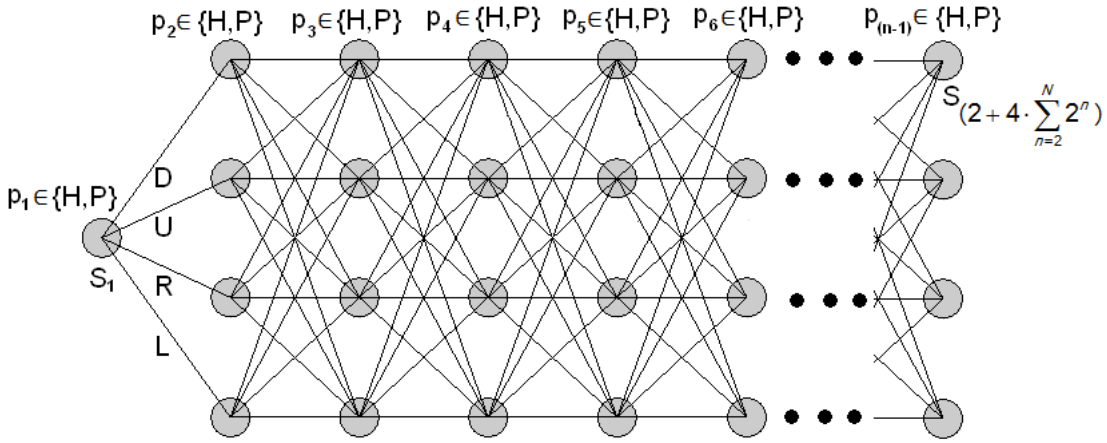

Figure 3.4 : The proposed state space for Scenario-2.

3.2.2 Reinforcement learning algorithms

A basic reinforcement learning model consists of a set of environment states, a set of actions, rules of transitioning between states, and rules that determine the scalar reward of a transition. Having defined the state-action spaces, now the transition rules and rules that determine the reward of a transition is given. These parts could be associated to the learning stage of a reinforcement learning method.

In [44-47], transition rules and the reward of a transition is defined as below:

If the transition generates a configuration that is not valid (i.e. self-avoiding) the received reward is 0.01; the reward received after a transition to a non terminal state is small positive constant greater than 0.01 (e.q 0.1); the reward received after a transition to a final state π_{n-1} is the minus energy of the protein $\mathcal{P}$ corresponding to the path π. These definitions are mathematically given in (3.4).

$$r(\pi_k \mid s_1, \pi_1, \pi_2, ...\pi_{k-1}) = \begin{cases} 0.01 & \text{if } a_\pi \text{ is not valid} \\ -E_\pi & \text{if } k = n-1 \\ 0.1 & \text{otherwise} \end{cases} \qquad \textbf{(3.4)}$$

where, $r(\pi_k \mid s_1, \pi_1, \pi_2, ...\pi_{k-1})$ represents the reward received by the agent in state π_k, after it has visited states $\pi_1, \pi_2, ...\pi_{k-1}$ and E_π represents the energy of the configuration formed by the path π.

Table 3.3 : State-action space for $n = 3$ (Scenario-2).

Agent in state	Can choose action	Resulting states
S1 = H or P (**let us consider S1 = H** Note that, all of this process must be repeated for **S1 = P**)	L, R, U, D	**if p2 = H** S2 = HHL S3 = HHR S4 = HHU S5 = HHD
		if p2 = P S6 = HPL S7 = HPR S8 = HPU S9 = HPD
S2 = HHL S3 = HHR S4 = HHU S5 = HHD	L, R, U, D	**if p3 = H** S10 = HHHL S11 = HHHR S12 = HHHU S13 = HHHD
		if p3 = P S14 = HHPL S15 = HHPR S16 = HHPU S17 = HHPD
S6 = HPL S7 = HPR S8 = HPU S9 = HPD	L, R, U, D	**if p3 = H** S18 = HPHL S19 = HPHR S20 = HPHU S21 = HPHD
		if p3 = P S22 = HPPL S23 = HPPR S24 = HPPU S25 = HPPD

With the above defined transition rules and the reward of a transition, Czibula et.al. [44-47] used the Q-learning algorithm as the reinforcement learning method to find the optimal policy.

The Q-learning algorithm is known to perform well on the small-sized state action spaces. However, as the size of the state-action space increases, random choices of the actions prevents the agent to converge the optimal solution. To avoid this problem, recently swarm based reinforcement learning methods are proposed [50-55]. In these methods, a number of agents learn concurrently by exchanging the information that they gain during the individual learning.

The performances of the swarm based reinforcement learning methods are highly affected by the way of exchanging the information among the agents. In [50], authors proposed an information exchange method based on ant colony optimization [56], which is inspired from behavior of real ants. Real ants deposit pheromone trails over the paths from their nest to the food source. Once the amount of the pheromone trail of a particular path is increased over time, this path becomes more attractive for the members of the ant colony. In [50], the same concept is used for the Q-learning algorithm, which is a widely used reinforcement learning algorithm. In this study, pheromone trails are deposited over the state-action space. Thus, the agents could avoid from random movements which helps to find the optimal policy rapidly, even

for the complicated problems. In the above mentioned study, the authors also compared the performance of the proposed ant based reinforcement learning method, namely the PHE-Q method, with the Q-learning algorithm and some other swarm based reinforcement learning methods (BEST-Q, AVE-Q and PSO-Q) that they proposed earlier. The performance of the PHE-Q method is examined by applying it to two shortest path problems. After several experiments, it is observed that the PHE-Q algorithm could find better policies in a small number of episodes when compared to the other methods. In another study, an ant based reinforcement method, the Ant-Q algorithm, is proposed for the solution of traveling salesman problem [55]. After several experiments, it is shown that the Ant-Q algorithm with delayed reinforcement is much more efficient than the other well known heuristic methods such as, Elastic Net (EN), Simulated Annealing (SA), Self Organizing Map (SOM), and Farthest Insertion (FI). Although, there are small differences between the Ant-Q and PHE-Q algorithms, in principle both algorithms use the same concept, the pheromone trails, to find out the optimal solution.

In this thesis, the Ant-Q algorithm [55] is used to overcome above mentioned drawbacks of the standard Q-learning algorithm. Different from the existing methods given above, the Ant-Q algorithm uses different state transition rules and rewarding mechanism. Details of the transition rules and the rewarding mechanism can be found in the Section 3.2.2.2.

3.2.2.1 The Q-learning algorithm

The Q-learning algorithm is an off-policy algorithm which is proposed to optimize the solutions in Markov decision process problems. It can be proven that, under sufficient training the algorithm converges to a close approximation of the action-value function for an arbitrary target policy. In Figure 3.5, a short description of the Q-learning algorithm is given.

In Figure 3.5, $Q(s,a)$ is the table entry for the corresponding state-action pair, $r(s,a)$ is the reward of the state-action pair *(s,a),* γ is the discount factor and finally $Q(s',a')$ is the table entry for the state-action pairs those can be reached from the state *s*. Once the training process is completed the agent learns the environment and constructs the optimal solution by starting from the initial state. For this purpose the Greedy selection mechanism is used. In Greedy selection mechanism, the agent

selects the next state according to the Q-values stored in the Q-table for each $Q(s,a)$ pairs. Thus, the agent moves from state *i* to a neighbor state *j* that have the maximum Q-value stored in the Q-table.

```
Initialize Q(s,a)=0 for each pair of (s,a)
Repeat (for each episode)
   Select the initial state s
   Choose a from s using policy derived from Q
  (e.g., ε greedy)
   Repeat (for each step of the episode)
     Take action a, observe r,s'.
     Update the table entry Q(s,a) as follows
     Q(s,a)←r(s,a)+γmax_a Q(s',a')
     s→s'
   until s is terminal
```

Figure 3.5 : A short description of the Q-learning algorithm.

3.2.2.2 The Ant-Q algorithm

Ant-Q algorithm was proposed by Gamberdalla and Dorigo [55] for the solution of symmetric and asymmetric travelling salesman problem. The Ant-Q algorithm was inspired by work on the ant system (AS), a distributed algorithm for combinatorial optimization based on the metaphor of ant colonies proposed in [57-58].

In this study the Ant-Q algorithm is adopted to the 2D-HP protein folding problem as follows:

Let $AQ(s,a)$ be a positive real value associated to the state-action pair (s,a). The AQ-values are the Ant-Q counterpart of Q-learning Q-values and is intended to indicate how useful it is to choose action a when the agent is in state s.

Let $HE(s,a)$ be a heuristic value associated to the state-action pair (s,a) which allows an heuristic evaluation of which transitions are better. In this study, the heuristic value is determined by the exponential of the number of new H-H contacts.

Let k be an agent whose task is to find the optimum folding starting from the initial state. Associated to k there is the list $J_k(s)$ of allowed states of to be visited, where s is the current state. This list implements a kind of memory, and is used to constrain agents to find feasible configurations, that is self-avoiding. When the agent is in the state s normally there are four possible actions. But since the resulting configuration

must be self-avoiding, all of these four actions are not always available. Here, $J_k(s)$ stores the states which could be reached by the agent k by performing an available action when the agent in the state s.

Based on the above considerations, when the agent is in the state s an action a is chosen with (3.5) which is known as the action choose rule (or the state transition) rule.

$$a = \begin{cases} \arg \max_{HE(s,u)} \left\{ [AQ(s,u)]^{\delta} \cdot [HE(s,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ r & \text{otherwise} \end{cases} \qquad \textbf{(3.5)}$$

where δ and β are parameters which weigh the relative importance of the learned AQ-values and heuristic values, q is a value chosen randomly with uniform probability in [0,1], q_0 $(0 \le q_0 \le 1)$ is a parameter such that the higher q_0 the smaller the probability to make a random choice, and r is a random variable selected according to a probability distribution given by a function of the $AQ(s,u)$'s and $HE(s,u)$'s, with $u \in J_k(s)$.

In Ant-Q algorithm m agents cooperates to find out the best solutions. Ant-Q values are updated by the following equation.

$$AQ(s,a) \leftarrow (1-\alpha) \cdot AQ(s,a) + \alpha \cdot \left(\Delta AQ(s,a) + \gamma \cdot \underset{z \in J_k(s)}{Max}\ AQ(s,z) \right) \qquad \textbf{(3.6)}$$

The update term given in (3.6) is composed of a reinforcement and of the discounted evolution of the next state. α and γ parameters are known as the learning step and the discount factor. In Ant-System (AS) and Ant-Q algorithm, the reinforcement term ΔAQ is always zero until the agent forms a complete fold. Different from the Q-learning algorithm in Ant-Q the reinforcement term $\Delta AQ(s,a)$ is computed at the end of the agent's tour. The computation method for this delayed reinforcement is given in the following parts of this section.

In Figure 3.6, a short description of the Ant-Q algorithm is given. In [55], a generic description of the Ant-Q algorithm also can be found. The algorithm given in [55] is called generic because in their studies Gamberdella and Dorigo tried some other action choice rules (*pseudo-random, pseudo-random-proportional, and random-proportional*) and delayed reinforcement methods (*global-best, and iteration-best*) to

further improve the performance. In [55], it was shown that the pseudo-random-proportional action choice rule with the iteration-best delayed reinforcement method performs better when compared to the other options. In (3.7), the pseudo-random-proportional action choice rule for the agent k is given.

```
Initialize AQ(s, a) = (0,1] for each pair of (s,a)
Repeat for each iteration
   Repeat for each agent
     Select the initial state s
     Repeat (for each step of the episode)
          Choose a from s by using the (3.7)
          Take action a, observe s'.
          Update the table entry AQ(s,a) as follows
          AQ(s,a) ← AQ(s,a) + γmax_a AQ(s',a'), (ΔAQ(s,a) = 0)
          s → s'
     until s is terminal
    end

    Repeat for each agent
     Compute the delayed reinforcement ΔAQ(s, a) by using the
     (3.8)
     Update the table entry AQ(s,a) as follows
     AQ(s,a) ← (1 − α)AQ(s,a) + α · ΔAQ(s,a), (γmax_a AQ(s',a') = 0)
    end
until the maximum number of iterations is reached.
```

Figure 3.6 : A short description of the Ant-Q algorithm.

$$p_k(s,a) = \begin{cases} \dfrac{[AQ(s,a)]^{\delta} \cdot [HE(s,a)]^{\beta}}{\sum_{u \in J_k(s)} [AQ(s,a)]^{\delta} \cdot [HE(s,a)]^{\beta}} & \text{if } a \in J_k(s) \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(3.7)}$$

where, $p_k(s,a)$ defines the probability of choosing an action a when the agent k is located in the state s.

As mentioned before, different from the Q-algorithm the Ant-Q algorithm uses delayed reinforcement method. In fact, due to the nature of the protein folding problem in 2D-HP model, it is much more convenient to use the delay reinforcement. As in the travelling salesman problem here also the complete tour (or configuration) defines whether the solution is good or not. Thus, instead of the immediate reward, the final configuration should be considered and rewarded. In (3.8) the way of computing the iteration-best delayed reinforcement is given.

$$\Delta AQ(s,a) = \begin{cases} \dfrac{E_{k_{ib}}}{L} & \text{if } (s,a) \in \text{configuration found by the agent } k_{ib} \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(3.8)}$$

where, $E_{k_{ib}}$ represents the energy of the configuration found by the agent k and L represents the length of the amino acid sequence.

4. OFF-LATTICE MODEL FOR THE SOLUTION OF PROTEIN FOLDING PROBLEM

4.1 The Off-Lattice AB Model

In lattice models, each amino-acid is mapped to a particular lattice point to form a continuous and self-avoiding amino-acid chain with fix bond lengths between successive amino-acid pairs. The lattice models benefits greatly from the discretization of protein phase space; however, it also suffers from this strategy. The discrete nature of the model surely affects the folding behaviors, especially the dynamics of the system [11]. To overcome this problem off-lattice model (or toy model) was proposed [12]. In the off-lattice model each amino-acid in a protein chain is considered either A (hydrophobic) or B (polar or hydrophilic) as in HP model. In this model, again the amino-acids are linked up with a fixed bond length, but different from the HP model the backbone can continuously bend between any pair of successive links. Additionally, in this model non-consecutive amino-acids interact through a modified Leonard-Jones potential and there is an energy contribution from each bond angle between successive bonds. Therefore, when compared to the HP model, the off-lattice AB model is much more realistic.

In the off-lattice AB model, amino-acids are linked by rigid unit-length bonds to form linear un-oriented polymers that reside in two dimensions. A configuration of n-mer sequence is defined by n angles $\theta_2,...,\theta_{n-1}$,where $-\pi \le \theta_i \le \pi$. A sample configuration is shown in Figure 4.1. It is obvious that $\theta_i = 0$ corresponds to linearity of successive bonds, and positive angles indicate counterclockwise rotation.

The free energy function Φ of a configuration is defined as in (4.1).

$$\Phi = \sum_{i=2}^{n-1} \frac{1-\cos\theta_i}{4} + 4\sum_{i=1}^{n-2}\sum_{j=i+2}^{n}\left[r_{ij}^{-12} - C(\xi_i,\xi_j)r_{ij}^{-6}\right] \qquad \textbf{(4.1)}$$

The first component of the energy function models the backbone bend potentials and it is independent of the A, B sequence, while the second one models the non-bonded

interactions and it depends on the A, B sequence and receives a contribution from the each amino-acid pairs that are not directly attached by a backbone bond. In Figure 4.2, bonded and non-bonded interactions contributing the energy function is shown. Thus, $C(\xi_i,\xi_j)=1$ for AA pairs, $C(\xi_i,\xi_j)=-0.5$ for AB or BA pairs, and $C(\xi_i,\xi_j)=0.5$ for BB pairs. r_{ij} represents the distance between the amino-acid i and the amino-acid j and can be computed by using (4.2).

$$r_{ij} = sqrt\left(\left[1+\sum_{k=i+1}^{j-1}\cos\left(\sum_{l=i+1}^{k}\theta_l\right)\right]^2+\left[\sum_{k=i+1}^{j-1}\sin\left(\sum_{l=i+1}^{k}\theta_l\right)\right]^2\right) \quad \textbf{(4.2)}$$

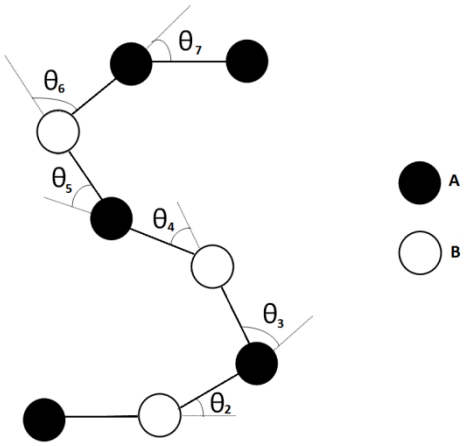

Figure 4.1 : A sample configuration for off-lattice AB model in two-dimensional space.

Since the energy function of the off-lattice AB model is continuous, existing studies generally utilizes improved versions or some combinations of the well-known continuous optimization algorithms (such as PSO, ABC, Tabu Search, DE, etc.) to minimize the energy function given in (4.1). Thus, a ground state configuration of a given amino-acid sequence can be found. Below, a review of some existing methods are given for the solution protein folding problem in off-lattice AB model.

4.1.1 Literature review for the off-lattice AB model

In [59], Canonical Particle Swarm Optimization (Canonical PSO) Algorithm was applied to search the ground state of toy model for protein folding. Experiments were performed both on artificial data and real protein data to evaluate the PSO-based method. The results showed that, when compared to the Simulated Annealing (SA) algorithm the PSO method is feasible and more effective to search for ground state of

toy model. In [60], authors studied two different AB model by means of the multicanonical Monte Carlo method. First, the ability of the algorithm to find lowest-energy conformations was checked. The results were compared with minimum energy values obtained with the energy-landscape paving (ELP) algorithm by measuring the root mean square deviation and a generalized overlap parameter that in addition to torsional degrees of freedom also allows the comparison of bond angles. Authors found very good coincidences for minimum energies and associated conformations for all sequences under study, with the exception of the 34mer in model II and the 55mer in both models. In the latter case, the random walk in the energy space, which is considered as the system parameter, is not sufficient to find the global energy minimum, and a more detailed study of the origin of the free energy barriers, i.e., the identification of an appropriate order parameter, is required. Nonetheless, for all sequences authors obtained much lower values for the respective putative global-energy minimum than formerly quoted in the literature using an off-lattice chain-growth algorithm and also considerably lower values than with the annealing contour Monte Carlo (ACMC) method.

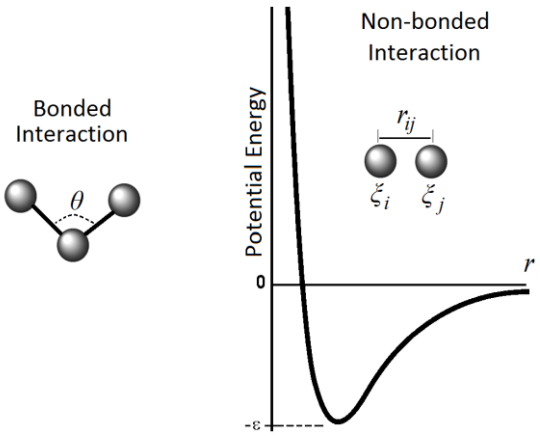

Figure 4.2 : Bonded and non-bonded interactions contributing the energy function of the off-lattice AB model.

In [61], authors proposed a heuristic algorithm for solving the three-dimensional (3D) off-lattice protein folding problem. Based on a physical model, the problem was converted from a nonlinear constraint-satisfied problem to an unconstrained optimization problem which can be solved by the well-known gradient method. To improve the efficiency of the proposed algorithm, a strategy was introduced to generate initial configuration. Computational results showed that the proposed

algorithm could find states with lower energy than previously proposed ground states obtained by nPERM algorithm for all chains with length ranging from 13 to 55. In [62], an improved particle swarm optimization algorithm was proposed for protein folding prediction by using off-lattice AB model. The algorithm introduced a new architecture that is characterized by balancing exploration and exploitation capability of particle swarm optimization algorithm. In the architecture, the population in each generation consists of three parts: an elitist part, an exploitative part, and an explorative part. In the meantime, it makes protein folding prediction more effective with the global search and local search ability in the improved algorithm. The proposed method was applied to sequences with up to 55 monomers and the experimental results were compared with the global energy minimum reported in some literatures. It was shown that, the proposed algorithm is effective to search for the native state of proteins with the lowest free energy. In [63], a heuristic gradient algorithm was proposed for the solution of protein folding problem by using off-lattice AB model. By incorporating extra energy contributions into the original potential function, the constrained optimization problem of AB model was converted into an unconstrained optimization problem which can be solved by the gradient method. After the gradient minimization led to the basins of the local energy minima, the heuristic off-trap strategy and subsequent neighborhood search mechanism were then proposed to get out of local minima and search for the lower-energy configurations. Furthermore, in order to improve the efficiency of the proposed algorithm, authors applied the improved version called the new PERM with importance sampling (nPERMis) of the chain-growth algorithm, pruned-enriched-Rosenbluth method (PERM), to face-centered-cubic (FCC)-lattice to produce the initial configurations. The numerical results showed that the proposed methods are very promising for finding the ground states of proteins. In several cases, authors found the ground state energies lower than the best values reported in the literature. In [64], authors used energy landscape paving (ELP) and subsequent local search (ELP+) to optimize low-energy configurations of a 2D off-lattice AB protein model. In ELP+, the ELP was first applied to search for the low-energy states. After the ELP led to the basins of the local energy minima, the additional degree-of-freedom of bond length was introduced, and the gradient descent method was then used to search for lower energy states near the local minima. The numerical results showed that the proposed methods are very promising for finding the ground states of

proteins. Authors also stated that, although the ELP+ has powerful global optimization performance, it is still easy to get into traps of local minima as a result of random sampling. In [65], a comparison of the Parallel Tempering algorithm to the multicanonical algorithm was performed by using the coarse-grained off-lattice model with the selected monomers. In [66], two different versions of the DE algorithm (basic and adaptive) using a parallel architecture was proposed for the solution of off-lattice AB model both in 2D and 3D space. In the experiments four benchmark sequences, ranging from 13 to 55 amino acids were used. The results for both parallel DE algorithms using both 2D and 3D models were compared with other works in the literature. It was shown that the DE algorithm performs well and the obtained results encouraged further research towards the use of knowledge-based operators to improve further the performance of DE. In [67], authors combined the particle swarm optimization (PSO) and levy flight to solve the problem of protein folding by using 3D AB off-lattice model. To improve the precision and to avoid from the local optima authors introduced the levy flight for the PSO algorithm. In the process of levy flight, the traveling particle undergoes a random sequence of some short jumps, intermittently broken by a single or merely a few long steps. Experimental results showed that the proposed method outperforms other algorithms. In [68], a hybrid algorithm, which combines genetic algorithm based on matrix coding (GAMC) and tabu search (TS) algorithm, was developed for the solution of protein folding problem by using off-lattice AB model. Experiments were performed with Fibonacci sequences and real protein sequences. Experimental results showed that the lowest energy obtained by the proposed GTAMC algorithm is lower than that obtained by previous methods. In [69], authors improved the Tabu Search (TS) algorithm by defining the new neighborhood conformation, tabu object and acceptance criteria of current conformation based on the original TS algorithm. By integrating the heuristic initialization mechanism, the heuristic conformation updating mechanism, and the gradient method into the improved TS algorithm, a heuristic-based tabu search (HTS) algorithm was presented for predicting the two-dimensional (2D) protein folding structure in AB off-lattice model. The tabu search minimization led to the basins of local minima, near which a local search mechanism was then proposed to further search for lower energy conformations. To test the performance of the proposed algorithm, experiments were performed on four Fibonacci sequences and two real protein sequences. The experimental results

showed that the proposed algorithm found the lowest-energy conformations so far for three shorter Fibonacci sequences and renewed the results for the longest one, as well as two real protein sequences, demonstrating that the HTS algorithm is quite promising in finding the ground states for AB off-lattice model proteins. In [70], to solve the protein folding problem by using the off-lattice AB model an improved Artificial Bee Colony (ABC) algorithm, internal feedback strategy based ABC (IF-ABC) was proposed. In the improved ABC algorithm, internal states are fully used in each of the iterations to guide subsequent searching process, and to balance local exploration with global exploitation. Simulations were conducted on artificial Fibonacci sequences and real sequences in the database of Protein Data Bank (PDB). The analysis implied that, IF-ABC is more effective to improve convergence rate than ABC, and can be employed for this specific protein structure prediction issues.

In thesis a new optimization algorithm, namely the Vortex Search algorithm, is proposed for global function optimization which is also utilized for the solution of the protein folding problem by using the off-lattice AB model. In the following sub-section details of this newly proposed optimization algorithm is given.

4.2 A New Optimization Algorithm for the Solution of Protein Folding Problem in Two-Dimensional AB Off-Lattice Model

Many real-life optimization problems are complex in their nature and are difficult to solve. Exact optimization methods usually cannot provide a solution for these types of optimization problems. Some of the properties of such problems, such as high dimensionality, multimodality, epistasis (parameter interaction), and non-differentiability, render exact optimization methods impotent [71]. Hence, the use of some approximate algorithms remains as an alternative approach for the solution of these problems.

Approximate algorithms can be further decomposed into two classes: specific heuristics and metaheuristics [71]. Specific heuristics are problems dependent and designed only for the solution of a particular problem. However, metaheuristics represent a family of approximate algorithms that are more general and thus applicable to a large variety of optimization problems.

The words "meta" and "heuristic" both have their origin in the old Greek: "meta" means "upper level", and "heuristic" denotes the art of discovering new strategies [71]. Heuristic methods provide near optimal solutions in a reasonable amount of computational time without guaranteeing the optimality. Thus, the term metaheuristics can be defined as some intelligent strategies that enhance the efficiency of the heuristic methods [72].

A number of classification criteria have been proposed in the literature for the metaheuristics classification, including the search path that they follow, the use of memory, the type of neighborhood exploration used or the number of current solutions transferred from one iteration round to the next. Among these criteria, the metaheuristic classification, which differentiates between the Single-Solution Based Metaheuristic and the Population-Based Metaheuristic, is often taken to be a fundamental distinction in the literature [71, 73]. Single-solution based metaheuristics are also known as trajectory methods, which are based on a single solution at any time and comprise local search-based metaheuristics such as Simulated Annealing (SA) [74-75], Tabu Search (TS) [76], Iterated Local Search (ILS) [77], Guided Local Search (GLS) [78], Pattern Search (PS) [79], Random Search (RS) [80], and Variable Neighborhood Search (VNS) [81]. However, in population-based metaheuristics, a number of solutions is first created and then updated iteratively until the termination condition is satisfied. Population-based metaheuristics are generally studied under two major groups: Evolutionary algorithms and Swarm-based algorithms. Evolutionary algorithms are based on the notion of natural selection, which can be considered a competition between the species during the evolution process. In these algorithms, individual solutions are selected from a population of solutions according to their fitness value to generate new offspring by using some operators, such as the crossover and the mutation operators. Some well-known examples of the evolutionary algorithms are the Genetic Algorithm (GA) [82], Differential Evolution (DE) [83-84], and Estimation of Distribution Algorithms (EDA) [85]. Swarm-based algorithms are another class of population-based metaheuristics, which are inspired by the collective behavior of species, such as ants, bees, fish and birds. Swarm-based algorithms have the following characteristics: their particles are simple and non-sophisticated agents, they cooperate by an indirect communication medium, and they

perform movements in the decision space [71]. Ant Colony Optimization (ACO) [56], Particle Swarm Optimization (PSO) [86] and Artificial Bee Colony algorithm (ABC) [87-90] are well-known examples of the swarm-based algorithms.

In the past two decades, both the single-solution based and population-based metaheuristics have been successfully applied to many real-world optimization problems. These algorithms produce solutions by exploring the search space efficiently while reducing the effective size of the search. Thus, the success of a metaheuristic method on a given optimization problem is defined by its ability to provide a good balance between the exploration and exploitation. The exploration defines the global search ability of the algorithm, whereas the exploitation is the ability to find the optimum around a near-optimal solution, which can also be considered as the local search ability. Because there is no any information provided regarding the search space in the initial steps, more exploration is required. However, as the algorithm converges to a near-optimal solution, more exploitation is required to tune the current solution towards the optimal one. The main differences between the existing metaheuristics concern the particular manner in which they attempt to achieve this balance [73]. Single-solution based metaheuristics are accepted to be more exploitation oriented, whereas population-based metaheuristics are more exploration oriented.

In this thesis, we propose a new single-solution based metaheuristic, namely, the Vortex Search (VS) algorithm, for the solution of bound-constrained global optimization problems. The proposed algorithm can be studied within the family of the search algorithms that comprises the Random Search and Pattern Search algorithms. The Random Search algorithm (which is also known as the Fixed Step Size Random Search) was proposed by Rastrigin [80], who introduced RS along with a basic mathematical analysis. RS functions by iteratively moving to better positions in the search space that are sampled from a hypersphere surrounding the current position. The PS, which was proposed by Hooke and Jeeves is a type of algorithm similar to the RS [80]. The problem with the above-mentioned algorithms is "the step size", which significantly affects the performance of the algorithms. To overcome this problem, a number of RS variants (e.g., Optimum Step Size Random Search (OSSRS) [91], Adaptive Step Size Random Search (ASSRS) [91], Optimized Relative Step Size Random Search (ORSSRS) [92]) were proposed. However, none

of these algorithms could challenge the performance of population-based metaheuristics. Here, the proposed VS algorithm uses a new adaptive step size adjustment scheme that considerably improves the performance of the search process. Since the search behavior of the VS algorithm is inspired from the vortex pattern, we named the newly proposed algorithm the "Vortex Search" algorithm.

4.2.1 The proposed vortex search algorithm

Single-solution based metaheuristics iteratively apply the generation and replacement procedures from the current single solution [71]. In the generation phase, a set of candidate solutions $C(s)$ is first created from the current solution s, while in the replacement phase a solution $s' \in C(s)$ is selected from $C(s)$ to replace the current solution s. This process iterates until the termination condition is met. Figure 4.3 shows a high-level representation of the single-solution based metaheuristics [71].

Input: Initial solution s_0
$t = 0;$
Repeat
/* Generate candidate solutions (partial or complete neighborhood) from s_t */
Generate($C(s_t)$) ;
/* Select a solution from $C(s)$ to replace the current solution s_t */
s_{t+1} = Select($C(s_t)$) ;
$t = t + 1;$
Until the termination condition is met.
Output: Best solution found

Figure 4.3 : High-level representation of the single-solution based metaheuristics.

Generation of candidate solutions by using some neighborhood structures is of critical importance for the success of the single-solution based metaheuristics. In single-solution based methods one of the main properties searched for a neighborhood is the locality. When small changes are made on the current solution, the neighborhood is said to have a strong locality. In contrast, a weak locality is characterized by a large effect on the solution, which results in the search being a random search in the search space. As mentioned in the previous section, an efficient exploration (a weak locality) is required in the initial steps. Once the algorithm converges to a near-optimal solution, further exploitation (strong locality) is required

to tune the current solution towards to the optimal one. In the proposed VS algorithm, this balance is achieved by using a vortex-like search method.

Let us consider a two-dimensional optimization problem. In a two dimensional space a vortex pattern can be modeled by a number of nested circles. Here, the outer (largest) circle of the vortex is first centered on the search space, where the initial center μ_0 can be calculated using (4.2)

$$\mu_0 = \frac{upperlimit + lowerlimit}{2} \tag{4.2}$$

where *upperlimit* and *lowerlimit* are $d \times 1$ vectors that define the bound constraints of the problem in d dimensional space. Next, a number of neighbor solutions $C_t(s)$, (t represents the iteration index and initially $t = 0$) are randomly generated around the initial center μ_0 in the d-dimensional space by using a Gaussian distribution. Here, $C_0(s) = \{s_1, s_2, ..., s_k\}$ $k = 1,2,...,n$ represents the solutions, and n represents the total number of candidate solutions. In (4.3), the general form of the multivariate Gaussian distribution is given.

$$p(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\} \tag{4.3}$$

where d represents the dimension, x is the $d \times 1$ vector of a random variable, μ is the $d \times 1$ vector of sample mean (center) and Σ is the covariance matrix. If the diagonal elements (variances) of the values of Σ are equal and if the off-diagonal elements (covariance) are zero (uncorrelated), then the resulting shape of the distribution will be spherical (which can be considered circular for a two-dimensional problem, as in our case). Thus, the value of Σ can be computed by using equal variances with zero covariance by using (4.4).

$$\Sigma = \sigma^2 \cdot [I]_{d \times d} \tag{4.4}$$

In (4.4), σ^2 represents the variance of the distribution and I represents the $d \times d$ identity matrix. The initial standard deviation (σ_0) of the distribution can be calculated by using (4.5).

$$\sigma_0 = \frac{\max(upperlimit) - \min(lowerlimit)}{2} \tag{4.5}$$

Here, σ_0 can also be considered as the initial radius (r_0) of the outer circle for a two dimensional optimization problem. Since a weak locality is required in the initial phases, r_0 is chosen to be a large value. Thus, a full coverage of the search space by the outer circle is provided in the initial step. This process provides a bird's-eye view for the problem at hand.

In the selection phase, a solution (which is the best one) $s' \in C_0(s)$ is selected and memorized from $C_0(s)$ to replace the current circle center μ_0. Prior to the selection phase, the candidate solutions must be ensured to be inside the search boundaries. For this purpose, the solutions that exceed the boundaries are shifted into the boundaries, as in (4.6).

$$s_k^i = \begin{cases} rand \cdot (upperlimit^i - lowerlimit^i) + lowerlimit^i, & s_k^i < lowerlimit^i \\ s_k^i, & lowerlimit^i \le s_k^i \le upperlimit^i \\ rand \cdot (upperlimit^i - lowerlimit^i) + lowerlimit^i, & s_k^i > upperlimit^i \end{cases} \quad \textbf{(4.6)}$$

where $k = 1,2,...n$ and $i = 1,2,...,d$ and $rand$ is a uniformly distributed random number. Next, the memorized best solution s' is assigned to be the center of the second circle (the inner one). In the generation phase of the second step, the effective radius (r_1) of this new circle is reduced, and then, a new set of solutions $C_1(s)$ is generated around the new center. Note that in the second step, the locality of the generated neighbors increased with the decreased radius.

In the selection phase of the second step, the new set of solutions $C_1(s)$ is evaluated to select a solution $s' \in C_1(s)$. If the selected solution is better than the best solution found so far, then this solution is assigned to be the new best solution and it is memorized. Next, the center of the third circle is assigned to be the memorized best solution found so far. This process iterates until the termination condition is met. An illustrative sketch of the process is given in Figure 4.4. In this manner, once the algorithm is terminated, the resulting pattern appears as a vortex-like structure, where the center of the smallest circle is the optimum point found by the algorithm. A representative pattern is sketched in Figure 4.5 for a two-dimensional optimization

problem for which the upper and lower limits are between the [-10,10] interval. A description of the VS algorithm is also provided in Figure 4.6.

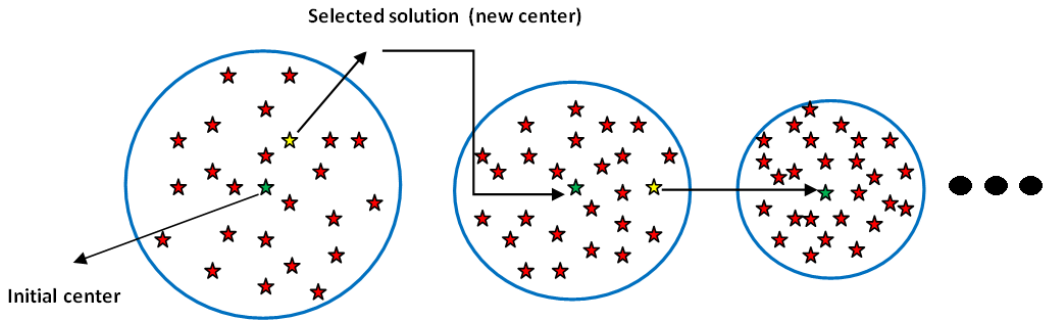

Figure 4.4 : An illustrative sketch of the search process.

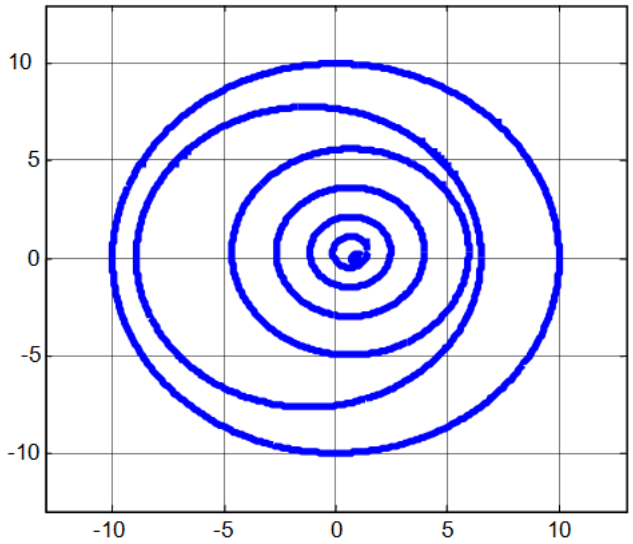

Figure 4.5 : A representative pattern showing the search boundaries (circles) of the VS algorithm after a search process, which has a vortex-like structure.

As indicated by Figure 4.6, the proposed VS algorithm is quite simple. Different from the high-level representation of the single-solution based metaheuristics shown in Figure 4.3, the proposed VS algorithm uses a poor memory (in which only the best solution is memorized) and an additional step in which the radius is iteratively decreased. The use of a poor memory is not new for the single-solution based metaheuristics. The iterated local search algorithm (ILS) and PS algorithm also use a similar type of memory approach [93]. The radius decrement process can be considered as a type of adaptive step-size adjustment process, which is also used in RS (Random Search) algorithms. However, the method by which this adjustment is performed is of critical importance for the success of the algorithms. This process should be performed in such a way that it allows the algorithm to behave in an explorative manner in the initial steps and in an exploitative manner in the latter

steps. To achieve this type of process, the value of the radius must be tuned properly during the search process. In the VS algorithm, the inverse incomplete gamma function is used to decrease the value of the radius during each iteration pass.

Inputs: Initial center μ_0 is calculated by using (4.2)
Initial radius r_0 (or the standard deviation, σ_0) is computed by using (4.5)
Fitness of the best solution found so far $f(s_{best}) = \inf$
$t = 0;$
Repeat
/* Generate candidate solutions by using Gaussian distribution around the center μ_t with a standard deviation (radius) r_t */
Generate($C_t(s)$) ;
If exceeded, then shift the $C_t(s)$ values into the boundaries as in (4.6)
/* Select the best solution from $C_t(s)$ to replace the current center μ_t */
$s^{'}$ = Select($C_t(s)$) ;
if $f(s^{'}) < f(s_{best})$
$s_{best} = s^{'}$
$f(s_{best}) = f(s^{'})$
else
keep the best solution found so far s_{best}
end
/* Center is always shifted to the best solution found so far */
$\mu_{t+1} = s_{best}$
/* Decrease the standard deviation (radius) for the next iteration */
r_{t+1} = Decrease(r_t)
$t = t + 1;$
Until the maximum number of iterations reached
Output: Best solution found so far s_{best}

Figure 4.6 : A description of the proposed VS algorithm.

The incomplete gamma function given in (4.7) most commonly arises in probability theory, particularly in those applications involving the chi-square distribution [94].

$$\gamma(x,a) = \int_0^x e^{-t} t^{a-1} dt \quad a > 0 \qquad \textbf{(4.7)}$$

where $a > 0$ is known as the shape parameter and $x \geq 0$ is a random variable. In conjunction with the incomplete gamma function, its complementary $\Gamma(x,a)$ is usually also introduced (4.8).

$$\Gamma(x,a) = \int_{x}^{\infty} e^{-t} t^{a-1} dt \quad a > 0 \tag{4.8}$$

Thus, it follows that,

$$\gamma(x,a) + \Gamma(x,a) = \Gamma(a) \tag{4.9}$$

where $\Gamma(a)$ is known as the gamma function. There exist many studies in the literature on different proposed methods for the numerical calculation of the incomplete gamma function [95-97].

MATLAB® also provides some tools for the calculation of functions including, gamma function (*gamma*), incomplete gamma function (*gammainc*), and inverse incomplete gamma (*gammaincinv*) function. The inverse incomplete gamma function (*gammaincinv*), computes the inverse of the incomplete gamma function with respect to the integration limit x and represented as *gammaincinv(x,a)* in MATLAB®. In Figure 4.7 the inverse incomplete gamma function is plotted for $x = 0.1$ and $a \in [0,1]$. Here, for our case the parameter a of the inverse incomplete gamma function defines the resolution of the search. By equally sampling a values within $[0,1]$ interval at a certain step size, the resolution of the search can be adjusted. For this purpose, at each iteration, a value of a is computed by using the (4.10).

$$a_t = a_0 - \frac{t}{MaxItr} \tag{4.10}$$

where a_0 is selected as $a_0 = 1$ to ensure a full coverage of the search space at the first iteration, t is the iteration index, and *MaxItr* represents the maximum number of iterations.

Thus, for different values of *MaxItr* , the sampling rate of the a values are changed, thereby changing the resolution of the search. In Figures 4.8a and 4.8b sample plots for the inverse incomplete gamma function are given with respect to the iteration number for a fixed value of $x = 0.1$. In Figure 4.8a, *MaxItr* is selected to be 100, and in Figure 4.8b, *MaxItr* is selected to be 10000, which results a values ranging from 1 down to 0 with a step size of 0.01 and 0.0001, for Figures 4.8a and 4.8b, respectively. Note that $(1/x) \cdot gammaincinv(x,a) \approx 1$ for $a = 1$, which means $gammaincinv(x,a) \approx x$ for $a = 1$. Here, the choice for x determines the value of the

$(1/x)\cdot gammaincinv(x,a)$ that will be reached at approximately half of the number of iterations (Figure 4.8b).

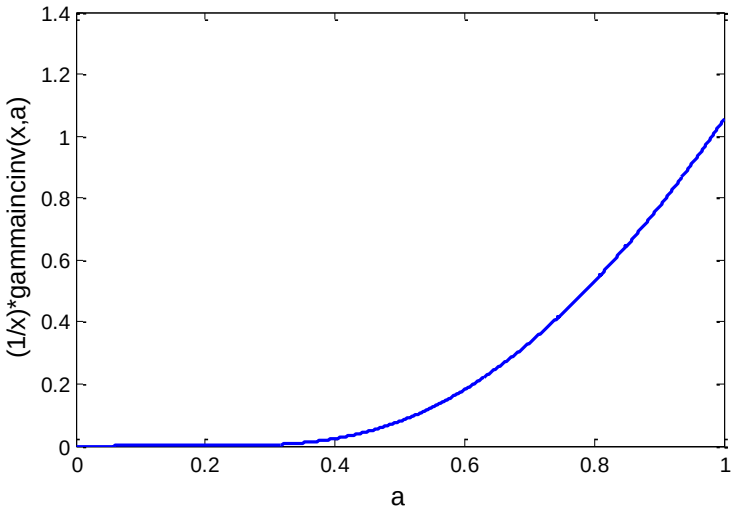

Figure 4.7 : *(1/x) gammaincinv(x,a)* where $x = 0.1$ and $a \in [0,1]$.

From Figure 4.8a and 4.8b, it can be clearly shown that as the number of iterations increases (step size decreases) the value of the function decreases significantly. For example, in Figure 4.8b, for the iteration number of 9978, the resulting value of the function is 1.113e-308, which is a very small number. However, until half of the number of iterations is reached, the function behaves approximately linearly. This behavior allows us to analyze the function in two separate regions.

Let us consider an optimization problem defined within the [-10,10] region. The initial radius r_0 can be calculated with (4.11). Because $a_0 = 1$, the resulting function value is $(1/x)\cdot gammaincinv(x,a_0) \approx 1$, which means $r_0 \approx \sigma_0$ as indicated before.

$$r_0 = \sigma_0 \cdot (1/x)\cdot gammaincinv(x,a_0) \qquad \textbf{(4.11)}$$

By means of (4.5), the initial radius value r_0 can be calculated as $r_0 \approx 10$. In **(4.12)**, a general formula is also given to obtain the value of the radius at each iteration pass.

$$r_t = \sigma_0 \cdot (1/x)\cdot gammaincinv(x,a_t) \qquad \textbf{(4.12)}$$

Here, t represents the iteration index. In Figure 4.9, the change of the radius with respect to the iteration number is given for this case. From Figure 4.9, it can be shown that, for the first half of the number of iterations, the radius changes approximately linearly. Thus, the algorithm has a weak locality in this region. In

contrast, in the other half, the value of the function decreases significantly. Thus, the algorithm has a strong locality in this region.

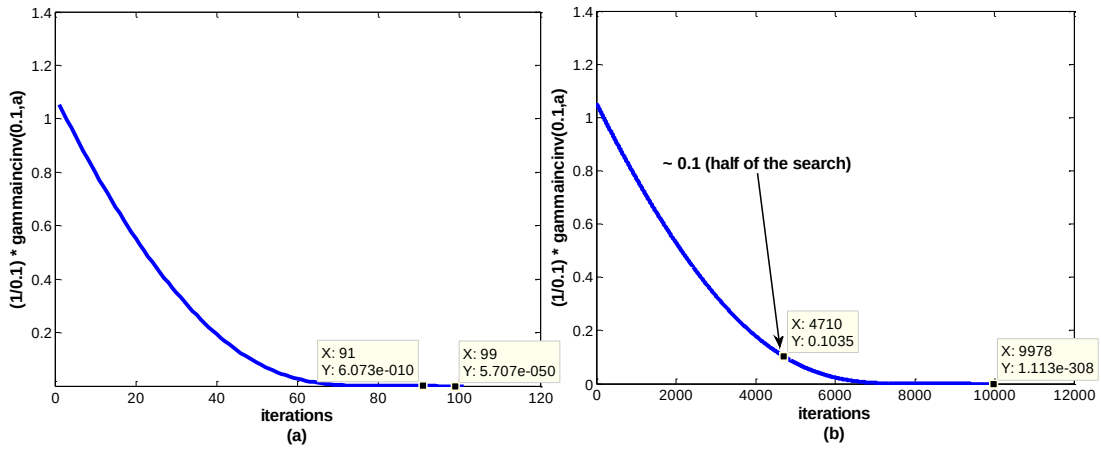

Figure 4.8 : *(1/0.1) gammaincinv(0.1,a)* for (a) *MaxItr* = 100. (b) *MaxItr* = 1000.

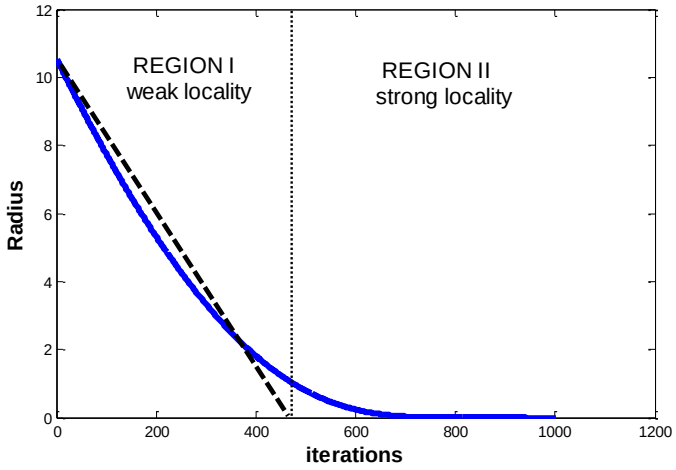

Figure 4.9 : Change of the radius for a problem defined within the [-10,10] interval (step size = 0.001).

As shown above, the inverse incomplete gamma function meets the objectives expected from a successful search. The main drawback of using such a method to tune the radius is the dependence of the convergence speed on the number of iterations. As the number of iterations increase (step size decreases), the resolution of the search also increases. Since the corresponding search space is thoroughly explored, in some cases, this could be seen as an advantage. However, usually, a small step size (e.g., iteration number > 500000) lowers the convergence speed of the problem. Fortunately, most of the problems are solved within the 100000 iterations, which is trivial for the VS algorithm. In Figures 4.10a, 4.10b, and 4.10c, the effect of step size on the search boundaries is given.

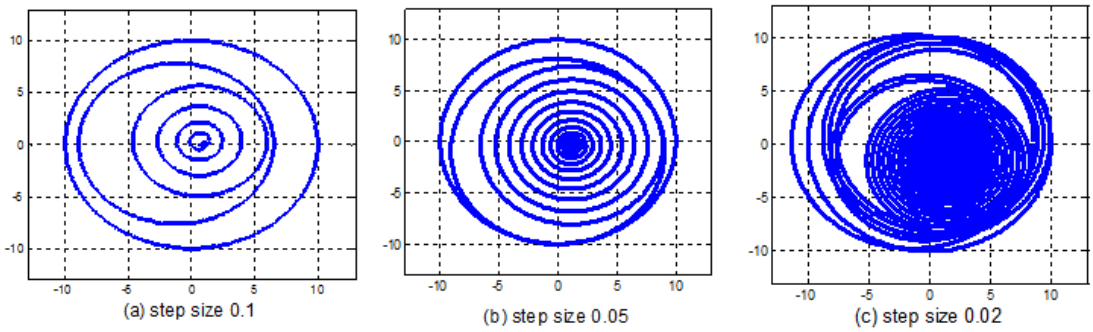

Figure 4.10 : Resolution of the search increases with a decrease in the step size (increased iteration number).

The proposed VS algorithm is quite simple and does not require any additional parameters, except the number of iterations, the number of neighbor solutions, the upper and lower limits of the problem and the dimension of the problem, which are common parameters for all of the other metaheuristics. The only parameter that could be is the x value. However, the x value can also be selected as a fixed value. In Figure 4.11, $(1/x) \cdot gammaincinv(x,a)$ is plotted for different x values of a certain step size.

As found from Figure 4.11, at approximately half of the number of iterations, the value of the $gammaincinv(x,a) \approx x$. Although a detailed analysis is not performed, for a fixed value of $x = 0.1$ the VS algorithm performs well.

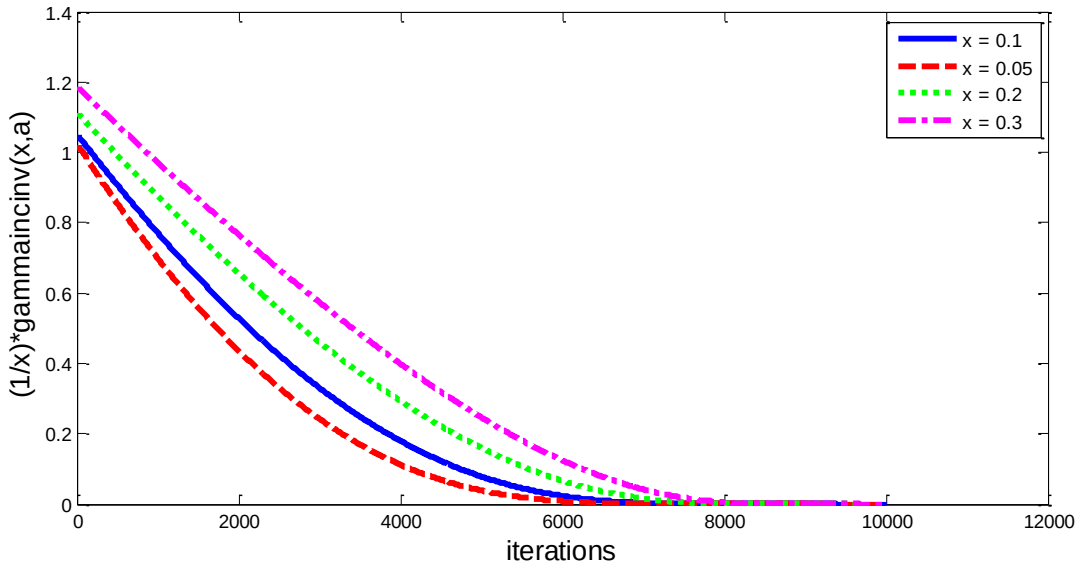

Figure 4.11 : *(1/x) gammaincinv(x,a)* function for different *x* values (step size = 0.0001).

4.2.2 Vortex search algorithm and the off-lattice AB model for the solution of the protein folding problem

The off-lattice AB model's energy function can be minimized by using any optimization algorithm. In this thesis the proposed Vortex Search algorithm and some other well-known metaheuristics are utilized for this purpose. Since a configuration of n-mer sequence is defined by n angles $\theta_2,...,\theta_{n-1}$,where $-\pi \le \theta_i \le \pi$, the Vortex Search algorithm searches for these $n-2$ angles that minimizes the energy of that particular conformation which was given earlier in (4.1). Experimental results are provided in Chapter 6.

4.3 A Modified Energy Function for the Solution of Protein Folding Problem in Two-Dimensional AB Off-Lattice Model

Although, it is a more realistic model of the protein folding problem, even the simplified off-lattice AB model is far to be solved in polynomial time (it is NP complete). In Section 4.1.1 a review of the existing studies that have been proposed to solve the protein folding problem in off-lattice AB model was given. Existing studies, mainly utilizes well-known optimization algorithms or their extensions. When the proposed studies are compared to each other, it can be shown that, the improvements from one study over another mainly arise in terms of the fitness value reached by each method. The computational efficiency or the convergence behaviors of the used methods are usually not compared.

In this thesis, it is aimed to increase the convergence speed of the algorithms to a near optimal or an optimal protein fold by modifying the off-lattice AB model energy function. In their initial study, Stillinger et. al. pointed out that, the given energy function of the off-lattice AB model makes no mention of solvent [12]. They also stated that, one could implicitly modify the energy function to include a solvent effect. Thus, with a small modification on the energy function of the off-lattice AB model, the energy surface of this function can be smoothed and thus, the convergence speed of the algorithms can be significantly improved.

In Figure 4.12a, the known ground state conformation of the protein ABBABBABABBAB is given. As it can be shown from this figure, although there is not too much difference between the conformations given in Figure 4.12a-d, their

energy values are slightly different and far away from the ground state conformation of the protein. In contrast, the energy values of the conformations given in Figure 4.12f-g are relatively closer than the known ground state conformation but the shape of the conformations are totally different from the one given in Figure 4.12a. This means that, the energy landscape of the original energy function has a number of deep valleys and hills. As a result, it is very challenging for the algorithms to find the optimum fold of a protein without trapping into a local minimum point.

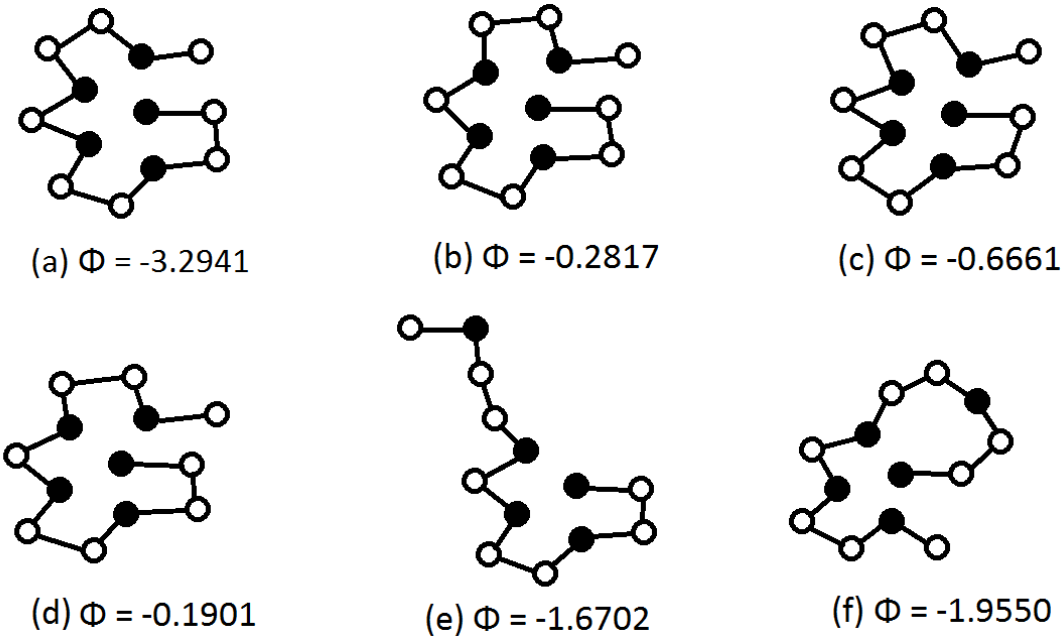

Figure 4.12 : a) Known ground state conformation of the protein ABBABBABABBAB computed with the original energy function. (b-f) Some other conformations and their energies computed by the original energy function.

4.3.1 The modified energy function

In their initial study, Stillinger et. al. pointed out that, the given energy function of the off-lattice AB model makes no mention of solvent [12]. They also stated that, one could implicitly modify the energy function to include a solvent effect. Protein-solvent interactions and the influence of the solvent on protein's thermodynamic properties is in fact very complex and a simulation with a real solvent medium is therefore computationally expensive. Thus, in this study instead of a real solvent medium, the solvent effect is implicitly modeled by an hydrophobic core. It is accepted that the first-order driving force of the protein folding is due to a "hydrophobic collapse" in which those amino-acids which prefer to be shielded from water are driven to the core of the protein, while those which interact more favorably

with water remain on the outside of the protein [98]. Hence, the energy function of the off-lattice AB model is modified to force the amino-acid chain to form a hydrophobic core which in turn favors hydrophobic-hydrophobic interactions. To achieve this, a simple modification is performed on the original energy function.

The modified energy function includes an additional term h which is a function of the distance for non-consecutive AA pair of amino-acids, while for other pairs it is independent of the distance (4.13).

$$h = \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} P_{ij} \qquad \textbf{(4.13)}$$

In (4.13), $P_{ij} = \dfrac{C(\xi_i, \xi_j)}{r}$ for AA pairs, and $P_{ij} = C(\xi_i, \xi_j)$ for other pairs, n represents the total number of amino-acids for a given amino-acid chain. In this manner, AA pairs are forced to form a hydrophobic core. Since the neighbor BB pairs are also welcomed, they are also included in the additional term but they are independent of the distance to prevent a possible formation of a hydrophilic core. Both the AB and BA pairs are also selected to be independent of distance. Otherwise, they form a high energy barrier during search process which prevents the algorithm to visit other configurations. The additional term h helps the algorithm to form a hydrophobic core during the search process. However, using this term alone causes to algorithm to form configurations that have high AA pair interactions by suppressing the effect of the bend potentials. The desired case is to have configurations in which both the bend potential effects and the hydrophobic effects are balanced. Thus, the (4.13) is compensated by an additional term which balances these two forces and forms more realistic configurations (4.14).

$$f(h, \Phi) = h \cdot \frac{n_A - \Phi}{n_A} \qquad \textbf{(4.14)}$$

In (4.14), n_A represents the total number of A type amino-acids for a given amino-acid chain, Φ is the original energy function. Since the energy value of the original function can never exceed the $-n_A$ value, a normalization can be performed by using the total number of A type amino-acids. This normalization process balances the effect of the terms which is required to find desired configurations.

Then the modified off-lattice AB model energy function is given as in (4.15).

$$\Phi_m = \Phi - f(h, \Phi) \quad \textbf{(4.15)}$$

The modified energy function is thought to have a more smoothed energy surface when compared to the original function which has many local minimum points with deep valleys and hills. Thus, it is much easier for algorithms to converge the optimum or a near optimal point during the search process.

5. ALL-ATOM MODELS FOR THE SOLUTION OF PROTEIN FOLDING PROBLEM

5.1 Force Fields and Molecular Dynamic Simulations

The most straightforward way of simulating the protein folding process is to use an all-atom model in conjunction with a force field. A force field is a collection of equations and parameters for use in determining the potential energy of a system [99]. A basic force field includes the energies arise from the deformations of covalent bonds as well as van der Waals interactions, charge–charge interactions, hydrogen bonds, and so on. The most popular force fields currently available include AMBER [13], CHARMM [14], GROMOS [15], ECEPP [16], etc. Clearly, the wide variety for force fields available indicates that one should carefully choose which is the most appropriate for a particular application. However, the general purpose of a force field is to use a variety of energy functions to accurately describe a range of molecular properties and inter molecular interactions [99].

Traditionally, most of the force fields are evaluated through the molecular dynamics simulations in which Newton's equations of motion are numerically solved by calculating the forces acting on atoms and computing accelerations, velocities, and atomic displacements. However, prior to the details of the molecular dynamic simulations, the most basic force field, the molecular mechanics force field, is introduced for better understanding.

5.1.1 The molecular mechanics force field

In the simple molecular mechanics force field, four basic energy components are considered. These basic energy components sketched in Figure 5.1, can be further categorized into two different groups. The bond stretching, angle bending, and bond rotation (torsion) are studied within the bonded interactions, whereas, the electrostatic interactions and van der Waals interactions are studied within the non-bonded interactions. The energy function of the molecular mechanics force field is

given in (5.1). In (5.1), E_s represents the bond stretching, E_b represents the angle bending, E_{tor} represents the bond rotation (or torsion) energy, E_{nb} represents the energy arises from the non-bonded interactions, van der Waals interactions and the electrostatic interactions. More sophisticated force fields may have additional terms but they invariably contains these four components [100].

$$E = E_s + E_b + E_{tor} + E_{nb} \tag{5.1}$$

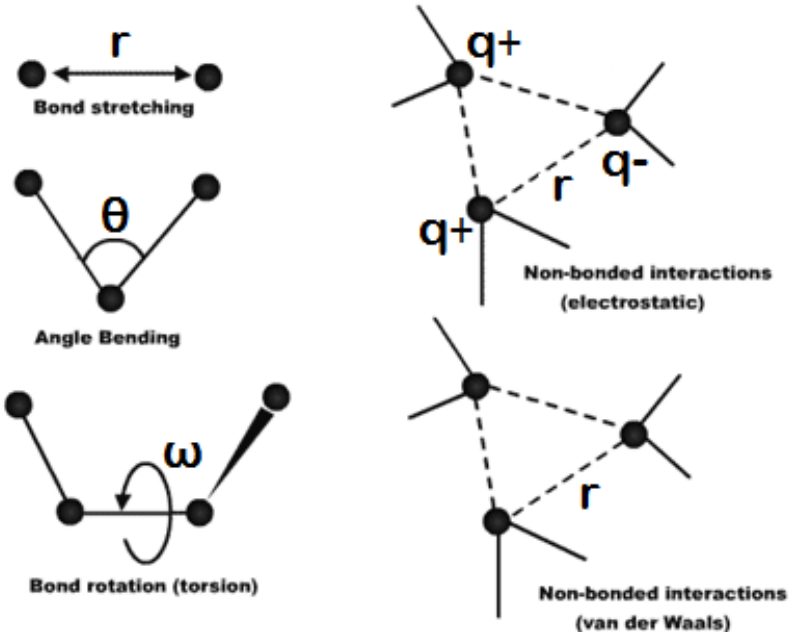

Figure 5.1 : Main energy contributions of the molecular mechanics force field [100].

The stretching energy equation given in (5.2) is based on Hooke's law. The k_b parameter controls the stiffness of the bond spring, while r_o defines its equilibrium length. Unique k_b and r_o parameters are assigned to each pair of bonded atoms based on their types (e.g. C-C, C-H, O-C, etc.). This equation estimates the energy associated with vibration about the equilibrium bond length [101].

$$E_s = \frac{k_b}{2}(r - r_o)^2 \tag{5.2}$$

The bending energy equation given in **(5.3)** is also based on Hooke's law. The k_θ parameter controls the stiffness of the angle spring, while θ_o defines its equilibrium angle. This equation estimates the energy associated with vibration about the equilibrium bond angle [101].

$$E_b = \frac{k_\theta}{2}(\theta - \theta_o)^2 \tag{5.3}$$

The torsion energy given in (5.4) is modeled by a simple periodic function (the cosine function). The torsion energy in molecular mechanics is primarily used to correct the remaining energy terms rather than to represent a physical process. The torsional energy represents the amount of energy that must be added to or subtracted from the Stretching $E_s + E_b + E_{nb}$ energy terms to make the total energy agree with experiment or rigorous quantum mechanical calculation for a model dihedral [101].

$$E_{tor} = \frac{V_n}{2}\left[1 + \cos(n\omega - \varphi)\right] \tag{5.4}$$

The V_n parameter controls the amplitude of the curve, the n parameter controls its periodicity, and φ shifts the entire curve along the rotation angle axis (ω).

The non-bonded energy given in (5.5) represents the pair-wise sum of the energies of all possible interacting non-bonded atoms i and j.

$$E_{nb} = \sum_{i}^{N} \sum_{j=i+1}^{N} \left(4\varepsilon \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \frac{1}{4\pi\varepsilon_o} \frac{q_i q_j}{r_{ij}} \right) \tag{5.5}$$

In **(5.5)**, N represents the number of particles, ε_{ij} represents the depth of the potential well, σ_{ij} represents the finite distance at which the inter-particle potential is zero, r_{ij} represents the distance between the particles, and q_i, q_j represent the charges of the particles and ε_o represents the permittivity of the free space. The non-bonded energy accounts for repulsion, van der Waals attraction, and electrostatic interactions. van der Waals attraction occurs at short range, and rapidly dies off as the interacting atoms move apart by a few Angstroms. The electrostatic contribution is modeled using a Coulomb potential. The electrostatic energy is a function of the charge on the non-bonded atoms, their inter-atomic distance, and a molecular dielectric expression that accounts for the attenuation of electrostatic interaction by the environment (e.g. solvent or the molecule itself) [101].

5.1.2 Molecular dynamics simulations

As the other complicated force fields such as AMBER [13], CHARMM [14], GROMOS [15], etc., the above mentioned simple molecular mechanics force field is used to compute the energy of a given protein conformation. However, to simulate the interactions that occur among the molecules and to observe the movements of atoms of a molecule within a certain period of time interval molecular dynamics simulations are used. In the molecular dynamics simulations, Newton's equation of motion (5.6) is utilized to determine the position of atoms at a feature point in time [102].

$$F = m.a \tag{5.6}$$

where F represents the force, m represents the mass and a represents the acceleration.

In literature, many algorithms are available to implement the laws of motion by using finite difference methods. Some of these algorithms widely use the molecular dynamics simulations. In these algorithms, atomic positions and some other features (e.g. speed and acceleration) are approximated by Taylor series expansions. It is assumed that the next and previous positions of the atoms can be approximated in time by using Taylor series expansions (5.7).

$$\begin{aligned} r(t+\Delta t) &= r(t) + \Delta t \cdot v(t) + \frac{\Delta t^2 \cdot a(t)}{2} + \frac{\Delta t^3 \cdot b(t)}{6} \\ r(t-\Delta t) &= r(t) - \Delta t \cdot v(t) + \frac{\Delta t^2 \cdot a(t)}{2} - \frac{\Delta t^3 \cdot b(t)}{6} \end{aligned} \tag{5.7}$$

where r represents the atomic position, t represents the time, v represents the velocity, a represents the acceleration and b represents the rate of change in acceleration. By adding the Taylor series expansions given in (5.7), the atomic position at $t+\Delta t$ is obtained (5.8). The (5.8) is known as the Verlet algorithm.

$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + \Delta t^2 \cdot a(t) \tag{5.8}$$

As it can be shown from (5.8), in order to calculate the atomic position at $t+\Delta t$, only the acceleration $a(t)$ is needed. The $a(t)$ can easily be calculated by dividing force with the mass and the force can be calculated as the derivative of the energy with respect to the change in position of the atom (5.9).

$$F = -\frac{dE}{dr} \tag{5.9}$$

where F represents the force, E represents the energy and r represents the atomic position.

The basic Verlet algorithm has a disadvantage. Because, it does not explicitly calculate velocity values for the atoms in the system. Therefore, some other algorithms are proposed which are often used in preference to the basic Verlet algorithm. One of these algorithms is known as the "leapfrog" algorithm which calculates the velocities at the half–step (5.10), and the coordinates at the full step (5.11).

$$v\left(t + \frac{\Delta t}{2}\right) = v\left(t - \frac{\Delta t}{2}\right) + \left[\Delta t \cdot a(t)\right] \tag{5.10}$$

$$r(t + \Delta t) = r(t) + \left[\Delta t \cdot v\left(t + \frac{\Delta t}{2}\right)\right] \tag{5.11}$$

Note that, in the leapfrog algorithm the term $\Delta t^2 \cdot a(t)$ does not exist, which means that this algorithm is more accurate than the standard Verlet algorithm. It also requires only one set of atomic coordinates to be stored in memory at any given time, rather than the three sets used with standard Verlet [102].

Another popular algorithm used for the velocity calculation is known as the "Velocity Verlet" algorithm. The Velocity Verlet algorithm has the advantage of calculating both the atomic positions and coordinates at the full step. In this algorithm, firstly the positions at $t + \Delta t$ are calculated (5.12) and then the velocities at the half–step are calculated (5.13).

$$r(t + \Delta t) = r(t) + \Delta t \cdot v(t) + \frac{\Delta t^2 \cdot a(t)}{2} \tag{5.12}$$

$$v\left(t + \frac{\Delta t}{2}\right) = v(t) + \frac{\Delta t \cdot a(t)}{2} \tag{5.13}$$

Once the velocities at the half-step are obtained the positions at the full step can be calculated by using (5.14).

$$r(t + \Delta t) = r(t) + \left[v\left(t + \frac{\Delta t}{2}\right) \cdot \Delta t\right] \tag{5.14}$$

Next, the velocities at $t + \Delta t$ can be calculated by using (5.15).

$$v(t+\Delta t) = v(t) + \frac{\Delta t}{2}\left[a(t) + a(t+\Delta t)\right] \quad \textbf{(5.15)}$$

Although, the molecular dynamic simulations are massively used for the determination of the three-dimensional structures of the proteins, they exist some factors that prevent the continued development of these methods. One of the main factors is the nature of pair-wise interactions that must be computed at each step of the simulations. As a result, once the size of the system increases, the number of pair-wise interactions also increases which brings along a huge computational load. Although, today's machines are powerful enough when compared to the past, the time taken to simulate a protein folding process is still impractical especially for the long chains.

Therefore, in this thesis, a relatively simple force field, the ECEPP (Empirical Conformational Energy Program for Peptides) force field [16], is used to determine the three-dimensional structures of the proteins. Different from the other complicated force fields, in the ECEPP force field molecular dynamics simulations are not used. As discussed previously, the native state of a protein is a low–energy conformation. For this reason it is often desirable to derive low–energy structures computationally. While this can be done by studying how a system evolves over time, there are alternative ways to sample a protein's conformational space. One commonly used method in such work is that of Monte Carlo simulation. In Monte Carlo simulations, a starting structure is passed to the algorithm, and the energy calculated. A change is made to the structure, and the energy is re–calculated. If the new structure has a lower energy than the original, it is carried forward and the process repeated. If the new structure is less stable, there is still the possibility of accepting it, according to a probability known as the Metropolis criterion [102].

The algorithmic steps of the above mentioned Monte Carlo simulations can be implemented by using any other continuous optimization algorithm. Starting from this point of view, in this thesis, we utilized the newly proposed Vortex Search algorithm [18] in conjunction with the ECEPP force field to determine the three-dimensional structures of the proteins.

5.2 The ECEPP Force Field and the SMMP Software Package

5.2.1 The ECEPP force field

In the ECEPP force field, the lengths of covalent bonds, along with the bond angles, are taken to be constant at their equilibrium value, and the independent degrees of freedom become the torsional angles of the system. The potential energy function E is the sum of the electrostatic term E_c, Lennard-Jones term (or van der Waals) E_{LJ}, and the hydrogen-bonding term E_{HB} for all pairs of peptides, together with the torsion term E_{tor} for all torsion angles (5.16 - 5.20).

$$E = E_c + E_{vdw} + E_{HB} + E_{tor} \quad \textbf{(5.16)}$$

$$E_c = \sum_{(i,j)} \frac{332 q_i q_j}{\varepsilon r_{ij}} \quad \textbf{(5.17)}$$

$$E_{LJ} = \sum_{(i,j)} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right) \quad \textbf{(5.18)}$$

$$E_{HB} = \sum_{(i,j)} \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right) \quad \textbf{(5.19)}$$

$$E_{tor} = \sum_{l} U_l \left(1 \pm \cos(\eta_l \chi_l)\right) \quad \textbf{(5.20)}$$

where r_{ij} is the distance between atoms i and j, and χ_l is the torsion angle for chemical bond l. The bond lengths and bond angles (which are hard degrees of freedom) are fixed at experimental values, and dihedral angles φ, ψ, ω and χ_i are independent variables. The various parameters (q_i, A_{ij}, B_{ij}, C_{ij}, D_{ij}, U_l, and η_l) were determined by a combination of a priori calculations and minimization of the potential energies of the crystal lattices of single amino acids [103].

5.2.2 The SMMP software package

The SMMP (Simple Molecular Mechanics for Proteins) is a software package developed for the simulation of proteins which implements ECEPP and FLEX force fields. The first SMMP package were presented in 2001 by Eisenmenger et. al [104]. After then, two improved versions of the SMMP software package has also been proposed [105-106].

The initial package was developed in Fortran and it implements both the ECEPP/2 and ECEEP/3 force fields. In the initial package, a set of energy minimization routines and modern Monte Carlo algorithms were also included. The package is fast and can easily be exploited even on a single PC. Thus, it allows researchers and students in a simple and inexpensive way to become familiar with protein simulation techniques.

In the last version SMMP v3.0 of the software package [106], Python bindings are also added to provide an access to the software package from Python. In this version, a Pearu Peterson's program f2py, which is part of SciPy, is used to create Python bindings to SMMP. If f2py is installed, `make pybind` will build the Python bindings. The bindings are stored in the library file smmp.so, which is completely self contained. In Figure 5.2, the hierarchy of Python software development by using the SMMP software package is provided.

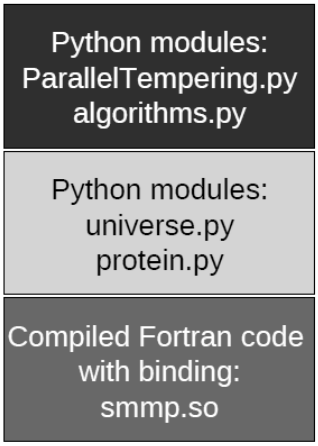

Figure 5.2 : The hierarchy of Python software development by using the SMMP software package.

In this thesis, by utilizing the Python binding tools provided in the SMMP package, the proposed Vortex Search algorithm is implemented in Python under the Linux Mint 16.0 and then combined to the SMMP package to determine the three-dimensional structures of the proteins. Experimental results are provided in Chapter 6.

6. COMPUTATIONAL RESULTS AND DISCUSSION

6.1 Computational Results of the Proposed Reinforcement Learning Based Method for the Two Dimensional HP Lattice Model

In this sub-section experimental results for both scenarios mentioned in Chapter 3 for the 2D HP lattice model are given and results are compared to the standard Q-learning algorithm.

To allow the agent to form self-avoiding configurations (to guide the agent), first an MxM grid is created and the agent is located in the center of this grid. Without loss of generality the agent's first move could be chosen to any direction and in this study the agent is first moved to 'R'. At each move of the agent the corresponding state-action pairs are found from the AQ-table and the resulting state transitions are stored. Thus, at the end of the tour the AQ-values for these transitions can be updated by using the total reward taken by the agent. In Figure 6.1, a schematic representation of the agent's moves is given. As it can be shown in Figure 6.1, the state-action pairs can be further reduced. Because, the agent has 3 or less (due to the self-avoiding constraint) possible moves when it is at a certain position.

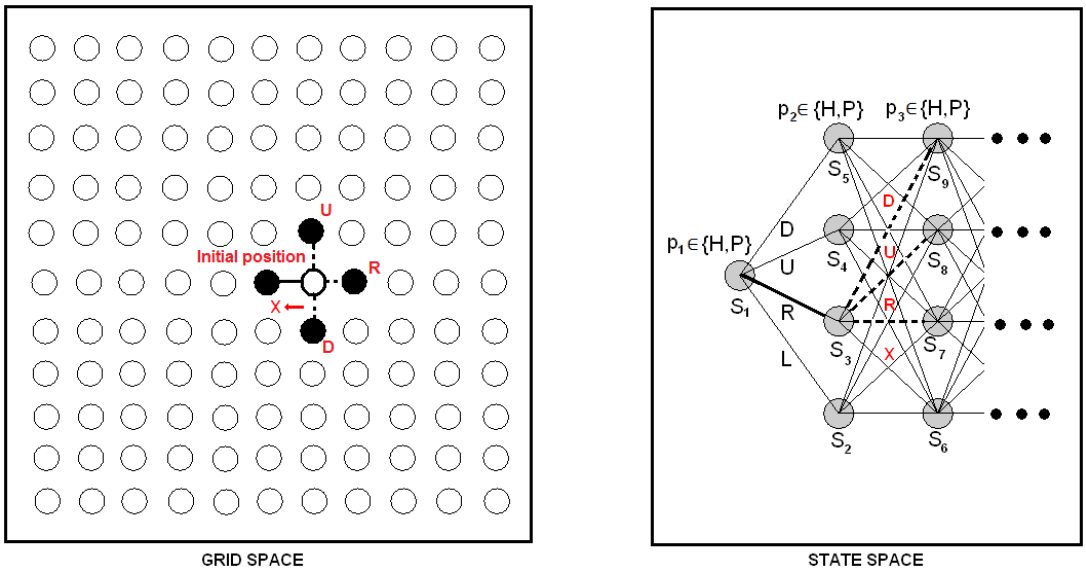

Figure 6.1 : Agent's move over the grid space and the corresponding state transitions.

Given the above grid and state transitions in Figure 6.1, k agents cooperate to find optimum protein folds by using the Ant-Q algorithm with the following parameters: $\delta = 1, \beta = 2, q_0 = 1, \alpha = 0.1,$ and $\gamma = 0.3.$ The number of agents k is chosen to be equal to the length of corresponding amino acid sequence.

Having established the parameters for the Ant-Q algorithm, let us now examine the results for the scenarios given in Chapter 3, respectively.

Scenario 1: As it is stated in Chapter 3, in the first scenario the agent tries to find the optimum fold for a particular amino acid sequence. Let us again consider the sequences $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH and $\mathcal{P}_2$ = HPHPPH. In Figure 6.2, one of the optimum configurations found by Ant-Q algorithm and the resulting state-action space for the $\mathcal{P}_2$ = HPHPPH is given.

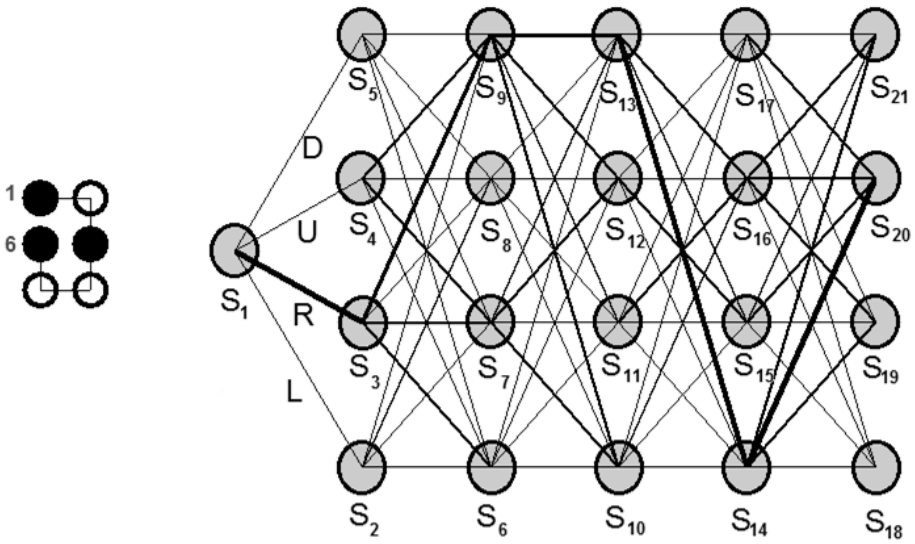

Figure 6.2 : Optimum fold and the resulting state-action space for $\mathcal{P}_2$ = HPHPPH.

The edges between the states shown in Figure 6.2 are weighted according to the resulting AQ-values. Thus, the state transitions for the optimum fold of amino acid sequence $\mathcal{P}_2$ = HPHPPH can be given as S1-S3-S9-S13-S14-S20. From Table 3.2 it can be shown that the resulting sequence of directions is RDDLU for the amino acid sequence $\mathcal{P}_2$ = HPHPPH as verified in Figure 6.2. Note that, the resulting sequence of directions is not directly encoded as in the existing method but it is combined with the characteristics of the amino acids. Thus, for the longer sequence of $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH (where $\mathcal{P}_2 \subset \mathcal{P}_1$) a part of the state action space is already learned by the agent.

The Ant-Q algorithm is able to find the optimum configuration in nine out of ten experiments for the sequence $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH. All of the experiments are performed in MATLAB and the average computation time of ten experiments is 46.64s in a Pentium IV 3GHz 1.5GB RAM PC. In Figure 6.3, one of the optimum configurations and the corresponding fitness evaluation for each iteration is given. As it can be shown in Figure 6.3, the Ant-Q algorithm found the optimum configuration in 35 iterations which takes 34s of computational time. In Table 6.1 and Figure 6.4 solutions found by Ant-Q algorithm for a number of benchmark amino acid sequences are also provided. In literature, there exist many studies proposed to find out the optimum fold for these sequences. In these studies, the problem is generally handled in two phases. In the first phase an algorithm is used to find a local minimum protein configuration (like Ant-Q) and then another algorithm (local search methods) tries to further improve this local configuration to find out the global optimum fold. However, in this study it is mainly aimed to show the advantages of the proposed state action space. Thus, the second phase is not used and for this reason, except the Seq1 the obtained configurations are local minimums.

Table 6.1 : Solutions found by Ant-Q for some benchmark sequences.

Seq ID	Length	Sequence	Optimum	Ant-Q Solution
Seq1	24	HHPPHPPHPPHPPHPPHPPHPPHH	-9	-9
Seq2	36	PPPHHPPHHPPPPPHHHHHHHPPHH PPPPHHPPHPP	-14	-13
Seq3	48	PPHPPHHPPHHPPPPPHHHHHHHHH HPPPPPPHHPPHHPPHPPHHHHH	-23	-19
Seq4	64	HHHHHHHHHHHHPHPHPPHHPPHH PPHPPHHPPHHPPHPPHHPPHHPPHP HPHHHHHHHHHHHH	-42	-36

Czibula et.al. [44-47] also used the sequence $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH for their studies. However, as mentioned before with the existing method it is not possible to create state-action space at the beginning of the algorithm. So, in this study to compare the performance of the proposed method with the standard Q-learning algorithm, again the newly proposed state-action space is used with delayed reinforcement.

As mentioned before, in the standard Q-learning algorithm the agent chooses an action randomly with a probability of 0.25. When the amino acid sequence is not

long enough these random movements converges the agent to optimum solution in a number of iterations.

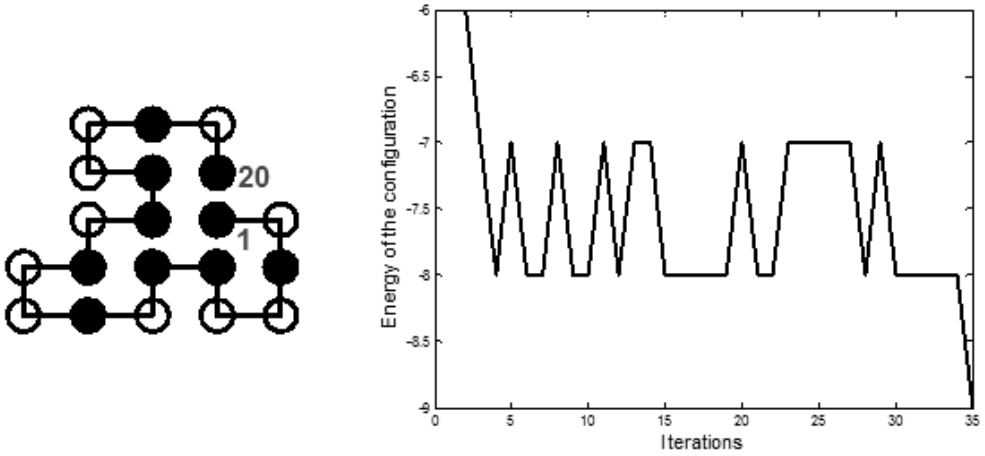

Figure 6.3 : Optimum configuration and corresponding fitness evaluation for the sequence $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH found by Ant-Q algorithm.

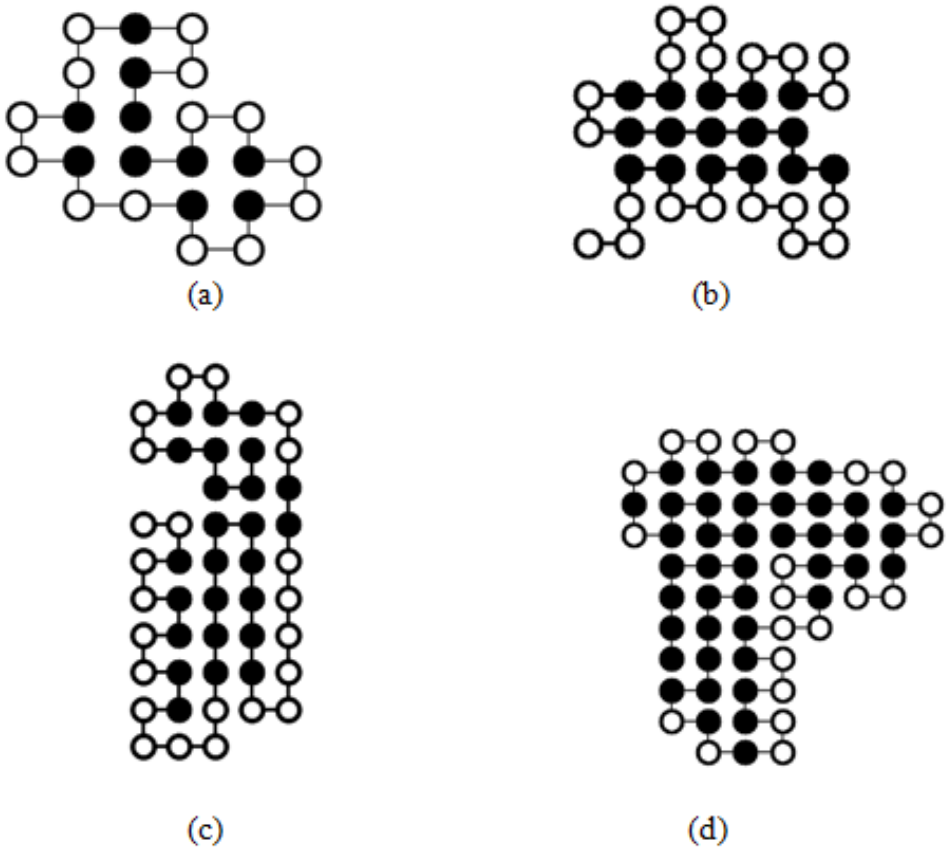

Figure 6.4 : Example solutions found by Ant-Q algorithm for the sequences given in Table : 6.1 : (a) Seq1. (b) Seq2. (c) Seq3. (d) Seq4.

However, as the length of the amino acid sequence increases, the number of the configurations that must be visited by the agent also increases dramatically. Thus, in such a situation agent can not be able to find the optimum configuration. For example, the agent finds the optimum configuration for the $\mathcal{P}_2$ = HPHPPH. However, when it comes to sequence $\mathcal{P}_1$ = HPHPPHHPHPPHPHHPPHPH the agent fails to find the optimum configuration even in 100000 iterations. Because, the state space is not thoroughly explored by the agent even in 100000 iterations with the standard Q-learning algorithm.

Scenario 2: In this case, the agent learns the state-action space for all of the combinations of an *n* length amino acid sequence. As it can be shown from Figure 6.5, for $n = 6$ there are $2^6 = 64$ sequences that agent must learn.

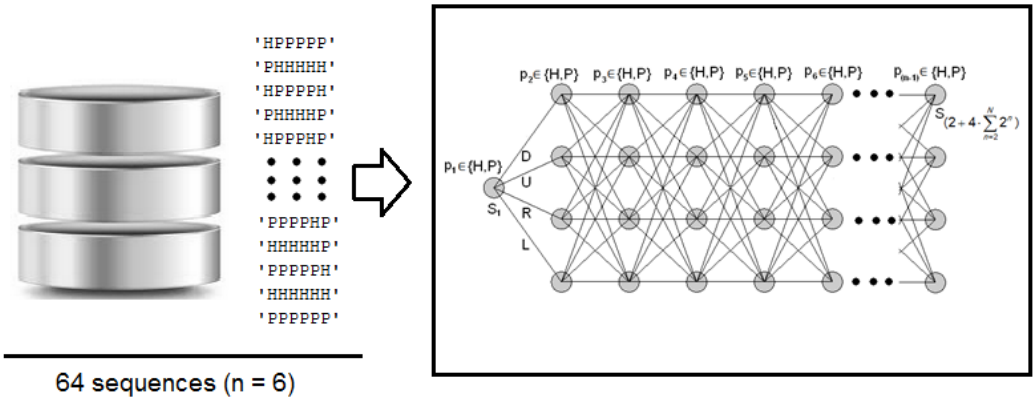

Figure 6.5 : Agent learns the state-action space for all of the n length sequences in Scenario-2.

Let us consider that the agent wants to learn state-action space for $n = 6$. For this purpose, first the state-action space which consists of $S = (2 + 4 \cdot \sum_{n=2}^{6} 2^n) = 498$ states is created and then the AQ-table is initialized according to the possible state transitions. Note that, in this case the number of transitions are much more than the one in Scenario-1. Because, all of the sequences are considered. Then, the agent picks up one of the sequences and tries to learn state-action space for this particular sequence in a number of iterations. In this study, for $n = 6$ the maximum number of iterations is selected as 100 for each sequence. Once the agent learns the state-action space for a particular sequence the resulting AQ-table is stored for the next sequence. Thus, additively updating the AQ-table, at the end of the learning process a universal AQ-table is obtained. This table is called universal because, after the learning

process for any sequence of length 6, this table can guide the agent to form optimum configurations.

In Figure 6.6, an example for the sequence $\mathcal{P}_2$ = HPHPPH is given. Note that, in the proposed method in order to find the optimum configuration all the agent needs is the amino acid sequence itself but not the directions as in the existing methods. In the existing methods the state space only encodes the folding directions of a particular amino acid sequence. So, it is not possible to obtain optimum configuration for any sequence after the learning process.

6.1.1 Discussion on the proposed reinforcement learning based method

The newly proposed state-action space allows the protein folding problem to be studied by using reinforcement learning methods. Compared to the existing one the newly proposed state-action space has several advantages. First, the size of the state-action space is less dependent to the length of the amino-acid sequence. In the existing methods, the size of the state-action space is highly affected by the amino-acid sequence length. Moreover, if the problem is handled in 3D lattice structures the size of the state-action space will be further increased. Because in 3D lattice structures there are two additional positions that agent could be moved. Thus, with the existing definition the size of the state-action space will change by 6^n.

Additionally, the Ant-Q algorithm is shown to be a good candidate for the solution of the protein folding problem. It is shown that the Ant-Q algorithm overwhelmingly outperforms the standard Q-learning algorithm which is used in the existing methods.

Besides its advantages, the proposed state-action space has also some limitations. Especially, for long sequences the resulting state-action space (for scenario-2) is still huge. As a further study, a recursive algorithm could be proposed to predict the optimum fold of $n+a$ length sequences, where $a \geq 1$, from the learned AQ-values of n length sequences.

Future directions include the design of the above mentioned recursive method to handle the problems for long sequences. Additionally, a comparison of the swarm based algorithms over the newly proposed state-action space in 2D and 3D will also be studied.

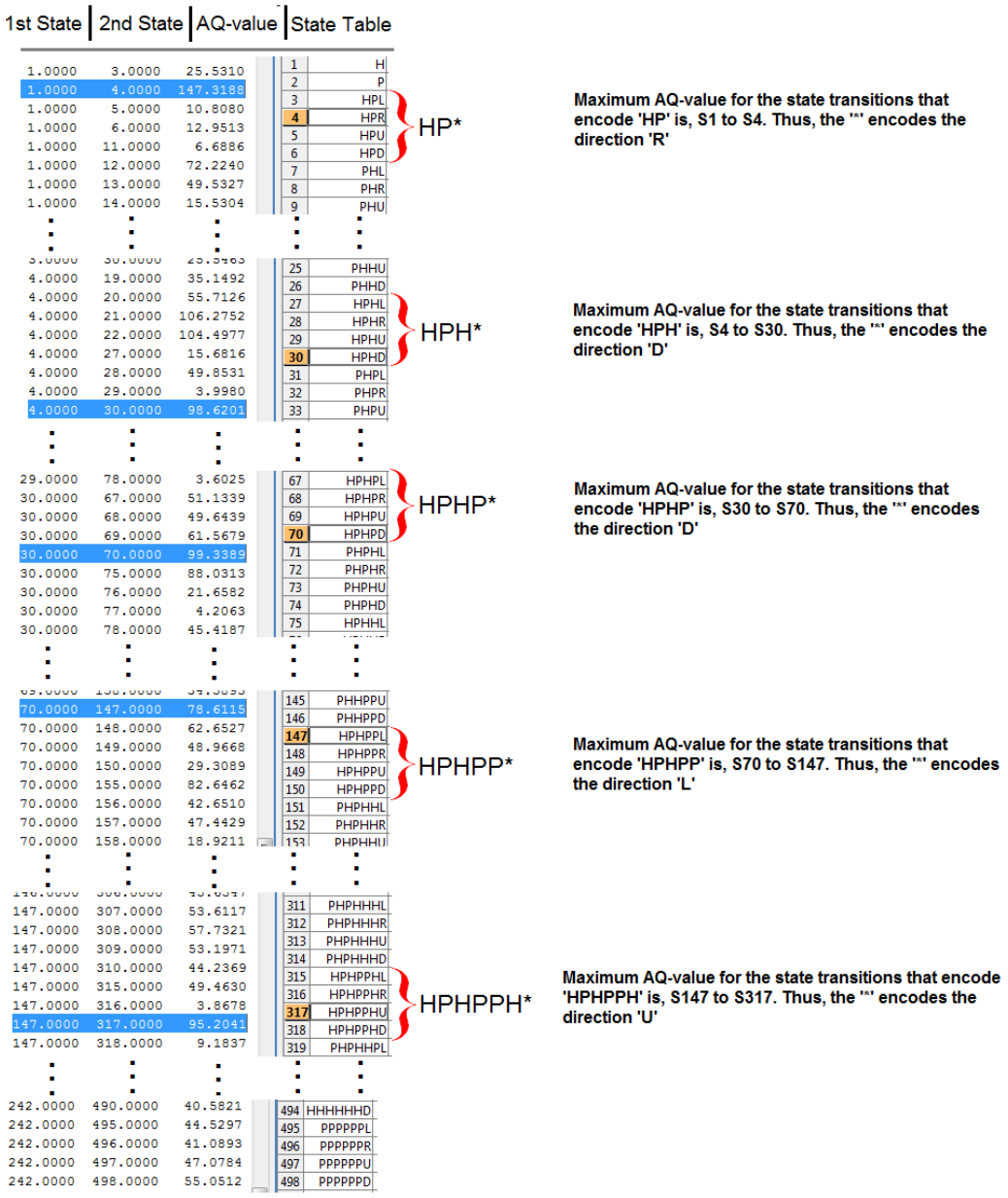

Figure 6.6 : After the learning process the universal AQ-table guides the agent to form the optimum configuration for the sequence $\mathcal{P}_2$ = HPHPPH. The resulting state transition chain is 1-4-30-70-147-317 which encodes the sequence of directions RDDLU as in Figure : 6.3.

6.2 Computational Results of the Off-Lattice AB Model

6.2.1 Computational results of the proposed Vortex Search algorithm on the benchmark numerical function set

The proposed VS algorithm is tested on 50 benchmark functions that are obtained from the study performed by Karaboğa and Akay [107]. In their study, Karaboğa and Akay compared the performance of the ABC algorithm to the GA, PSO, and DE

algorithms. By using the same functions, in this study, the performance of the proposed VS algorithm is compared to the algorithms SA, PS, PSO2011 and ABC. SA and PS are two well-known single-solution based algorithms, and PSO2011 [108 , 109] is an extension of the standard PSO algorithm. In the literature, in addition to the newly proposed optimization algorithms [110-111], some extensions of the above-mentioned algorithms are also proposed [112-121]. However, using well-known standard algorithms in the comparisons enables the interpretation of the results over a larger group.

To evaluate the performances of the algorithms, two different set of experiments are performed. In the first set of experiments, the overall performances of the algorithms are studied for a constant number of iterations. After a certain number of iterations, the algorithms are evaluated according to the mean and best fitness values found for each benchmark function. In the second set of experiments, the convergence behavior of the algorithms is studied. For this purpose, a different number of iterations is selected and the algorithms are run to evaluate the mean fitness value found for each case. Thus, an iteration based convergence behavior of the algorithms is obtained. In literature, a time-based comparison of the optimization algorithms is also used to evaluate the convergence behavior [112]. However, in our case, this comparison was not possible. Because, the proposed VS algorithm is an iteration-dependent algorithm, to perform the VS algorithm, one must set a certain number of iterations. However, the proposed VS algorithm behaves quite different for different numbers of iterations. Since the resolution of the search differs for different number of iterations, the VS algorithm gives quite different results in a time-based comparison.

6.2.1.1 Benchmark functions

The functions used in the test are listed in Table 6.2, and include many different types of problems, such as unimodal, multimodal, regular, irregular, separable, non-separable, and multidimensional. For a list of constant parameters used in some functions, please refer to Appendix A.

A function is called unimodal if it has one global optimum point with no or a single local optimum point, whereas a multimodal function is a function with many local optimum points. For multimodal functions, an optimization algorithm is tested for its

ability to avoid local optimum points. If the algorithm has a poor level of exploration ability, it cannot thoroughly explore the search space and is thus likely to select a local optimum point. Functions that have a flat search space (Stepint, Matyas, PowerSum) are also difficult for the algorithms, as in the multimodal functions because, a flat search space does not provide any gradient information to direct the algorithm towards the optimum point [107]. Another group of functions includes separable and non-separable functions. A p-variable separable function can be expressed as the sum of p functions of one variable. Non-separable functions cannot be written in this form because they have an interrelation among their variables [107]. Therefore, optimization of the non-separable functions is more difficult than the separable ones.

In a high-dimensional space, algorithms usually face another problem, which is known as the curse of dimensionality. As the dimensionality of the problem increases, the volume of the space increases so rapidly that the data points become sparse. A search algorithm that is effective in small dimensions might exhibit poor performances in such high dimensional spaces. Therefore, it is common to test algorithms to evaluate their ability of finding global optima in high dimensional spaces. In some functions, the global minima is very small when compared to the entire search space, such as the functions of Easom, Michalewicz (m = 10), and Powell. For problems such as Perm, Kowalik, and Schaffer, the global minimum is located very close to the local minima. If the algorithm cannot adapt to the direction changes in the functions having a narrow curving valley (e.g., Beale, Colville), it will fail when applied to these types of problems [122].

Another problem that algorithms suffer is the scaling problem, with a difference of many magnitude orders between the domain and the frequency of the hypersurface (GoldStein-Price, Trid) [123]. Some functions, such as Fletcher-Powell and Langerman, are non-symmetrical and their local optima are randomly distributed. Because the objective functions have no implicit symmetry, optimization of these functions is difficult for certain algorithms. A quartic function is padded with random noise (either Gaussian or uniform), which ensures the algorithm to never produce the same value on the same point. This function allows us to test algorithms for their performance on the noisy data. The algorithms that do not perform well on this function will also fail on noisy data.

Table 6.2 : Benchmark functions used in experiments D: Dimension, C: Characteristics, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
F1	[-5.12, 5.12]	5	US	Stepint	$f(x) = 25 + \sum_{i=1}^{5} \lceil x_i \rceil$
F2	[-100, 100]	30	US	Step	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
F3	[-100, 100]	30	US	Sphere	$f(x) = \sum_{i=1}^{n} (x_i)^2$
F4	[-10, 10]	30	US	SumSquares	$f(x) = \sum_{i=1}^{n} (ix_i)^2$
F5	[-1.28, 1.28]	30	US	Quartic	$f(x) = \sum_{i=1}^{n} (ix_i)^4 + \text{random}[0,1)$
F6	[-4.5, 4.5]	5	UN	Beale	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$
F7	[-100, 100]	2	UN	Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2 - (x_2-\pi)^2)$
F8	[-10, 10]	2	UN	Matyas	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$
F9	[-10, 10]	4	UN	Colville	$f(x) = 100(x_1^2 - x_2)^2 + (x_1-1)^2 + (x_3-1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2-1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1)$
F10	[-D^2,D^2]	6	UN	Trid6	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F11	[-D^2,D^2]	10	UN	Trid10	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F12	[-5,10]	10	UN	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^{\,2} + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$
F13	[-4,5]	24	UN	Powell	$f(x) = \sum_{i=1}^{n/k} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$
F14	[-10, 10]	30	UN	Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F15	[-10, 10]	30	UN	Schwefel 1.2	$f(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$
F16	[-30, 30]	30	UN	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$
F17	[-10, 10]	30	UN	Dixon-Price	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
F18	[-65.536, 65.536]	2	MS	Foxholes	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})6}\right]^{-1}$
F19	[-5,10]x[0,15]	2	MS	Branin	$f(x) = (x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$
F20	[-100, 100]	2	MS	Bohachevsky1	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
F21	[-10, 10]	2	MS	Booth	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
F22	[-5.12, 5.12]	30	MS	Rastrigin	$f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
F23	[-500, 500]	30	MS	Schwefel	$f(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$
F24	$[0, \pi]$	2	MS	Michalewicz2	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$
F25	$[0, \pi]$	5	MS	Michalewicz5	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$

Table 6.2 (continued) : Benchmark functions used in experiments D: Dimension, C: Characteristics, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
F24	$[0, \pi]$	2	MS	Michalewicz2	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$
F25	$[0, \pi]$	5	MS	Michalewicz5	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$
F26	$[0, \pi]$	10	MS	Michalewicz10	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$
F27	[-100, 100]	2	MN	Schaffer	$f(x) = 0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$
F28	[-5, 5]	2	MN	Six Hump Camel Back	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
F29	[-100, 100]	2	MN	Bohachevsky2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
F30	[-100, 100]	2	MN	Bohachevsky3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
F31	[-10, 10]	2	MN	Shubert	$f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$
F32	[-2, 2]	2	MN	GoldStein-Price	$f(x) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \cdot \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$
F33	[-5, 5]	4	MN	Kowalik	$f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$
F34	[0, 10]	4	MN	Shekel5	$f(x) = -\sum_{i=1}^{5}\sum_{j=1}^{4}\left[(x_j - a_{ij})(x_j - a_{ij})^T + c_i\right]^{-1}$
F35	[0, 10]	4	MN	Shekel7	$f(x) = -\sum_{i=1}^{7}\sum_{j=1}^{4}\left[(x_j - a_{ij})(x_j - a_{ij})^T + c_i\right]^{-1}$
F36	[0, 10]	4	MN	Shekel10	$f(x) = -\sum_{i=1}^{10}\sum_{j=1}^{4}\left[(x_j - a_{ij})(x_j - a_{ij})^T + c_i\right]^{-1}$
F37	[D, D]	4	MN	Perm	$f(x) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i^k + \beta)((x_i/i)^k - 1)\right]^2$
F38	[0, D]	4	MN	PowerSum	$f(x) = \sum_{k=1}^{n}\left[(\sum_{i=1}^{n} x_i^k) - b_k\right]^2$
F39	[0, 1]	3	MN	Hartman3	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$
F40	[0, 1]	6	MN	Hartman6	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$
F41	[-600, 600]	30	MN	Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$

Table 6.2 (continued) : Benchmark functions used in experiments D: Dimension, C: Characteristics, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
F42	[-32, 32]	30	MN	Ackley	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$
F43	[-50, 50]	30	MN	Penalized	$f(x) = \frac{\pi}{n}\left\{\begin{array}{c} 10\sin^2(\pi y_1) \\ +\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2 \end{array}\right\} + \sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i = 1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$
F44	[-50, 50]	30	MN	Penalized2	$f(x) = 0.1\left\{\begin{array}{c} \sin^2(\pi x_1) \\ +\sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+ \\ (x_n-1)^2\left[1+\sin^2(2\pi x_n)\right] \end{array}\right\} + \sum_{i=1}^{n} u(x_i,5,100,4)$
F45	[0, 10]	2	MN	Langerman2	$f(x) = -\sum_{i=1}^{m} c_i(\exp(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2)\cos(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2))$
F46	[0, 10]	5	MN	Langerman5	$f(x) = -\sum_{i=1}^{m} c_i(\exp(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2)\cos(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2))$
F47	[0, 10]	10	MN	Langerman10	$f(x) = -\sum_{i=1}^{m} c_i(\exp(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2)\cos(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2))$
F48	$[-\pi,\pi]$	2	MN	Fletcher Powell2	$f(x) = \sum_{i=1}^{n}(A_i-B_i)^2$ $A_i = \sum_{j=1}^{n}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j),\ B_i = \sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)$
F49	$[-\pi,\pi]$	5	MN	Fletcher Powell5	$f(x) = \sum_{i=1}^{n}(A_i-B_i)^2$ $A_i = \sum_{j=1}^{n}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j),\ B_i = \sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)$
F50	$[-\pi,\pi]$	10	MN	Fletcher Powell10	$f(x) = \sum_{i=1}^{n}(A_i-B_i)^2$ $A_i = \sum_{j=1}^{n}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j),\ B_i = \sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)$

6.2.1.2 Algorithm settings

Population based metaheuristics (ABC, PSO2011) are selected to have a population size of 50, which is also the number of neighborhood solutions of the proposed VS algorithm. The SA algorithm always performs with a single solution, and the PS algorithm creates its own neighbor vectors (pattern). The acceleration coefficients (

c_1 and c_2) of the PSO2011 algorithm are both set to 1.8, and the inertia coefficient is set to 0.6, as in [107]. The *limit* value for the ABC algorithm is determined as *limit = SN * D,* where *SN* represents the number of food sources and *D* represents the dimension. As mentioned before in the previous section, the proposed VS algorithm is a parameter-free algorithm. There are no additional parameters for the VS algorithm.

For the first set of experiments, the maximum number of iterations is selected as 500000 to evaluate the overall performances of the algorithms. For the second set of experiments, the number of iterations is selected as 100, 1000 and 10000 to evaluate the convergence behavior of the algorithms.

6.2.1.3 Overal performances of the algorithms

In the first set of experiments, the proposed VS algorithm is compared to the SA, PS, PSO2011 and ABC algorithms by using the 50 benchmark functions given in Table 6.1. For each algorithm, 30 different runs are performed, and the mean and the best values are recorded. The maximum number of iterations is selected to be 500000, as mentioned previously. For the SA and PS algorithms, the MATLAB® Global Optimization Toolbox is used, and the other algorithms are also coded in MATLAB®. For the PSO2011, ABC and VS algorithms please refer to [109], [124], and [125]. For each algorithm, all of the functions are run in parallel using a 32 core Intel® CPU 32 GB RAM workstation. For the first set of experiments, results are presented in Table 6.2.

Although the statistical results presented in Table 6.3 provide a first insight into the performance of the algorithms, a pair-wise statistical test is typically used for a better comparison. For this purpose, by using the results obtained from 30 runs of each algorithm, a Wilcoxon Signed-Rank Test is performed with a statistical significance value $\alpha = 0.05$. The null hypothesis H_0 for this test is: "There is no difference between the median of the solutions produced by algorithm A and the median of the solutions produced by algorithm B for the same benchmark problem", i.e., median (A) = median (B). To determine whether algorithm A reached a statistically better solution than algorithm B, or if not, whether the alternative hypothesis is valid, the sizes of the ranks provided by the Wilcoxon Signed-Rank Test (i.e., T+ and T-, as defined in [126]) are examined.

Most of the modern software development tools use an arithmetic precision of 10^{-16} in the double-precision mode. An arithmetic precision value that is higher than necessary makes it difficult to compare the local search abilities of the algorithms [126]. For this purpose, during the statistical pair-wise comparison, resulting values below 10^{-16} are considered as 0.

Table 6.3 : Statistical results of 30 runs obtained by SA, PS, PSO2011, ABC and VS algorithms (values < 10^{-16} are considered as 0)

No	Min.		SA	PS	PSO2011	ABC	VS
F1	0	Mean	1.866666667	0	0	0	0
		StdDev	1.136641554	0	0	0	0
		Best	0	0	0	0	0
F2	0	Mean	0	0	0.066666667	0	0.2
		StdDev	0	0	0.253708132	0	0.406838102
		Best	0	0	0	0	0
F3	0	Mean	0	0	0	2.78624E-16	0
		StdDev	0	0	0	0	0
		Best	0	0	0	2.23487E-16	0
F4	0	Mean	0	0	0	2.75098E-16	0
		StdDev	0	0	0	0	0
		Best	0	0	0	1.85594E-16	0
F5	0	Mean	0.4028326	0.049370406	1.64098E-05	0.013732963	0.000145026
		StdDev	0.301544881	0.046578461	5.56581E-06	0.002379448	7.30549E-05
		Best	0.001414536	1.61333E-05	7.13993E-06	0.008413424	5.54996E-05
F6	0	Mean	0.000430475	0	0	6.37598E-16	0
		StdDev	0.000943865	0	0	3.58687E-16	0
		Best	2.51078E-08	0	0	0	0
F7	-1	Mean	-0.028827505	-8.11022E-05	-1	-1	-1
		StdDev	0.157894721	0	0	0	0
		Best	-0.864825008	-8.11022E-05	-1	-1	-1
F8	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F9	0	Mean	1.83377047	0.002199995	0	0.00576453	0
		StdDev	2.351638954	0	0	0.003966867	0
		Best	0.000971812	0.002199995	0	0.000383073	0
F10	-50	Mean	-49.84789091	-50	-50	-50	-50
		StdDev	0.150775917	0	3.61345E-14	4.94748E-14	2.96215E-14
		Best	-49.98701123	-50	-50	-50	-50
F11	-210	Mean	-209.5023223	-209.9954224	-210	-210	-210
		StdDev	0.230476381	0	2.30778E-13	9.62204E-12	6.19774E-13
		Best	-209.8801988	-209.9954224	-210	-210	-210
F12	0	Mean	0	0	0	7.56674E-14	0
		StdDev	0	0	0	3.76382E-14	0
		Best	0	0	0	2.31887E-14	0
F13	0	Mean	0	0	2.04664E-07	9.09913E-05	1.43967E-05
		StdDev	0	0	1.21051E-08	1.42475E-05	2.27742E-06
		Best	0	0	1.72679E-07	5.23427E-05	5.71959E-06
F14	0	Mean	0	0	1.094284383	8.51365E-16	0
		StdDev	0	0	0.870781136	0	0
		Best	0	0	0.107097937	6.93597E-16	0

Table 6.3 (continued) : Statistical results of 30 runs obtained by SA, PS, PSO2011, ABC and VS algorithms (values < 10^{-16} are considered as 0).

No	Min.		SA	PS	PSO2011	ABC	VS
F15	0	Mean	0	0	0	0.000760232	0
		StdDev	0	0	0	0.000440926	0
		Best	0	0	0	0.00027179	0
F16	0	Mean	0.224618742	9.84185348	0.930212233	0.003535257	0.367860114
		StdDev	0.097171414	0	1.714978077	0.003314818	1.130879848
		Best	0.082077849	9.84185348	0	7.08757E-05	9.42587E-05
F17	0	Mean	0.990721802	0.666666667	0.666666667	1.91607E-15	0.666666667
		StdDev	0.029412712	0	4.38309E-16	2.55403E-16	7.68909E-16
		Best	0.871516993	0.666666667	0.666666667	1.1447E-15	0.666666667
F18	0.998	Mean	5.5682975	0.998003838	34.26621987	0.998003933	0.998003838
		StdDev	4.367922182	4.51681E-16	126.6004794	4.33771E-07	0
		Best	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838
F19	0.398	Mean	0.398269177	0.397887358	0.397887358	0.397887358	0.397887358
		StdDev	0.001624387	0	0	0	0
		Best	0.397887361	0.397887358	0.397887358	0.397887358	0.397887358
F20	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F21	0	Mean	5.28496E-05	0	0	0	0
		StdDev	7.35674E-05	0	0	0	0
		Best	4.08428E-08	0	0	0	0
F22	0	Mean	0	0	26.11016129	0	57.60799224
		StdDev	0	0	5.686650032	0	13.94980276
		Best	0	0	16.91429893	0	33.82857771
F23	-12569.5	Mean	-1891.275468	-3686.285205	-8316.185447	-12569.48662	-11283.05416
		StdDev	137.3913021	2.77513E-12	463.9606712	1.85009E-12	352.1869262
		Best	-2188.304761	-3686.285205	-9466.201047	-12569.48662	-11799.62928
F24	-1.8013	Mean	-1.792778285	-1.80130341	-1.80130341	-1.80130341	-1.80130341
		StdDev	0.043874926	1.35504E-15	9.03362E-16	9.03362E-16	9.03362E-16
		Best	-1.801296643	-1.80130341	-1.80130341	-1.80130341	-1.80130341
F25	-4.6877	Mean	-3.670604734	-4.495893207	-4.67700874	-4.687658179	-4.670953055
		StdDev	0.496257736	2.71009E-15	0.036487971	2.60778E-15	0.020809276
		Best	-4.684023442	-4.495893207	-4.687658179	-4.687658179	-4.687658179
F26	-9.6602	Mean	-6.060491565	-8.461507306	-9.204154798	-9.660151716	-8.793361668
		StdDev	0.504024688	5.42017E-15	0.298287637	0	0.382153549
		Best	-6.880235805	-8.461507306	-9.660151716	-9.660151716	-9.410563187
F27	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F28	-1.03163	Mean	-1.031621639	-1.031628453	-1.031628453	-1.031628453	-1.031628453
		StdDev	2.1595E-05	4.51681E-16	6.71219E-16	6.77522E-16	6.77522E-16
		Best	-1.031628448	-1.031628453	-1.031628453	-1.031628453	-1.031628453
F29	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F30	0	Mean	0	0	0	0	0
		StdDev	0	0	0	0	0
		Best	0	0	0	0	0
F31	-186.73	Mean	-186.7309087	-123.5767709	-186.7309088	-186.7309088	-186.7309088
		StdDev	5.76173E-07	0	4.49449E-13	1.18015E-14	3.76909E-14
		Best	-186.7309088	-123.5767709	-186.7309088	-186.7309088	-186.7309088
F32	3	Mean	3.000000254	30	3	3	3
		StdDev	4.36073E-07	1.08403E-14	1.22871E-15	1.7916E-15	1.44961E-15
		Best	3	30	3	3	3

Table 6.3 (continued) : Statistical results of 30 runs obtained by SA, PS, PSO2011, ABC and VS algorithms (values < 10^{-16} are considered as 0).

No	Min.		SA	PS	PSO2011	ABC	VS
F33	0.00031	Mean	0.002635099	0.00031966	0.000307486	0.000319345	0.000307486
		StdDev	0.001644496	0	0	5.4385E-06	0
		Best	0.000780214	0.00031966	0.000307486	0.00030894	0.000307486
F34	-10.15	Mean	-9.002836723	-5.055197729	-9.363375596	-10.15319968	-10.15319968
		StdDev	2.071338221	9.03362E-16	2.081063878	7.2269E-15	7.2269E-15
		Best	-10.15247689	-5.055197729	-10.15319968	-10.15319968	-10.15319968
F35	-10.4	Mean	-8.979515615	-5.087671825	-10.40294057	-10.40294057	-10.40294057
		StdDev	2.744123131	3.61345E-15	1.80672E-15	1.04311E-15	1.61598E-15
		Best	-10.40272816	-5.087671825	-10.40294057	-10.40294057	-10.40294057
F36	-10.53	Mean	-8.498294797	-5.128480787	-10.53640982	-10.53640982	-10.53640982
		StdDev	2.645882377	3.61345E-15	0	2.13774E-15	1.47518E-15
		Best	-10.5362255	-5.128480787	-10.53640982	-10.53640982	-10.53640982
F37	0	Mean	0.844338683	0.295334941	0.002854996	0.003526435	0.002815467
		StdDev	1.292248967	1.1292E-16	0.007218334	0.001604834	0.002374325
		Best	0.001422252	0.295334941	1.30581E-08	0.00097117	0
F38	0	Mean	0.021367747	0	3.14986E-05	0.000288005	1.78046E-06
		StdDev	0.032095897	0	6.43525E-05	0.00013892	1.28089E-06
		Best	3.02411E-05	0	1.50435E-11	5.82234E-05	4.82E-09
F39	-3.86	Mean	-3.861918832	-3.862782148	-3.862782148	-3.862782148	-3.862782148
		StdDev	0.001277024	2.25841E-15	2.71009E-15	2.71009E-15	2.69625E-15
		Best	-3.86274843	-3.862782148	-3.862782148	-3.862782148	-3.862782148
F40	-3.32	Mean	-3.281839942	-3.203161918	-3.318394475	-3.322368011	-3.322368011
		StdDev	0.03736791	1.80672E-15	0.021763955	6.54548E-16	5.14996E-16
		Best	-3.318182614	-3.203161918	-3.322368011	-3.322368011	-3.322368011
F41	0	Mean	0	0	0.004761038	0	0.032798017
		StdDev	0	0	0.008047673	0	0.018570459
		Best	0	0	0	0	0.00739604
F42	0	Mean	8.88178E-16	8.88178E-16	0.660186991	2.44545E-14	1.15463E-14
		StdDev	0	0	0.711496752	3.02083E-15	3.61345E-15
		Best	8.88178E-16	8.88178E-16	7.99361E-15	2.22045E-14	7.99361E-15
F43	0	Mean	1.668971097	0	0.024187276	2.63417E-16	0.114662313
		StdDev	1.1292E-15	0	0.080213839	0	0.532276418
		Best	1.668971097	0	0	1.29727E-16	0
F44	0	Mean	3	0	0	2.7797E-16	0
		StdDev	0	0	0	0	0
		Best	3	0	0	2.22214E-16	0
F45	-1.08	Mean	-1.072219306	-1.080938442	-1.080938442	-1.080938442	-1.080938442
		StdDev	0.017963587	9.03362E-16	4.51681E-16	4.96507E-16	4.51681E-16
		Best	-1.080936396	-1.080938442	-1.080938442	-1.080938442	-1.080938442
F46	-1.5	Mean	-0.491126582	-0.303738989	-1.499999223	-1.499999223	-1.499999223
		StdDev	0.154632766	0	6.77522E-16	1.05365E-15	6.77522E-16
		Best	-0.797704495	-0.303738989	-1.499999223	-1.499999223	-1.499999223
F47	NA	Mean	-0.017734088	-0.422042106	-1.069011938	-1.482016588	-1.271399999
		StdDev	0.018288461	1.6938E-16	0.422205043	0.097662612	0.313658787
		Best	-0.075917322	-0.422042106	-1.5	-1.499998488	-1.5
F48	0	Mean	1.12203E-07	0	0	0	0
		StdDev	4.54332E-07	0	0	0	0
		Best	4.99041E-12	0	0	0	0
F49	0	Mean	765.3235916	0	3.083487114	1.48707E-12	0
		StdDev	1025.49687	0	4.389694328	8.11041E-12	0
		Best	3.65426E-05	0	0	3.1715E-16	0
F50	0	Mean	7148.686851	18.79802113	580.0839029	1.111095363	0
		StdDev	14013.62432	3.61345E-15	1280.698395	0.598962098	0
		Best	0.016208788	18.79802113	0	0.182153237	0

In Table 6.4, the statistical pair-wise results of the VS algorithm compared to those of other algorithms are given. In this table, '+' indicates cases in which the null hypothesis is rejected and the VS algorithm exhibited a statistically superior performance in the pair-wise Wilcoxon Signed-Rank Test at the 95% significance level ($\alpha = 0.05$); '-' indicates cases in which the null hypothesis is rejected and the VS algorithm displayed an inferior performance; and '=' indicates cases in which there is no statistical different between two algorithms. The last row of the Table 6.4 shows the total count of (+/=/-) the three statistical significance cases in the pair-wise comparison. From this table, it can be shown that the VS algorithm outperforms the SA, PS and ABC algorithms and compete with the PSO2011 algorithm. The SA algorithm performs a pure random search over the search space for which obtained results become meaningful. The PS algorithm is also a single-solution based algorithm that performs poorly compared to the VS algorithm. The ABC algorithm is a powerful swarm-based algorithm that is used successfully for the solution of many types of optimization problems. From Table 6.3, it can be shown that the difference between the VS and ABC algorithms is mainly due to the local search ability of the VS algorithm. For a number of functions, the ABC algorithm fails to exceed the 10^{-16} limit that is used during the pair-wise statistical comparison. Once the algorithm converges to a near optimal point, the inverse incomplete gamma function provides excellent local search ability to the VS algorithm, which helps the algorithm to further improve the solution. The PSO2011 algorithm exhibits similar performance to the proposed VS algorithm. Thus, for 30 functions, the null hypothesis is accepted in the pair-wise comparison of the two algorithms. Due to the results obtained from the pair-wise statistical test, PSO2011 performs better, but the VS algorithm is highly competitive.

In Table 6.5, a problem-based (US, UN, MS and MN) comparison of the algorithms is also provided. Each cell in the Table 6.5 shows the total count of the three statistical significance cases (+/=/-) in the pair-wise comparison obtained from Table 6.4. From Table 6.5, it can be shown that, for MN (multimodal non-separable) functions, the proposed VS algorithm performs better than the others. For UN (unimodal non-separable) functions the proposed VS algorithm also performs better than the SA, PS, and ABC algorithms.

Table 6.4 : Pair-wise statistical comparison of the algorithms by Wilcoxon Signed-Rank Test ($\alpha = 0.05$).

Function	VS vs. SA				VS vs. PS				VS vs. PSO2011				VS vs. ABC			
	p-value	T+	T-	winner	p-value	T+	T-	winner	p-value	T+	T-	winner	p-value	T+	T-	winner
F1	2.871E-06	0	406	+	1	0	0	=	1	0	0	=	1	0	0	=
F2	0.0143059	21	0	-	0.014306	21	0	-	0.0455	10	0	-	0.014306	21	0	-
F3	1	0	0	=	1	0	0	=	1	0	0	=	1.73E-06	0	465	+
F4	1	0	0	=	1	0	0	=	1	0	0	=	1.73E-06	0	465	+
F5	1.734E-06	0	465	+	1.92E-06	1	464	+	1.73E-06	465	0	-	1.73E-06	0	465	+
F6	1.734E-06	0	465	+	1	0	0	=	1	0	0	=	3.79E-06	0	406	+
F7	1.014E-07	0	465	+	4.32E-08	0	465	+	1	0	0	=	1	0	0	=
F8	1	0	0	=	1	0	0	=	1	0	0	=	0.067889	0	10	=
F9	1.734E-06	0	465	+	4.32E-08	0	465	+	1	0	0	=	1.73E-06	0	465	+
F10	1.734E-06	0	465	+	1.96E-07	0	465	+	0.0455	10	0	-	1.21E-06	0	465	+
F11	1.734E-06	0	465	+	9.57E-07	0	465	+	0.001986	99	6	-	1.69E-06	0	465	+
F12	1	0	0	=	1	0	0	=	1	0	0	=	1.73E-06	0	465	+
F13	1.734E-06	465	0	-	1.73E-06	465	0	-	1.73E-06	465	0	-	1.73E-06	0	465	+
F14	1	0	0	=	1	0	0	=	1.73E-06	0	465	+	1.73E-06	0	465	+
F15	1	0	0	=	1	0	0	=	1	0	0	=	1.73E-06	0	465	+
F16	0.0027653	87	378	+	1.73E-06	0	465	+	0.130592	306	159	=	0.002957	88	377	+
F17	1.717E-06	0	465	+	1.71E-06	465	0	-	0.00085	63.5	371.5	+	1.72E-06	465	0	-
F18	1.724E-06	0	465	+	4.32E-08	0	465	+	0.10247	0	6	=	1.73E-06	0	465	+
F19	1.734E-06	0	465	+	1	0	0	=	1	0	0	=	1	0	0	=
F20	1	0	0	=	1	0	0	=	1	0	0	=	1	0	0	=
F21	1.734E-06	0	465	+	1	0	0	=	1	0	0	=	1	0	0	=
F22	1.733E-06	465	0	-	1.73E-06	465	0	-	1.92E-06	464	1	-	1.73E-06	465	0	-
F23	1.734E-06	0	465	+	1.73E-06	0	465	+	1.73E-06	0	465	+	1.73E-06	465	0	-
F24	1.734E-06	0	465	+	4.32E-08	0	465	+	1	0	0	=	1	0	0	=
F25	1.734E-06	0	465	+	1.42E-06	0	465	+	0.018985	182.5	48.5	-	2.12E-06	435	0	-
F26	1.734E-06	0	465	+	0.000332	58	407	+	0.000306	408	57	-	1.73E-06	465	0	-
F27	1	0	0	=	1	0	0	=	1	0	0	=	1	0	0	=
F28	1.734E-06	0	465	+	4.32E-08	0	465	+	0.317311	0	1	=	1	0	0	=
F29	1	0	0	=	1	0	0	=	1	0	0	=	1	0	0	=
F30	1	0	0	=	1	0	0	=	1	0	0	=	0.001766	0	66	+
F31	1.734E-06	0	465	+	1.2E-06	0	465	+	0.000492	22.5	230.5	+	0.000141	342	36	-

Table 6.4 (continued) : Pair-wise statistical comparison of the algorithms by Wilcoxon Signed-Rank Test ($\alpha = 0.05$).

Function	VS vs. SA				VS vs. PS				VS vs. PSO2011				VS vs. ABC			
	p-value	T+	T-	winner	p-value	T+	T-	winner	p-value	T+	T-	winner	p-value	T+	T-	winner
F32	1.734E-06	0	465	+	7.45E-07	0	465	+	0.067045	196.5	79.5	=	0.002375	33.5	219.5	+
F3 3	1.734E-06	0	465	+	1.59E-06	0	465	+	0.00016	281.5	18.5	-	1.73E-06	0	465	+
F34	1.734E-06	0	465	+	4.32E-08	0	465	+	0.0656	0	10	=	1	0	0	=
F35	1.734E-06	0	465	+	3.26E-07	0	465	+	0.014306	21	0	-	0.000967	19	152	+
F36	1.734E-06	0	465	+	6.25E-07	0	465	+	7.74E-06	210	0	-	0.001054	0	78	+
F37	4.286E-06	9	456	+	1.73E-06	0	465	+	0.082206	317	148	=	0.184622	168	297	=
F38	1.734E-06	0	465	+	1.73E-06	465	0	-	0.001593	79	386	+	1.73E-06	0	465	+
F39	1.734E-06	0	465	+	7.24E-08	0	435	+	0.317311	1	0	=	0.317311	1	0	=
F40	1.734E-06	0	465	+	1.44E-07	0	465	+	0.705457	6	4	=	0.032509	22.5	82.5	+
F41	1.733E-06	465	0	-	1.73E-06	465	0	-	5.22E-06	454	11	-	1.73E-06	465	0	-
F42	8.207E-07	465	0	-	8.21E-07	465	0	-	0.00499	66	285	+	1.38E-06	0	465	+
F43	2.933E-07	1	464	+	0.042168	15	0	-	0.632281	26.5	18.5	=	0.057096	140	325	=
F44	4.32E-08	0	465	+	1	0	0	=	1	0	0	=	1.73E-06	0	465	+
F45	1.734E-06	0	465	+	4.32E-08	0	465	+	1	0	0	=	0.025347	0	15	+
F46	1.734E-06	0	465	+	4.32E-08	0	465	+	1	0	0	=	8.83E-07	0	435	+
F47	1.734E-06	0	465	+	8.01E-07	0	465	+	0.061202	76.5	199.5	=	0.416534	272	193	=
F48	1.734E-06	0	465	+	1	0	0	=	1	0	0	=	1	0	0	=
F49	1.734E-06	0	465	+	1	0	0	=	0.003346	0	66	+	1.73E-06	0	465	+
F50	1.734E-06	0	465	+	4.32E-08	0	465	+	8.84E-05	0	210	+	1.73E-06	0	465	+
+/=/-	35/10/5				25/17/8				8/30/12				26/16/8			

Thus, from these results it can be inferred that the VS algorithm performs better for non-separable functions. For MS (multimodal separable) functions the proposed VS algorithm outperforms the single-solution based algorithms, but population-based algorithms are superior to the VS algorithm for this case. There is no strict difference between the results obtained for the US (unimodal separable) functions. In fact, the provided number of functions for US problems is not sufficient to make a statistically convincing inference.

Table 6.5 : Problem-based comparison of the proposed VS algorithm.

Problem Type	**VS vs. SA**	**VS vs. PS**	**VS vs. PSO2011**	**VS vs. ABC**
US	2/2/1	1/3/1	0/3/2	3/1/1
UN	7/4/1	5/5/2	2/7/3	9/2/1
MS	7/1/1	5/3/1	1/5/3	1/4/4
MN	19/3/2	14/6/4	5/15/4	13/9/2
Total (+/=/-)	35/10/5	25/17/8	8/30/12	26/16/8

In Figure 6.7, the average computational time of 30 runs for 500,000 iterations is provided for the VS, PSO2011 and ABC algorithms. Because the proposed VS algorithm is quite simple, it is computationally inexpensive compared to the population-based methods. A comparison of the SA and PS algorithms could not be provided due to the limitations of MATLAB® Global Optimization toolbox; however, because these algorithms are also single-solution based metaheuristics, the computational time for these algorithms is expected to be similar to that of the VS algorithm.

6.2.1.4 Convergence behaviours of the algorithms

In the second set of experiments, convergence behaviors of the algorithms are studied. For this purpose, an iteration based comparison of the algorithms (for 100, 1000 and 10,000 iterations) is performed. It is shown that, for most of the functions, algorithms generally tend to converge to their optimum for 10,000 iterations. Therefore, to capture the convergence behavior of the algorithms, iteration numbers smaller than 10,000 are used. In Table 6.6, the average results of 30 runs for 100, 1000, and 10,000 iterations are given for the convergence analysis of the algorithms.

From Table 6.6, it can be shown that the proposed VS algorithm is quite competitive and performs better than the SA, PS and ABC algorithms, while again being competitive with the PSO2011 algorithm.

Table 6.6 : Average results of 30 runs to study the convergence behavior of the algorithms. Exp-1 = 100, Exp-2 = 1000 and Exp-3 = 10,000 iterations (values < 10^{-16} are considered as 0).

No	Min.		SA	PS	PSO2011	ABC	VS
F1	0	Exp-1	8.266666667	1	0	0	1.7
		Exp-2	3.366666667	0	0	0	0
		Exp-3	4	0	0	0	0
F2	0	Exp-1	0	0	21.76666667	1407.7	246.2333333
		Exp-2	0	0	0.133333333	0	5.266666667
		Exp-3	0	0	0.1	0	2
F3	0	Exp-1	0	0	17.83386799	1295.311832	207.816882
		Exp-2	0	0	0	1.64239E-11	2.51339E-08
		Exp-3	0	0	0	4.64604E-16	0
F4	0	Exp-1	0	0	11.84694367	152.3545786	69.58526034
		Exp-2	0	0	0.000333638	2.08038E-12	0.009149095
		Exp-3	0	0	0	4.37095E-16	1.41264E-15
F5	0	Exp-1	0.47953561	0.283638723	0.066563724	1.472308609	0.436218853
		Exp-2	0.427474235	0.064990433	0.006226974	0.111736883	0.032896182
		Exp-3	0.364413261	0.067813379	0.000748077	0.02993958	0.005195975
F6	0	Exp-1	0.754518564	0.061084986	6.2751E-05	0.009077853	2.07862E-09
		Exp-2	0.022823365	3.31263E-07	1.24268E-10	3.40978E-06	0
		Exp-3	0.037855479	0	2.02464E-15	3.86825E-12	0
F7	-1	Exp-1	-4.33472E-09	-8.11021E-05	-1	-0.666175773	-1
		Exp-2	-2.51646E-07	-8.11022E-05	-1	-0.99872753	-1
		Exp-3	-1.98058E-06	-8.11022E-05	-1	-0.999999336	-1
F8	0	Exp-1	0	0	6.20157E-14	0.001179646	1.0176E-14
		Exp-2	0	0	0	1.56512E-08	0
		Exp-3	0	0	0	7.43831E-15	0
F9	0	Exp-1	19.12830629	20.259375	0.361837754	1.407757675	1.788528801
		Exp-2	3.661805918	0.00330069	5.51134E-06	0.301430841	0.00102707
		Exp-3	2.488655875	0.002199995	0	0.075522598	3.53655E-16
F10	-50	Exp-1	-25.75587151	-5	-49.99994001	-49.56419749	-49.99920138
		Exp-2	-48.8472231	-49.75	-50	-49.99998522	-50
		Exp-3	-48.85964687	-50	-50	-50	-50
F11	-210	Exp-1	-23.47203906	10	-204.178138	-154.3120633	-187.758909
		Exp-2	-126.8987485	-94	-210	-209.135915	-209.9999894
		Exp-3	-126.2521879	-209.9589844	-210	-209.9999999	-210
F12	0	Exp-1	0	0	0.243903567	42.85390927	0.011121307
		Exp-2	0	0	0	1.828101769	7.82812E-15
		Exp-3	0	0	0	0.000221146	0
F13	0	Exp-1	0	0	2.285803279	27.91221909	20.67093817
		Exp-2	0	0	0.047460485	0.122474319	0.284153185
		Exp-3	0	0	0.000456088	0.003219558	0.00716883
F14	0	Exp-1	0	0	3.759761969	2.92866529	36.51488013
		Exp-2	0	0	1.401158988	7.53555E-07	0.034092402
		Exp-3	0	0	1.210521235	1.15906E-15	8.26573E-08
F15	0	Exp-1	0	0	2559.631822	35203.6207	9251.760935
		Exp-2	0	0	1.10605399	11612.21356	15.93412113
		Exp-3	0	0	0	1294.286411	1.14751E-11
F16	0	Exp-1	29	29	1034.076117	293570.9219	18932.75325
		Exp-2	28.83571779	27.81680679	46.66422891	1.613886475	252.1545704
		Exp-3	28.82692118	25.14455549	16.31958178	0.392342246	75.95278982
F17	0	Exp-1	1	1	17.3424916	2044.257203	187.724627
		Exp-2	0.999958145	0.666992188	0.680708375	0.02370942	1.090003416
		Exp-3	1	0.666666667	0.666666667	5.81096E-15	0.669410059
F18	0.998	Exp-1	11.08157418	0.998003839	51.31249642	442.0799609	41.02927716
		Exp-2	6.467075194	0.998003838	84.2031109	39.81361114	1.19667749
		Exp-3	6.648011115	0.998003838	34.26881766	1.31818964	0.998003838

Table 6.6 (continued) : Average results of 30 runs to study the convergence behavior of the algorithms. Exp-1 = 100, Exp-2 = 1000 and Exp-3 = 10,000 iterations (values < 10^{-16} are considered as 0).

No	Min.		SA	PS	PSO2011	ABC	VS
F19	0.398	Exp-1	0.745748422	0.399103668	0.39788741	0.397887358	0.397887358
		Exp-2	0.398421335	0.397887358	0.397887358	0.397887358	0.397887358
		Exp-3	0.398272764	0.397887358	0.397887358	0.397887358	0.397887358
F20	0	Exp-1	0	0	5.43941E-12	0	2.42354E-12
		Exp-2	0	0	0	0	0
		Exp-3	0	0	0	0	0
F21	0	Exp-1	0.314734743	0	5.01241E-14	1.17502E-05	3.65947E-12
		Exp-2	0.000156561	0	0	0	0
		Exp-3	0.000175671	0	0	0	0
F22	0	Exp-1	0	0	166.8509083	91.85093274	134.1911749
		Exp-2	0	0	39.20933653	0.157157274	90.01036413
		Exp-3	0	0	28.28451301	0	71.43784888
F23	-12569.5	Exp-1	-516.8821802	-221.1097472	-4688.812483	-8574.098182	-7942.151101
		Exp-2	-837.1458474	-1695.174729	-6541.101632	-12184.34762	-9866.229322
		Exp-3	-836.2994779	-3686.207854	-8170.468922	-12569.48662	-10459.83993
F24	-1.8013	Exp-1	-1.159905575	-1.801293149	-1.80130341	-1.80130341	-1.80130341
		Exp-2	-1.79290997	-1.80130341	-1.80130341	-1.80130341	-1.80130341
		Exp-3	-1.794159626	-1.80130341	-1.80130341	-1.80130341	-1.80130341
F25	-4.6877	Exp-1	-2.153552487	-2.402497914	-4.568771105	-4.686708108	-4.145175998
		Exp-2	-3.199309786	-4.495893207	-4.634488914	-4.687658179	-4.449685947
		Exp-3	-3.20317597	-4.495893207	-4.64120811	-4.687658179	-4.567268598
F26	-9.6602	Exp-1	-3.145901017	-1.296343016	-6.721497307	-9.145677633	-6.991477404
		Exp-2	-4.85399641	-8.208602594	-8.603123358	-9.658759622	-7.913983601
		Exp-3	-4.962493257	-8.461507306	-8.965074725	-9.660151716	-8.686465036
F27	0	Exp-1	0	0	0.001632485	0.005802235	0.008744319
		Exp-2	0	0	0	3.48268E-05	0.001943182
		Exp-3	0	0	0	5.18835E-09	0
F28	-1.03163	Exp-1	-0.779030894	-1.031551548	-1.031628453	-1.031628453	-1.031628453
		Exp-2	-1.031473907	-1.031628453	-1.031628453	-1.031628453	-1.031628453
		Exp-3	-1.031453693	-1.031628453	-1.031628453	-1.031628453	-1.031628453
F29	0	Exp-1	0	0	2.48318E-12	1.28927E-09	1.95223E-12
		Exp-2	0	0	0	0	0
		Exp-3	0	0	0	0	0
F30	0	Exp-1	0	0	5.13608E-11	0.000235255	1.073E-09
		Exp-2	0	0	0	3.34607E-09	0
		Exp-3	0	0	0	1.05101E-15	0
F31	-186.73	Exp-1	-103.533756	-123.445392	-186.7273973	-186.7308492	-186.7309088
		Exp-2	-186.7309085	-123.5767709	-186.7309086	-186.7309088	-186.7309088
		Exp-3	-176.9865641	-123.5767709	-186.7309088	-186.7309088	-186.7309088
F32	3	Exp-1	6.822499574	30.0170927	3	3.003161304	3
		Exp-2	3.000001293	30	3	3.000304997	3
		Exp-3	3.000000669	30	3	3.000000076	3
F33	0.00031	Exp-1	0.033032546	0.00216132	0.001090008	0.001414731	0.000931416
		Exp-2	0.010334262	0.000437024	0.000670146	0.00058389	0.000771504
		Exp-3	0.009121193	0.00031966	0.000374177	0.000406602	0.000307486
F34	-10.15	Exp-1	-0.732013706	-5.055195641	-9.896441311	-10.14567432	-7.975216386
		Exp-2	-5.225705315	-5.055197729	-10.10667432	-10.15319968	-10.15319968
		Exp-3	-5.326989674	-5.055197729	-10.15319968	-10.15319968	-10.15319968
F35	-10.4	Exp-1	-1.220313721	-5.087666505	-10.4029402	-10.39761973	-8.917279506
		Exp-2	-4.755328631	-5.087671825	-10.40294057	-10.40293573	-10.05133272
		Exp-3	-4.886948793	-5.087671825	-10.40294057	-10.40294057	-10.40294057

Table 6.6 (continued) : Average results of 30 runs to study the convergence behavior of the algorithms. Exp-1 = 100, Exp-2 = 1000 and Exp-3 = 10,000 iterations (values < 10^{-16} are considered as 0).

No	Min.		SA	PS	PSO2011	ABC	VS
F37	0	Exp-1	98.83521851	30.09775949	0.570092614	0.705770381	1.326897344
		Exp-2	15.86484077	3.923713434	0.04660114	0.169743099	0.026398136
		Exp-3	8.293007379	0.295334941	0.002107886	0.043522003	0.002401675
F38	0	Exp-1	18.06593955	0	0.0341867	0.070928502	0.027452781
		Exp-2	1.161031853	0	0.000974964	0.022145972	0.000242164
		Exp-3	0.196493545	0	0.000144605	0.003404818	0.00012477
F39	-3.86	Exp-1	-3.525357459	-3.812405944	-3.862782105	-3.862782147	-3.862781719
		Exp-2	-3.859529942	-3.862782148	-3.862782148	-3.862782148	-3.862782148
		Exp-3	-3.860088635	-3.862782148	-3.862782148	-3.862782148	-3.862782148
F40	-3.32	Exp-1	-2.025767543	-2.275890604	-3.306796203	-3.322329207	-3.253836322
		Exp-2	-3.21392372	-3.203161892	-3.312304129	-3.322368011	-3.254815564
		Exp-3	-3.196408153	-3.203161918	-3.318394475	-3.322368011	-3.262764965
F41	0	Exp-1	0	0	1.174905607	13.76813973	2.586458043
		Exp-2	0	0	0.0064782	0.002035399	0.008888174
		Exp-3	0	0	0.00476161	0	0.020667864
F42	0	Exp-1	8.88178E-16	8.88178E-16	2.822882317	13.57612974	5.741707238
		Exp-2	8.88178E-16	8.88178E-16	0.594499875	1.13074E-05	0.661994795
		Exp-3	8.88178E-16	8.88178E-16	0.471201803	3.25073E-14	2.63493E-14
F43	0	Exp-1	1.668971097	1.668971097	3.055864107	5.641956224	67.26147149
		Exp-2	1.668971097	0.831213056	0.031100039	3.59957E-12	29.80917701
		Exp-3	1.668971097	0	0.041467608	4.17011E-16	7.698703865
F44	0	Exp-1	3	3	2.777568301	10.60775637	93.32996747
		Exp-2	3	1.9	0.003383422	1.18784E-11	3.77628E-09
		Exp-3	3	0	0	4.39646E-16	0
F45	-1.08	Exp-1	-0.773794902	-1.079704294	-1.080938442	-1.079661136	-1.080938442
		Exp-2	-1.04129998	-1.080938442	-1.080938442	-1.080937366	-1.080938442
		Exp-3	-1.056610147	-1.080938442	-1.080938442	-1.080938442	-1.080938442
F46	-1.5	Exp-1	-0.096506449	-0.29625687	-1.403550728	-0.946054327	-1.120088196
		Exp-2	-0.277626581	-0.303738989	-1.481074867	-1.460442332	-1.499999223
		Exp-3	-0.302885888	-0.303738989	-1.478771147	-1.499999069	-1.480265916
F47	NA	Exp-1	-4.99714E-05	-3.43953E-06	-0.593200476	-0.337366551	-0.492070745
		Exp-2	-0.007342278	-0.422042106	-0.934146078	-0.669342735	-0.457840889
		Exp-3	-0.00129138	-0.422042106	-1.047012352	-0.850070407	-0.799618675
F48	0	Exp-1	121.3553309	0.000387005	2.79024E-07	6.72588E-09	2.05801E-12
		Exp-2	0.000341217	0	0	0	0
		Exp-3	47.0197343	0	0	0	0
F49	0	Exp-1	5214.41377	2299.12496	33.75848613	9.826333934	229.1060221
		Exp-2	1390.299019	0.046676938	16.5584008	0.741561448	0.170804975
		Exp-3	798.3016669	0	9.822291507	0.043261128	0.002021546
F50	0	Exp-1	69386.43767	87961.08336	3322.708051	1386.230197	800.1367562
		Exp-2	14395.78759	1460.867978	1266.44514	53.81347745	60.58560925
		Exp-3	11286.43357	109.8065763	1057.962044	6.196070433	4.746764164

For a number of functions, the PS algorithm surprisingly converged to the global minimum earlier than the other algorithms; however, for most of the functions, it became trapped in the local minima. The ABC algorithm usually requires an additional number of iterations to converge to the optimum point. The reason for the choice of 500,000 iterations in [107] is now more meaningful. The SA algorithm failed to converge to the optimum point for most of the functions as expected

because, as the number of iterations increased, the probability of finding a good point also increased for the SA algorithm.

As mentioned previously, a time-based comparison was not possible to perform. However, the time-based statistics obtained in the previous experiments (Figure 6.7) implicitly provides information for a time-based comparison after performing an iteration-based convergence analysis. Since the proposed VS algorithm is quite rapid compared to the population-based algorithms, the VS algorithm can perform much more iterations than the population-based ones during a certain period of time.

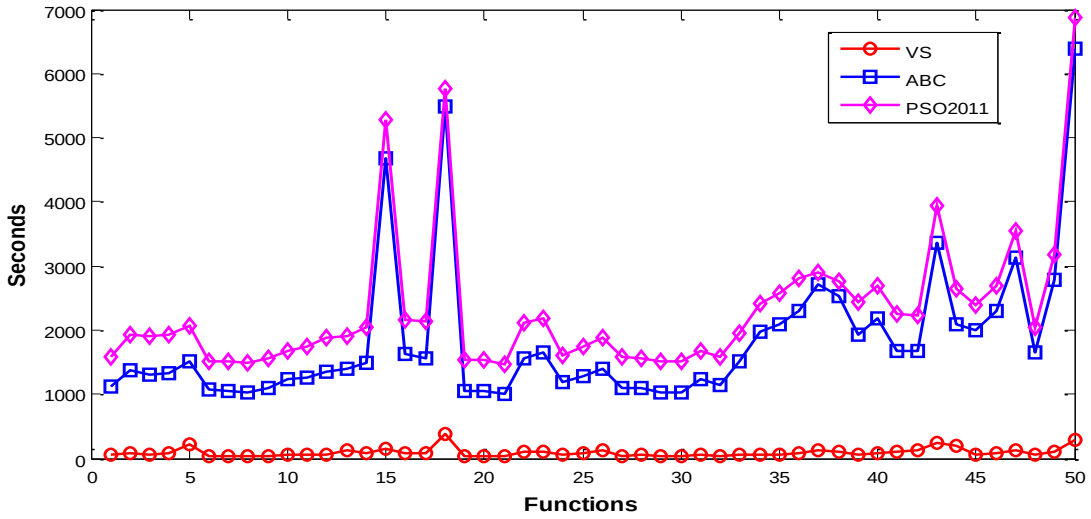

Figure 6.7 : Average computational time of 30 runs for 50 benchmark functions (500,000 iterations).

6.2.1.5 Discussion on the proposed Vortex Search algorithm

This thesis introduces the VS algorithm, which is a single-solution based metaheuristic proposed for the solution of bound-constraint numerical function optimization problems. The VS algorithm utilizes an adaptive step-size adjustment scheme that helps to balance the explorative and exploitative behavior of the search. The algorithm is quite simple and does not require any additional parameters.

The VS algorithm is tested over a large set of 50 benchmark functions that comprises unimodal, multimodal, separable and non-separable problems of different dimensions. The results are compared to the both single-solution based metaheuristics (SA and PS) and population-based metaheuristics (PSO2011 and ABC); the results revealed that besides its simplicity, the proposed VS algorithm is also highly competitive when compared to the performance of the other algorithms. Since the proposed algorithm is quite simple, it is also computationally efficient

when compared to population-based metaheuristics. These advantages of the proposed VS algorithm make it a good candidate for the solution of real-life optimization problems.

In the future studies, the proposed VS algorithm will be improved to handle constraint optimization problems. The VS algorithm will also be applied to some real-life optimization problems including neural network optimization, analog and digital circuit optimization, and optimum data partitioning by using hard and fuzzy clustering algorithms.

6.2.2 Computational results of the proposed Vortex Search algorithm on the protein folding problem

The performance of the proposed VS algorithm is evaulated by using artificial benchmark sequences and the obtained results are compared to the well-known optimization algorithms (PSO2011 and ABC algorithms). The results listed in Table 6.7 are statistical results of the 50 different trials and the maximum number of iterations is set as 5000 for each algorithm.

Table 6.7 : Statistical results of 50 runs obtained by VS, PSO2011, and ABC algorithms for the protein folding problem (5000 iterations).

Amino acid sequence	**n**	**Min.**	**VS**	**PSO2011**	**ABC**
ABAAB	5	-1.3765	Mean: -1.3765 Std: 0 Best: -1.3765	Mean: -1.3765 Std: 0 Best: -1.3765	Mean: -1.3765 Std: 0 Best: -1.3765
ABBBB	5	-0.0660	Mean: -0.0660 Std: 5.71e-007 Best: -0.0660	Mean: -0.0583 Std: 0.0157 Best: -0.0660	Mean: -0.0660 Std: 0 Best: -0.0660
AABABB	6	-1.3620	Mean: -1.3459 Std: 0.0183 Best: -1.3620	Mean: -1.3519 Std: 0.0127 Best: -1.3620	Mean: -1.3620 Std: 6.83e-016 Best: -1.3620
AAABAA	6	-3.6975	Mean: -3.5303 Std: 0.1587 Best: -3.6974	Mean: -3.6928 Std: 0.0210 Best: -3.6975	Mean: -3.6919 Std: 0.0061 Best: -3.6972
ABBABBABABBAB	13	-3.2941	Mean: -1.9143 Std: 0.6089 Best: -3.1891	Mean: -1.8111 Std: 0.2431 Best: -2.2674	Mean: -2.3988 Std: 0.0331 Best: -2.4301
BABABBABABBABBABABBAB	21	-6.1980	Mean: -3.5083 Std: 0.6524 Best: -4.7966	Mean: -2.4579 Std: 0.5274 Best: -3.4112	Mean: -3.8431 Std: 0.3219 Best: -4.2907
ABBABBABABBABBABABBAB ABBABBABABBAB	34	-10.860	Mean: -3.6250 Std: 0.8919 Best: -6.3077	Mean: -2.2261 Std: 0.6634 Best: -3.4940	Mean: -5.4893 Std: 0.4436 Best: -6.1621

In Table 6.7, n represents the length of the amino acid sequence, and Min. represents the known ground state energy of the corresponding amino acid sequence. As it can be shown from Table 6.7, all of the algorithms trap into local minimum points

especially for the long sequences and they can not find the known ground state energies of these sequences. Although, the VS algorithm can find the lowest energies, the results obtained by the ABC algorithm is more robust than the others.

In Figure 6.8a-g, known ground state conformations of the sequences listed in Table 6.7 are shown, respectively. In Figure 6.9a-c, Figure 6.10a-c and Figure 6.11a-c, the best conformations found by the algorithms VS, PSO2011 and ABC are also shown for the last three sequences listed in Table 6.7. Since for the remaining sequences algorithms can find the optimal folds, they are not sketched again. From these figures it is clear that, the algorithms fail to find the correct folds of these amino acid sequences. Since the energy landscape of the off-lattice AB model has a number of deep valleys and hills, the algorithms can easily be trapped into a local minimum point. To avoid this drawback, in literature usually some extensions or combinations of the well-known optimization algorithms are proposed. In this thesis, different from the above mentioned studies, we modified the energy function of the off-lattice AB model to help the algorithms to find the optimum fold of a sequence easily. Thus, without making an algorithmic improvement, a standard optimization algorithm can find the desired conformation in a reasonable amount of time. Experimental results of this modification are provided in the following subsection.

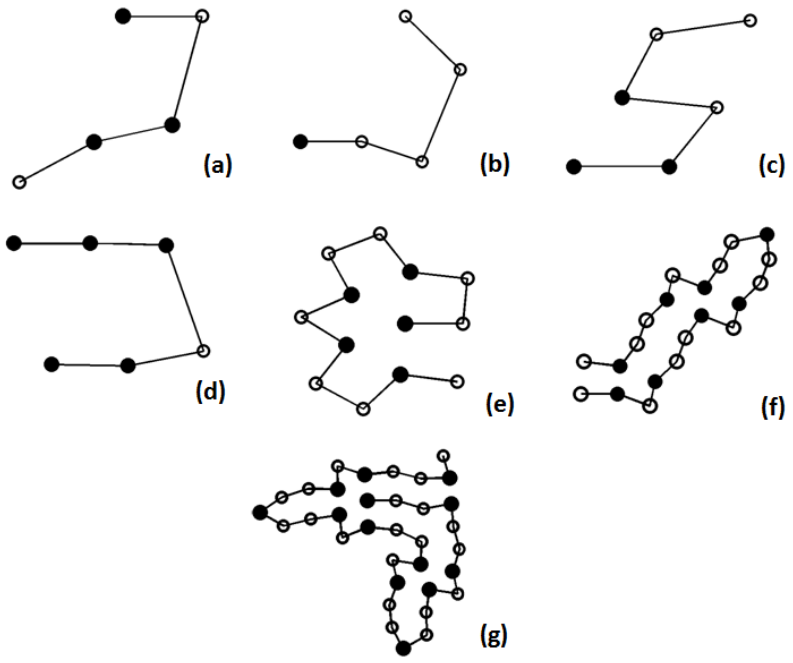

Figure 6.8 : Known ground state conformations of the sequences listed in Table 6.7.

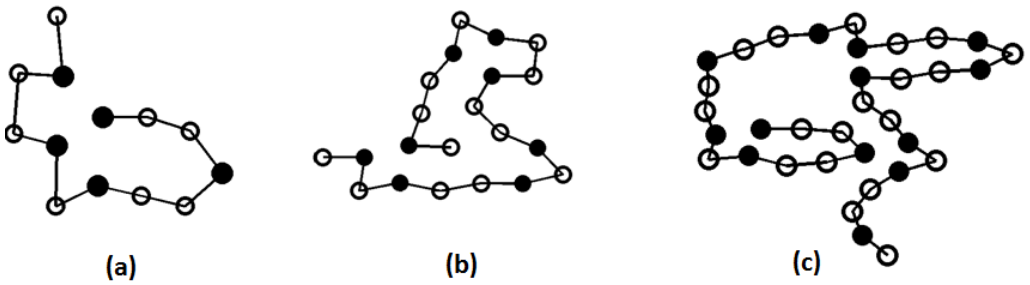

Figure 6.9 : Best conformations found by the VS algorithm for the last three sequences listed in Table 6.7.

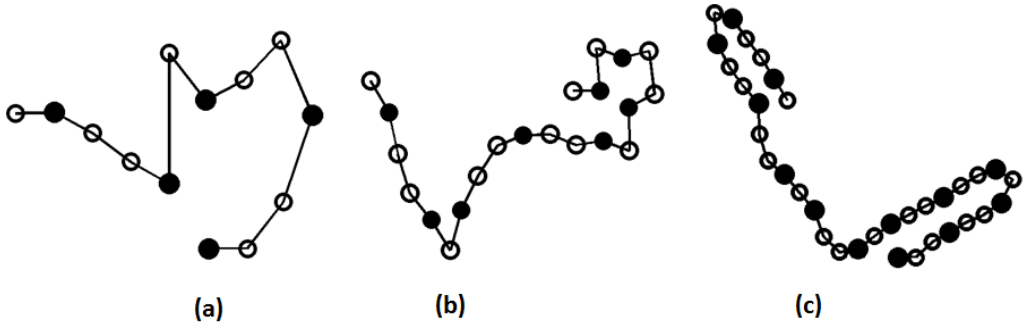

Figure 6.10 : Best conformations found by the PSO2011 algorithm for the last three sequences listed in Table 6.7.

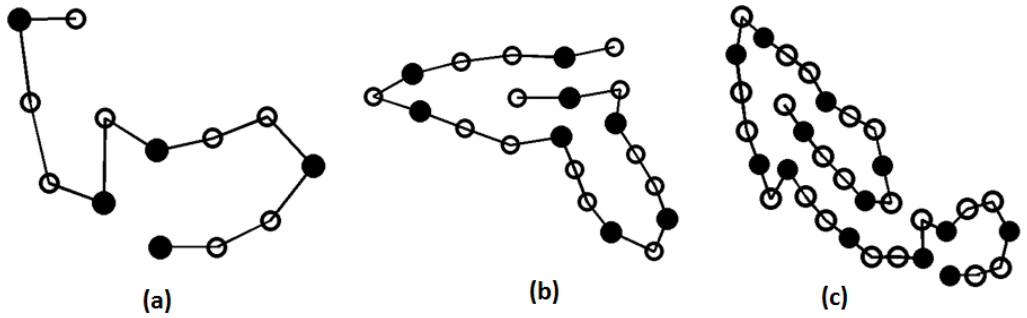

Figure 6.11 : Best conformations found by the ABC algorithm for the last three sequences listed in Table 6.7.

6.2.3 Computational results of the proposed Vortex Search algorithm on the protein folding problem with modified energy function

The modified energy function of the off-lattice AB model introduced in Section 4.3.1 is tested by the VS, PSO2011 and ABC algorithms, again by using the sequences listed in Table 6.7. In Table 6.8, the obtained results are given. Note that, the energy values listed in this table are corresponding original energy function values of the found configurations by using the modified energy function with the corresponding algorithm. The results are the statistical results of 50 different trials and for each trial the maximum number of iterations is set as 5000. From this table, it is clear that, the

modified energy function helps the algorithms to find the near optimal folds of the sequences more easier than the original function. Although, there is a small difference in the reached best energy values, the resulting conformations are quite similar to the original conformations. In Figure 6.12, conformations found by the VS algorithm with the modified energy function are given. As shown from this figure, by using the modified energy function, the VS algorithm can find the near optimal fold of the sequences with length 13 and 21 which was not the case for the original function. For the sequence length 34, the found conformation is not similar the optimal one, however it has hydrophobic cores as expected from the modified energy function.

Compared to the original energy function, the modified energy function does not require any algorithmic improvements to find the near optimal configurations. However, the modified energy function still needs improvements for longer sequences. In the future studies, a more efficient energy function will be searched to find the near optimal or optimal configurations for longer amino-acid chains.

Table 6.8 : Statistical results of 50 runs obtained by the VS, PSO2011, and ABC algorithms with modified energy function (5000 iterations).

Amino acid sequence	**n**	**Min.**	**VS with modified energy function**	**PSO2011 with modified energy function**	**ABC with modified energy function**
ABAAB	5	-1.3765	Mean: -1.3631 Std: 2.41e-4 Best: -1.3637	Mean: -1.3631 Std: 8.90e-006 Best: -1.3631	Mean: -1.3631 Std: 1.63e-009 Best: -1.3631
ABBBB	5	-0.0660	Mean: -0.06596 Std: 0 Best: -0.06596	Mean: -0.583 Std: 0.0157 Best: -0.0660	Mean: -0.0660 Std: 3.77e-007 Best: -0.0660
AABABB	6	-1.3620	Mean: -1.3288 Std: 0.0128 Best: -1.3406	Mean: -1.3293 Std: 0.0127 Best: -1.3392	Mean: -1.3388 Std: 2.52e-009 Best: -1.3388
AAABAA	6	-3.6975	Mean: -3.5433 Std: 0.1332 Best: -3.6757	Mean: -3.6735 Std: 0.0014 Best: -3.6738	Mean: -3.6690 Std: 0.0060 Best: -3.6769
ABBABBABABBAB	13	-3.2941	Mean: -2.4335 Std: 0.5966 Best: -3.2522	Mean: -1.8551 Std: 0.5505 Best: -2.8333	Mean: -2.4850 Std: 0.3145 Best: -3.2351
BABABBABABBABBABABBAB	21	-6.1980	Mean: -3.4190 Std: 0.7965 Best: -5.8249	Mean: -2.9347 Std: 0.7472 Best: -4.4791	Mean: -3.4141 Std: 0.5323 Best: -4.5718
ABBABBABABBABBABABBABABABBABBABABBAB	34	-10.860	Mean: -3.9327 Std: 0.9312 Best: -6.3852	Mean: -2.4136 Std: 0.8435 Best: -4.1066	Mean: -5.2873 Std: 0.5942 Best: -6.2637

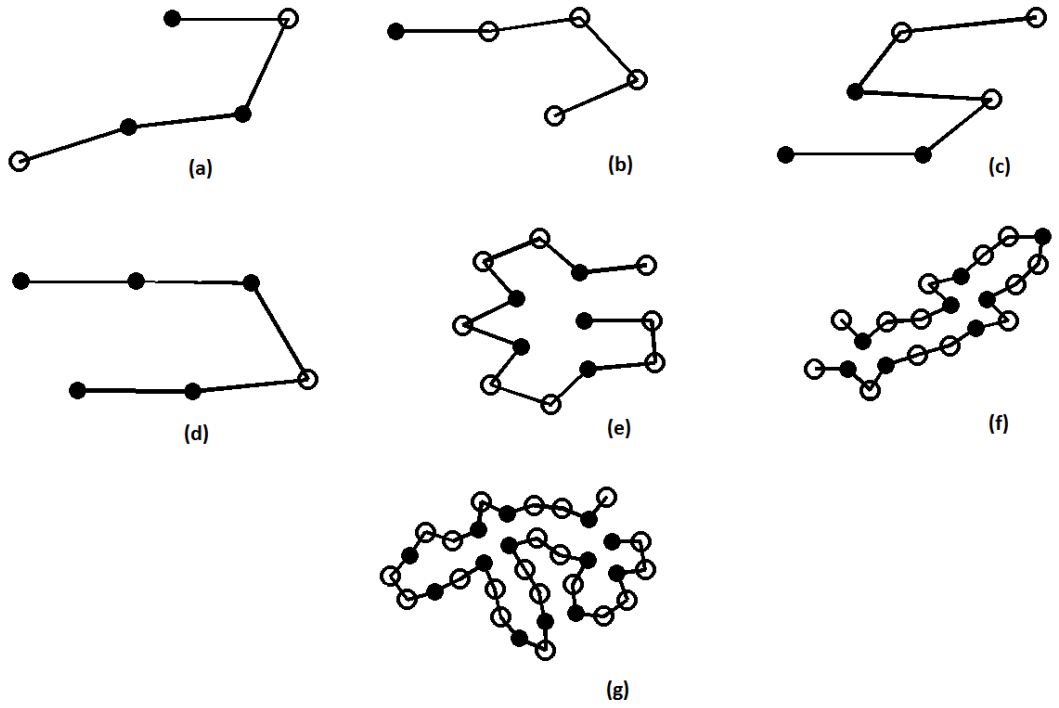

Figure 6.12 : Best conformations found by the VS algorithm with the modified energy function for the sequences listed in Table 6.7.

6.3 Computational Results of the All-Atom Model

In all-atom simulations, usually small peptides are used to evaluate the performance of the algorithms. From a computational perspective, this provides the scientist to evaluate their newly developed algorithms within a reasonable amount of computational time.

Met-enkephalin (PDB:1PLW), is the most widely used peptide in all atom model simulations. It has a short sequence of Tyr-Gly-Gly-Phe-Met. Lowest energies of Met-enkephalin without explicit solvation effects were previously determined based on the ECEPP potentials (ECEPP/2 and ECEPP/3) [127].

Here, in thesis Met-enkephalin along with the some other peptides are used to evaluate the performance of the VS algorithm on the protein folding problem by using ECEPP/3 potential. The list of peptides used in the experiments and their amino acid sequences are provided in Table 6.9. Experiments are performed with an Intel 1.86 GHz 2GB RAM notebook under the Linux Mint 16.0 environment. The VS algorithm is coded in Python by using PyCharm Community Edition and used in conjunction with the SMMP software package. Obtained results of the VS algorithm are compared to the results those found by the simulated annealing (SA) algorithm.

For the SA algorithm, library provided by the SMMP software package is utilized. Both the VS algorithm and the SA algorithm are run for 10,000 iterations to provide a true comparison. Provided results are the best results of ten different trials.

Table 6.9 : A list of peptides used in the experiments.

PDB ID	Seq. Length	Sequence Information
1PLW	5	Tyr-Gly-Gly-Phe-Met
1UAO	10	Gly-Tyr-Asp-Pro-Glu-Thr-Gly-Thr-Trp-Gly
1C98	10	Gly-Asn-Leu-Trp-Ala-Thr-Gly-His-Phe-Met
1QCM	11	Gly-Ser-Asn-Lys-Gly-Ala-Ile-Ile-Gly-Leu-Met

In Table 6.10, results found by the VS algorithm is given for the Meth-enkephalin (1PLW) peptide along with the result corresponding to the best known minimum free energy and the result that is found by the SA algorithm. As shown in this table, the SA algorithm performs better than the VS algorithm for the protein folding problem. However, both of the algorithms fail to find the best known minimum energy of the 1PLW peptide obtained by the ECEPP/3 potential. In Figure 6.13a-d, experimentally determined structure of the 1PLW is shown along with the structure with the best known minimum energy, the structure found by the VS algorithm, and the structure found by the SA algorithm.

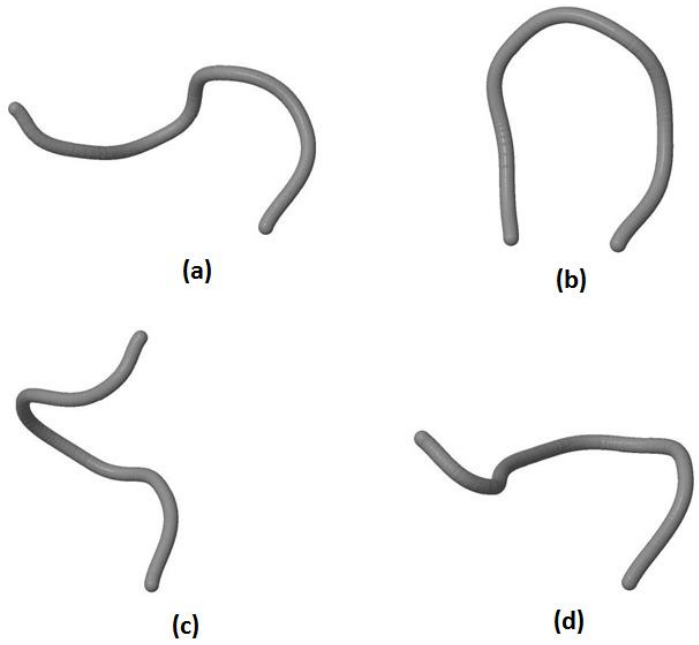

Figure 6.13 : (a) Experimentally determined structure of the 1PLW. (b) Structure with the best known minimum free energy. (c) Best structure found by the SA algorithm (d) Best structure found by the VS algorithm.

From this Figure 6.13, it is clear that none of the structures given in Figure 6.13b-d are similar to the experimentally determined structure of the 1PLW peptide which is given in Figure 6.13a.

Table 6.10 : Internal coordinates corresponding to the energies found by SA and VS algorithms for the peptide 1PLW (Met-enkephalin).

	Torsion	Minimum	SA algorithm	VS algorithm
Tyr^1	χ_1	-173.2	70.364	-59.613
	χ_2	-100.7	-85.486	94.703
	χ_6	13.7	-26.352	0.369
	φ	-83.1	163.809	69.953
	ψ	155.8	-2.675	18.808
	ω	-177.1	-166.982	176.273
Gly^2	φ	-154.2	95.379	81.931
	ψ	85.8	-155.061	-70.399
	ω	168.5	177.257	-177.989
Gly^3	φ	83.0	63.721	-67.857
	ψ	-75.0	-99.246	-28.371
	ω	-170.0	-179.445	177.434
Phe^4	χ_1	58.9	173.772	63.413
	χ_2	-85.5	-116.357	-83.215
	φ	-136.8	-74.466	-154.350
	ψ	19.1	-25.789	155.731
	ω	-174.1	-176.641	-177.159
Met^5	χ_1	52.9	-65.781	57.546
	χ_2	175.3	-174.501	-172.156
	χ_4	-179.9	179.875	84.865
	χ_2	-178.6	59.883	-59.775
	φ	-163.4	-89.994	-167.953
	ψ	160.8	118.580	-11.286
	ω	-179.8	-172.214	-179.698
E (kcal/mol)		-12.43	-7.99	-7.13

In Figure 6.14a-c, Figure 6.15a-c and Figure 6.16a-c, obtained structures for the 1UAO, 1C98 and 1QCM peptides are respectively given along with the PDB structures of each peptide. As shown from these figures, SA algorithm performs better than the VS algorithm for all of the peptides and obtained results by the SA algorithm is more similar to the experimentally determined PDB structure when compared to the structures those found by the VS algorithm. Note that, the obtained energy values for both the SA and the VS algorithms presumably are not the minimum free energies of these peptides. Therefore, the minimum free energy

conformations of these peptides may be quite similar to the experimentally determined PDB structures of these peptides.

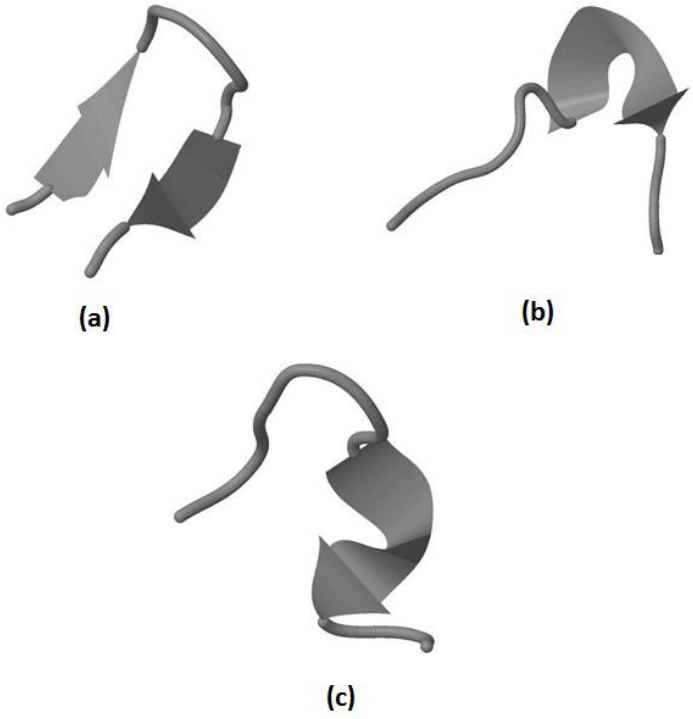

Figure 6.14 : (a) Experimentally determined structure of the 1UAO. (b) Best structure found by the SA algorithm, E = -69.70 kcal/mol. (c) Best structure found by the VS algorithm, E = -44.16 kcal/mol.

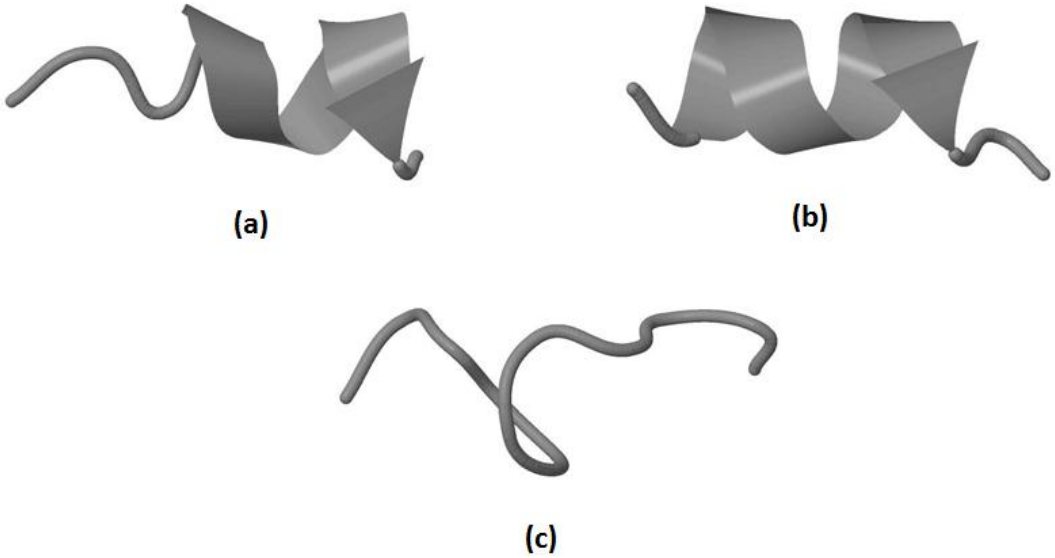

Figure 6.15 : (a) Experimentally determined structure of the 1C98. (b) Best structure found by the SA algorithm, E = -50.51 kcal/mol. (c) Best structure found by the VS algorithm, E = -32.84 kcal/mol.

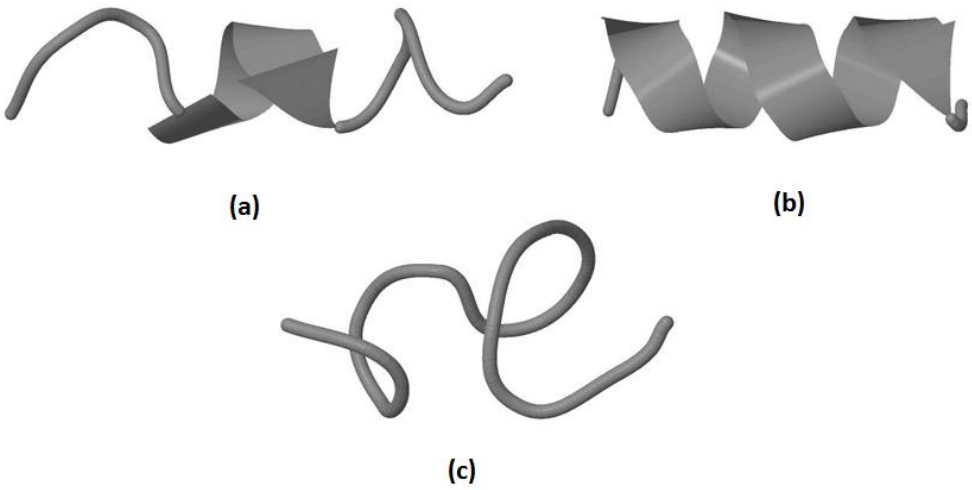

Figure 6.16 : (a) Experimentally determined structure of the 1QCM. (b) Best structure found by the SA algorithm, E = -36.67 kcal/mol. (c) Best structure found by the VS algorithm, E = -23.12 kcal/mol.

In Figure 6.17, average computational time of 10,000 iterations performed by the SA and VS algorithms is also provided for each peptide. As shown from this figure, the VS algorithm is rapid than the SA algorithm. Especially for the peptide 1QCM, the required computational time for the SA algorithm is doubled the VS algorithm. For the 1QCM peptide, the number of torsion angles are higher than the other peptides which increases the problem dimension for this particular peptide. As the problem dimension increases, the SA algorithm requires much more computational time to perform 10,000 iterations when compared to the VS algorithm.

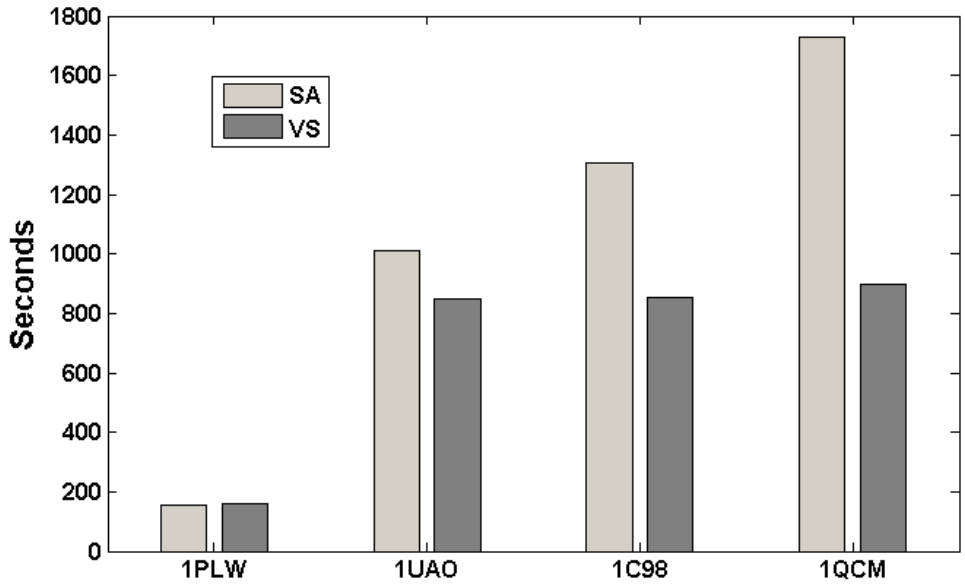

Figure 6.17 : Average computational time of 10,000 iterations performed by the SA and VS algorithms for each peptide.

7. CONCLUSION

This thesis presented artificial intelligence methods for the solution of protein folding problem both in coarse-grained and all-atom models. Coarse-grained models studied in thesis thesis include the well known HP lattice and AB off-lattice models. For the all-atom model simulations, the ECEPP force field is utilized along with the SMMP software package.

Lattice models benefit greatly from the discretization of protein phase space. Thus, the algorithms implemented for the solution of the protein folding problem by using the HP lattice model should perform well in a discrete search space. From this point of view, in this thesis a reinforcement learning based method is proposed for the solution of the protein folding. It is widely known that the reinforcement learning algorithms perform quite well for the robot path planning algorithms which are discrete in their nature and share quite similarities to the protein folding problem in lattice models. The proposed method introduced the concept of learning to the protein folding problem which is not the case for the existing algorithms. Experimental results showed that the proposed method performs quite well for the benchmark amino acid sequence set. However, the proposed method still requires some improvements. For long sequences the size of the state-action space is still huge and this issue will be addressed in our future studies.

Different from the HP lattice models, off-lattice AB model performs in continuous search space and thus, the algorithms proposed for the solution of the off-lattice AB model should be continuous search algorithms. From this point of view, in this thesis a new continuous optimization algorithm, the Vortex Search (VS) algorithm, is implemented for the solution of protein folding problem. Experimental results showed that the VS algorithm performs quite well on the benchmark numerical function set. Unfortunately, the proposed VS algorithm did not perform well on the protein folding problem because of the energy landscape provided by the energy function of the off-lattice AB model. This function has a number of deep valleys and hills and thus it is easy for the algorithms to being trapped in a local minimum point

during the search process. To overcome this drawback, in the literature usually algorithmic improvements are proposed. In this thesis, different from the literature, the original energy function is modified to provide a more smoothed search space for the algorithms. Thus, even the standard search algorithms could find the near optimal solutions easily. Experimental results showed that, the modified energy function with the VS algorithm can find the near optimal solutions much easier than the standard function. However, for long amino acid sequences the proposed energy function still need improvements. Future directions for the off-lattice AB model mainly covers the improvement of this modified energy function.

Finally, for all-atom model simulations of the protein folding problem, the ECEPP force field is combined into the VS algorithm in conjuction with the SMMP software package. A number of small peptides used in the experiments to evaluate the performance of the VS algorithm. Experimental results showed that, the VS algorithms performs worst than the SA algorithm. However, the obtained results are comparable. In the future studies, the ways of improving the VS algorithm for the solution of protein folding problem will be searched. For this purpose, the VS algorithm can also be combined into the other well-known metaheuristic algorithms.

REFERENCES

[1] **May, P.A.O**. (2010). *Protein Folding Simulations: Confinement, External Fields and Sequence Design* (Doctoral dissertation). Retrieved from https://kobra.bibliothek.uni-kassel.de/bitstream/urn:nbn:de:hebis:34-2010040832570/5/DissertationOjedaMay.pdf

[2] **Tramontano, A., and Arthur, M. L.** (2006). *Protein structure prediction: concepts and applications*. Weinheim: Wiley-VCH, 2006.

[3] **Hoque, Md T.** (2007). *Genetic algorithm for Ab initio protein structure prediction based on low resolution models* (Doctoral dissertation). Monash University, Gippsland School of Information Technology, Australia.

[4] **Pi, J., and Lee, S.** (2013). Mass Spectrometry Coupled Experiments and Protein Structure Modeling Methods. *International journal of molecular sciences*, 14.10, 20635-20657.

[5] **Nanias, M.** (2005). *Protein Folding with Coarse-Grained Off-Lattice Models of the Polypeptide Chain*. (Doctoral dissertation). Retrieved from https://ecommons.library.cornell.edu/bitstream/1813/2549/1/nanias-thesis.pdf

[6] **Zhang, Y., and Jeffrey, S.** (2005). The protein structure prediction problem could be solved using the current PDB library. *Proceedings of the National Academy of Sciences of the United States of America*, **102.4**, 1029-1034.

[7] **Helles, G.** (2008). A comparative study of the reported performance of ab initio protein structure prediction algorithms. *Journal of the Royal Society Interface*, 5.21, 387-396.

[8] **Reynaud, E.** (2010). Protein misfolding and degenerative diseases. *Nat Educ*, 3.9, 28.

[9] **Williams, H. W.** (2011). *Computer simulations of protein folding* (Doctoral dissertation). Retrieved from http://eprints.nottingham.ac.uk/12180/1/thesis.pdf

[10] **Dill, K. A.** (1985). Theory for the folding and stability of globular proteins. *Biochemistry*, 24.6, 1501-1509.

[11] **Zhang, J, et al.** (2009). Protein folding simulations: From coarse-grained model to all-atom model. *Iubmb Life* 61.6, 627-643.

[12] **Stillinger, F. H., Head-Gordon, T., and L. Hirshfeld, C.** (1993). Toy model for protein folding. *Physical review E* 48.2, 469.

[13] **Weiner, P. K., and Kollman, P.A.** (1981). AMBER: Assisted model building with energy refinement. A general program for modeling molecules

and their interactions. *Journal of Computational Chemistry*, 2.3, 287-303.

[14] **Brooks, B. R., et al.** (1983). CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of computational chemistry*, 4.2, 187-217.

[15] **Scott, W. RP, et al.** (1999). The GROMOS biomolecular simulation program package. *The Journal of Physical Chemistry A*, 103.19, 3596-3607.

[16] **Momany, F. A., et al.** (1975). Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids. *The Journal of Physical Chemistry*, 79.22, 2361-2381.

[17] **Doğan, B., and Ölmez, T.** (2015). A novel state space representation for the solution of 2D-HP protein folding problem using reinforcement learning methods. *Applied Soft Computing*, 26, 213-223.

[18] **Doğan, B., and Ölmez, T.** (2015). A new metaheuristic for numerical function optimization: Vortex Search algorithm. *Information Sciences*, 293, 125-145.

[19] **Doğan, B., and Ölmez, T.** (2015). Modified off-lattice AB Model for protein folding problem using the Vortex Search algorithm. *International Journal of Machine Learning and Computing*, (accepted)

[20] **McKee, T., and McKee, J. R.** (2012). *Biochemistry: the molecular basis of life*. Oxford University Press.

[21] **Whitford, D.** (2013). *Proteins: structure and function*. John Wiley and Sons.

[22] **Huang, Y.Y.** (2004). Protein folding prediction with genetic algorithms, *PhD Thesis*, National Sun Yat-sen University.

[23] **Mishra, N. C.** (2011). *Introduction to proteomics: principles and applications*. Vol. 148. John Wiley and Sons

[24] **Benvenuti, M., and Mangani, S.** (2007). Crystallization of soluble proteins in vapor diffusion for x-ray crystallography. *Nature protocols*, 2.7, 1633-1651.

[25] **Gamalielsson, J.** (2001). Models for Protein Structure Prediction by Evolutionary Algorithms, *PhD Thesis*, University of Skövde.

[26] <*http://www.rcsb.org/pdb/home/home.do*>, date retrieved 02.03.2015.

[27] <*http://en.wikipedia.org/wiki/Protein_Data_Bank_%28file_format%29* >, date retrieved 02.03.2015.

[28] **Taketomi, H., Yuzo, U., and Nobuhiro, G.** (1975). Studies on protein folding, unfolding and fluctuations by computer simulation. *International journal of peptide and protein research*, 7.6, 445-459.

[29] **Shmygelska, A., and Holger, H. H.** (2005). An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC bioinformatics*, 6.1, 30.

[30] **Thachuk, C., Shmygelska, A., and Holger, H. H.** (2007). A replica exchange Monte Carlo algorithm for protein folding in the HP model. *BMC bioinformatics*, 8.1, 342.

[31] **Huang, C., Xiangbo Y., and Zhihong, H.** (2010). Protein folding simulations of 2D HP model by the genetic algorithm based on optimal secondary structures. *Computational Biology and Chemistry*, 34.3, 137-142.

[32] **Rego, C., Haitao L., and Glover, F.** (2011). A filter-and-fan approach to the 2D HP model of the protein folding problem. *Annals of Operations Research*, 188.1, 389-414.

[33] **Hsieh, SY., and Lai, DW.** (2011). A new branch and bound method for the protein folding problem under the 2D-HP model. *NanoBioscience, IEEE Transactions on*, 10.2, 69-75.

[34] **Su, SC, Lin, CJ and Ting, CK.** (2011). An effective hybrid of hill climbing and genetic algorithm for 2D triangular protein structure prediction. *Proteome science*, 9.1, S19.

[35] **Liu, J., Gang, L., and Jun, Y.** (2011). Protein-folding simulations of the hydrophobic-hydrophilic model by combining pull moves with energy landscape paving. *Physical Review E*, 84.3, 031934.

[36] **Liu, J., et al.** (2012). Heuristic energy landscape paving for protein folding problem in the three-dimensional HP lattice model. *Computational biology and chemistry*, 38, 17-26.

[37] **Tsay, JJ., and Su, SC.** (2013). An effective evolutionary algorithm for protein folding on 3D FCC HP model by lattice rotation and generalized move sets. *Proteome science*, 11.1, 19.

[38] **Islam, ASM S., and Rahman, M. S.** (2013). On the protein folding problem in 2D-triangular lattices. *Algorithms for Molecular Biology*, 8.1.

[39] **Zhang, Y., Lenan, W., and Shuihua, W.** (2013). Solving two-dimensional HP model by firefly algorithm and simplified energy function. *Mathematical Problems in Engineering*.

[40] **Rashid, M. A., et al.** (2013). Spiral search: a hydrophobic-core directed local search for simplified PSP on 3D FCC lattice. *BMC bioinformatics*, 14.2, 16.

[41] **Maher, B., et al.** (2014). A firefly-inspired method for protein structure prediction in lattice models. *Biomolecules*, 4.1, 56-75.

[42] **Su, SC., and Jyh, J. T.** (2014). A Novel Offspring Selection Strategy in GAs for Protein Structure Prediction. *Computer, Consumer and Control (IS3C), 2014 International Symposium on*. IEEE.

[43] **García-Martínez, J. M., et al.** (2014). An efficient approach for solving the HP protein folding problem based on UEGO. *Journal of Mathematical Chemistry*, 1-13.

[44] **Czibula, G., Bocicor, MI., and Czibula, IG.** (2011). A distributed reinforcement learning approach for solving optimization problems. *Proceedings of the 5th WSEAS international conference on*

Communications and information technology. World Scientific and Engineering Academy and Society (WSEAS), 2011.

[45] **Czibula, G., Bocicor, MI., and Czibula, IG.** (2011). A reinforcement learning model for solving the folding problem. *International Journal of Computer Technology and Applications,* 2, 171-182.

[46] **Czibula, G., Bocicor, MI., and Czibula, IG.** (2011). An experiment on protein structure prediction using reinforcement learning. *Studia Universitatis Babes-Bolyai, Informatica,* 56.1.

[47] **Czibula, G., Bocicor, MI., and Czibula, IG.** (2011). Solving the protein folding problem using a distributed q-learning approach." *International Journal of Computer,* 5.3, 404-413.

[48] **Ma, X., et al.** (2013). State-chain sequential feedback reinforcement learning for path planning of autonomous mobile robots. *Journal of Zhejiang University Science C,* 14.3, 167-178.

[49] **Sendari, S., Shingo, M., and Kotaro, H.** (2011). Fuzzy genetic Network Programming with Reinforcement Learning for mobile robot navigation. *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*.

[50] **Iima, H., Yasuaki, K., and Shoko, M.** (2010). Swarm reinforcement learning method based on ant colony optimization. *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*.

[51] **Iima, H., and Yasuaki, K.** (2008). Swarm reinforcement learning algorithms based on particle swarm optimization." *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*.

[52] **Iima, H., and Yasuaki, K.** (2008). Swarm reinforcement learning algorithms based on Sarsa method. *SICE Annual Conference, 2008*.

[53] **Iima, H., and Yasuaki, K.** (2010). Swarm reinforcement learning method based on an actor-critic method. *Simulated Evolution and Learning*. Springer Berlin Heidelberg, 279-288.

[54] **Iima, H., Yasuaki, K., and Kazuo, E.** (2011). Swarm reinforcement learning methods for problems with continuous state-action space. *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*.

[55] **Gambardella, L. M., and Dorigo, M.** (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. *In Proceedings of Twelfth International Conference on Machine Learning*, 252–260.

[56] **Dorigo, M., Birattari, M., and Stutzle, T.** (2006). Ant colony optimization. *Computational Intelligence Magazine, IEEE,* 1.4, 28-39.

[57] **Dorigo, M. (1992).** Optimization, learning and natural algorithms. *PhD. Thesis,* Politecnico di Milano, Italy.

[58] **Dorigo, M., Maniezzo, V., and Colorni, A.** (1996). Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26.1, 29-41.

[59] **Liu, J., et al.** (2005). Analysis of toy model for protein folding based on particle swarm optimization algorithm. *Advances in Natural Computation*. Springer Berlin Heidelberg, 636-645.

[60] **Bachmann, M., Arkın, H., and Janke, W.** (2005). Multicanonical study of coarse-grained off-lattice models for folding heteropolymers. *Physical review E,* 71.3, 031906.

[61] **Chen, M., and Huang, W.** (2006). Heuristic algorithm for off-lattice protein folding problem. *Journal of Zhejiang University SCIENCE B,* 7.1, 7-12.

[62] **Zhang, X., and Tingting, L.** (2007). Improved particle swarm optimization algorithm for 2D protein folding prediction. *Bioinformatics and Biomedical Engineering, 2007. ICBBE 2007. The 1st International Conference on*.

[63] **Jingfa, L.** (2009). Structure optimization by heuristic algorithm in a coarse-grained off-lattice model. *Chinese Physics B,* 18.6, 2615.

[64] **Jingfa, L. et al.** (2009). Structure optimization of the two-dimensional off-lattice hydrophobic–hydrophilic model. *Journal of biological physics* 35.3, 245-253.

[65] **Arkin, H.** (2010). Comparison of the Parallel Tempering Algorithm and Multicanonical Method as Applied to Coarse-Grained Off-lattice Models for Folding Heteropolymers. *Journal of Statistical Physics,* 139.2, 326-332.

[66] **Kalegari, D. H., and Heitor, S. L.** (2010). An improved parallel differential evolution approach for protein structure prediction using both 2D and 3D off-lattice models. *Differential Evolution (SDE), 2013 IEEE Symposium on*.

[67] **Chen, X., et al.** (2011). An improved particle swarm optimization for protein folding prediction. *International Journal of Information Engineering and Electronic Business (IJIEEB),* 3.1, 1.

[68] **Mansour, R. F.** (2011). Applying an evolutionary algorithm for protein structure prediction. *American Journal of Bioinformatics Research,* 1.1, 18-23.

[69] **Jingfa, L. et al.** (2013). Heuristic-based tabu search algorithm for folding two-dimensional AB off-lattice model proteins. *Computational biology and chemistry* 47, 142-148.

[70] **Li, B., Li, Y., and Ligang, G.** (2014). Protein secondary structure optimization using an improved artificial bee colony algorithm based on AB off-lattice model. *Engineering Applications of Artificial Intelligence* 27, 70-79.

[71**] Talbi, EG.** (2009). *Metaheuristics: from design to implementation*. Vol. 74. John Wiley and Sons.

[72] **Beheshti, Z., and Shamsuddin, S.M.H.** (2013). A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl,* 5.1, 1-35.

[73] **Boussaïd, I., Lepagnot, J., and Siarry, P.** (2013). A survey on optimization metaheuristics. *Information Sciences*, 237, 82-117.

[74] **Kirkpatrick, S., DG Jr., and Vecchi, MP.** (1983). Optimization by simulated annealing. *Science*, 220.4598, 671-680.

[75] **Černý, V.** (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45.1, 41-51.

[76] **Glover, F.** (1985). Future Paths for Integer Programming and Links to Artificial Intelligence, CAAI Report 85-8. *Center for Applied Artificial Intelligence, University of Colorado, October*.

[77] **Lourenço, H. R., Olivier, C. M., and Stützle, T.** (2010). Iterated local search: Framework and applications. *Handbook of Metaheuristics*. Springer US, 363-397.

[78] **Voudouris, C., and Tsang, E PK.** *Guided local search*. Springer US, 2003.

[79] **Hooke, R., and Jeeves, T.A.** (1961). 'Direct Search' Solution of Numerical and Statistical Problems. *Journal of the ACM (JACM)* 8.2, 212-229.

[80] **Rastrigin, L. A.** (1964). The convergence of the random search method in the extremal control of a many parameter system. *Automation and Remote Control*, 24.11, 1337.

[81] **Hansen, P., Nenad, M., and Pérez, J.A.M.** (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research* 175.1, 367-407.

[82] **Holland, J. H.** (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.

[83] **Storn, R., and Price, K.** (1995). *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*, Vol. 3. Berkeley: ICSI.

[84] **Storn, R., and Price, K.** (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11.4, 341-359.

[85] **Larrañaga, P., and Lozano, J. E.** (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*, Vol. 2. Springer Science & Business Media.

[86] **Kennedy, K., Eberhart, R.C.,** (1995). Particle Swarm Optimization. *IEEE International Conference on Neural Networks*, 4, 1942–1948.

[87] **Karaboga, D. (2005).** *An idea based on honey bee swarm for numerical optimization*. Vol. 200. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

[88] **Basturk, B., and Karaboga, D.** (2006). An artificial bee colony (ABC) algorithm for numeric function optimization. *IEEE swarm intelligence symposium*.

[89] **Karaboga, D., and Basturk, B.** (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization*, 39.3, 459-471.

[90] **Karaboga, D., and Basturk, B.** (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied soft computing*, 8.1, 687-697.

[91] **Schumer, M., and Steiglitz, K.** (1968). Adaptive step size random search. *Automatic Control, IEEE Transactions on*, 13.3, 270-276.

[92] **Schrack, G., and Choit, M.** (1976). Optimized relative step size random searches. *Mathematical Programming*, 10.1, 230-244.

[93] **Birattari, M., et al.** (2001). Classification of Metaheuristics and Design of Experiments for the Analysis of Components. Technical report-AIDA-01-05 Darmstadt University of Technology.

[94] **Andrews, L. C.** (1985). *Special functions for engineers and applied mathematicians*. New York: Macmillan.

[95] **Gautschi, W.** (1999). A note on the recursive calculation of incomplete gamma functions. *ACM Transactions on Mathematical Software (TOMS)*, 25.1, 101-107.

[96] **Winitzki, S.** (2003). Computing the incomplete gamma function to arbitrary precision. *Computational Science and Its Applications—ICCSA 2003*. Springer Berlin Heidelberg, 790-798.

[97] **Giampietro, A. and Besenghi, R.** (1986). Numerical calculation of incomplete gamma functions by the trapezoidal rule. *Numerische Mathematik*, 50.4, 419-428.

[98] **Decatur, S. E.** (1996). Protein folding in the generalized hydrophobic-polar model on the triangular lattice. *T echnical M emo M It-Lc S.*

[99] **Jiao, Y.** (2012). The development of accurate force fields for protein simulation, *PhD Thesis*, Kansas State University.

[100] **Leach, A. R.** (2001). *Molecular modelling: principles and applications*. Pearson Education.

[101] **Vázquez, R., Aurea, N., and Félix., F.R.** (2014) *Computational chemistry applied in the analyses of chitosan/polyvinylpyrrolidone/mimosa tenuiflora*. Science Publishing Group.

[102] **Dalit, S.B. et al.** (2005). Monte Carlo studies of folding, dynamics, and stability in α-helices. *Biophysical journal*, 88.4, 2391-2402.

[103] **Eisenmenger, F., and Hansmann, U HE.** (1997). Variation of the Energy Landscape of a Small Peptide under a Change from the ECEPP/2 Force Field to ECEPP/3. *The Journal of Physical Chemistry B*, 101.16, 3304-3310.

[104] **Eisenmenger, F., et al.** (2001). [SMMP] A modern package for simulation of proteins. *Computer physics communications*, 138.2, 192-212.

[105] **Eisenmenger, F., et al.** (2006). An enhanced version of SMMP—open-source software package for simulation of proteins. *Computer physics communications*, 174.5, 422-429.

[106] **Meinke, J. H., et al.** (2008). SMMP v. 3.0—simulating proteins and protein interactions in Python and Fortran. *Computer Physics Communications*, 178.6, 459-470.

[107] **Karaboga, D., and Akay, B.** (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214.1, 108-132.

[108] **Clerc, M.** (2012). Standard Particle Swarm Optimisation. "The Particle Swarm Central."

[109] <*http://www.particleswarm.info/Programs.html*>, date retrieved 02.03.2015.

[110] **Ghosh, S., et al.** (2012). A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization. *Information Sciences*, 182.1, 199-219.

[111] **Han, M., Chuang, L., and Jun, X.** (2014). An evolutionary membrane algorithm for global numerical optimization problems. *Information Sciences*, 276, 219-241.

[112] **Espitia, H. E., and Sofrony, J. I.** (2013). Vortex particle swarm optimization. *Evolutionary Computation (CEC), 2013 IEEE Congress on*.

[113] **LaTorre, A., Santiago, M., and Peña, JM.** (2011). A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. *Soft Computing* 15.11, 2187-2199.

[114] **Ganapathy, K., et al.** (2014). Hierarchical particle swarm optimization with ortho-cyclic circles. *Expert Systems with Applications*, 41.7, 3460-3476.

[115] **Gao, WF., Liu, SY. and Huang, LL.** (2014). Enhancing artificial bee colony algorithm using more information-based search equations. *Information Sciences* 270, 112-133.

[116] **Chen, SM., Sarosh, A., and Dong, YF.** (2012). Simulated annealing based artificial bee colony algorithm for global numerical optimization. *Applied mathematics and computation*, 219.8, 3575-3589.

[117] **Vaz, A. I. F., and Luís N. V.** (2009). PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods & Software*, 24.4-5, 669-685.

[118] **Akay, B., and Karaboga, D.** (2012). A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, 192, 120-142.

[119] **Wang, L., Bo, Y., and Chen, Y.** (2014). Improving particle swarm optimization using multi-layer searching strategy. *Information Sciences*, 274, 70-94.

[120] **Neri, F., Mininno, E. and Iacca, G.** (2013). Compact particle swarm optimization. *Information Sciences*, 239, 96-121.

[121] **Han, H. et al.** (2012). Example-based learning particle swarm optimization for continuous optimization. *Information Sciences*, 182.1, 125-138.

[122] **Yang, XS., et al.** (2013). *Swarm intelligence and bio-inspired computation: theory and applications*. Newnes.

[123] **Dall'Igna Júnior, A., et al.** (2004). Performance and parameterization of the algorithm Simplified Generalized Simulated Annealing. *Genetics and Molecular Biology*, 27.4, 616-622.

[124] <*http://mf.erciyes.edu.tr/abc/* >, date retrieved 02.03.2015.

[125] <*http://web.itu.edu.tr/bdogan/VortexSearch/VS.htm*>, date retrieved 02.03.2015.

[126] **Civicioglu, P.** (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation*, 219.15, 8121-8144.

[127] **Zhan, L., Chen, J. ZY., and Liu, WK**. (2006). Conformational study of met-enkephalin based on the ECEPP force fields. *Biophysical journal*, 91.7, 2399-2404.

APPENDICES

APPENDIX A: Parameters of some benchmark numerical functions.

APPENDIX A: Parameters of some benchmark numerical functions.

Table A.1 : A parameter of the Fletcher-Powell Function.

i	A_{ij}, $j = 1, \ldots, 10$									
1	−79	56	−62	−9	92	48	−22	−34	−39	−40
2	91	−9	−18	−59	99	−45	88	−14	−29	26
3	−38	8	−12	−73	40	26	−64	29	−82	−32
4	−78	−18	−49	65	66	−40	88	−95	−57	10
5	−1	−43	93	−18	−76	−68	−42	22	46	−14
6	34	−96	26	−56	−36	−85	−62	13	93	78
7	52	−46	−69	99	−47	−72	−11	55	−55	91
8	81	47	35	55	67	−13	33	14	83	−42
9	5	−43	−45	46	56	−94	−62	52	66	55
10	−50	66	−47	−75	89	−16	82	6	−85	−62

Table A.2 : B parameter of the Fletcher-Powell Function.

i	B_{ij}, $j = 1, \ldots, 10$									
1	−65	−11	76	78	30	93	−86	−99	−37	52
2	59	67	49	−45	52	−33	−34	29	−39	−80
3	21	−23	−80	86	86	−30	39	−73	−91	5
4	−91	−75	20	−64	−15	17	−89	36	−49	−2
5	−79	99	−31	−8	−67	−72	−43	−55	76	−57
6	−89	−35	−55	75	15	−6	−53	−56	−96	87
7	−76	45	74	12	−12	−69	2	71	75	−60
8	−50	−88	93	68	10	−13	84	−21	65	14
9	−23	−95	99	62	−37	96	27	69	−64	−92
10	−5	−57	−30	−6	−96	75	25	−6	96	77

Table A.3 : α parameter of the Fletcher-Powell function.

α_j, $j = 1, \ldots, 10$
−2.7910
2.5623
−1.0429
0.5097
−2.8096
1.1883
2.0771
−2.9926
0.0715
0.4142

Table A.4 : *a* parameter of the FoxHoles function.

j	a_{ij}, $i = 1, 2$	
1	−32	−32
2	−16	−32
3	0	−32
4	16	−32
5	32	−32
6	−32	−16
7	−16	−16
8	0	−16
9	16	−16
10	32	−16
11	−32	0
12	−16	0
13	0	0
14	16	0
15	32	0
16	−32	16
17	−16	16
18	0	16
19	16	16
20	32	16
21	−32	32
22	−16	32
23	0	32
24	16	32
25	32	32

Table A.5 : *a* and *b* parameters of the Kowalik function.

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Table A.6 : *a* and *c* parameters of the Shekel functions.

i	a_{ij}, j = 1, …, 4				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Table A.7 : *a*, *c* and *p* parameters of the 3-parameter Hartman function.

i	a_{ij}, j = 1, 2, 3			c_i	p_{ij}, j = 1, 2, 3		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

Table A.8 : *a*, *c* and *p* parameters of the 6-parameter Hartman function.

i	a_{ij}, j = 1, ..., 6						c_i	p_{ij}, j = 1, ..., 6					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

CURRICULUM VITAE

Name Surname: Berat Doğan

Place and Date of Birth: Malatya, 1985

E-Mail: bdogan@itu.edu.tr

EDUCATION:

B.Sc.: Erciyes University, Electronics Engineering

M.Sc.: Istanbul Technical University, Biomedical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

Istanbul Technical University, Research Assistant, 2009-2015

Nortel Networks Netaş, Software Engineer, 2008 - 2009

TÜBİTAK-BİDEB Scholarship for Ph.D. Degree

TÜBİTAK-BİDEB Scholarship for M.Sc. Degree

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Doğan B.**, Ölmez T., A novel state-space representation for the solution of 2D-HP protein folding problem using reinforcement learning methods, *Applied Soft Computing*, Volume 26, January 2015, Pages 213-223

- **Doğan B.**, Ölmez T., A new metaheuristic for numerical function optimization: Vortex Search algorithm, *Information Sciences*, Volume 293, February 2015, Pages 125-145

- **Doğan B.**, Ölmez T., Modified off-lattice AB model for protein folding problem using the Vortex Search algorithm, *International Journal of Machine Learning and Computing*, vol. 5, no. 4, pp. 329-333, 2015 (from ICMLC 2015 conference)

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS :

- **Doğan B.**, Ölmez T., Vortex search algorithm for the analog active filter component selection problem, *AEÜ International Journal of Electronics and Communications*, doi: 10.1016/j.aeue.2015.05.005 (in press)
- **Doğan B.**, Korürek M., A new ECG beat clustering method based on kernelized fuzzy c-means and hybrid ant colony optimization for continuous domains, *Applied Soft Computing*, Volume 12, Issue 11, November 2012, Pages 3442-3451
- **Doğan B.**, Korürek M., ECG beat classification using particle swarm optimization and radial basis function neural network, *Expert Systems with Applications*, Volume 37, Issue 12, December 2010, Pages 7563-7569
- Taşkın V.**, Doğan B.**, Ölmez T., Prostate cancer classification from mass spectrometry data by using wavelet anaylsis and kernel partial least squares algorithm, *International Journal of Bioscience, Biochemistry and Bioinformatics*, vol. 3, no. 2, pp. 98-102, 2013
- **Doğan B.**, Ölmez T., Fuzzy clustering of ECG beats using a new metaheuristic approach, *IWBBIO 2014*, 7-9 April 2014, Granada, Spain
- **Doğan B.**, Yüksel A., Analog filtrelerin grup gecikmelerinin girdap arama algoritması ile optimizasyonu, *SIU 2015*, kabul edildi.
- Toker İ., **Doğan B.**, Kent Pınar S., Beyin MR görüntülerinin Tip-2 bulanık kümeleme algoritması ile bölütlemesi, *SIU 2015*, kabul edildi.
- Toker İ., **Doğan B.**, Kent Pınar S., EKG vurularının Tip-2 bulanık öbekleştirme algoritması ile öbekleştirilmesi, *TIPTEKNO 2014*, 25-27 Eylül 2014, Kapadokya
- Taşkın V., **Doğan B.**, Ölmez T., Yumurtalık kanserinin kütle spektrometresi verilerinden kısmi en küçük kareler yöntemi ile teşhisi, *BIYOMUT 2012*, 3-5 Ekim 2012, İstanbul
- **Doğan B.**, Korürek M., EKG vurularının bulanık c-ortalamalar algoritması ve parçacık sürü optimizasyonu yardımıyla öbekleştirilmesi, *SIU 2012*, 18-20 Nisan 2012, Muğla
- **Doğan B.**, Korürek M., EKG vurularının sürekli zaman karınca koloni optimizasyonu yardımıyla sınıflandırılması, *ELECO 2010*, 2-5 Araık 2010, Bursa
- **Doğan B.**, Korürek M., EKG vurularını sınıflamada radyal tabanlı fonksiyon yapay sinir ağının performans değerlendirmesi, *BİYOMUT 2009*, Dokuz Eylül Üniv. İzmir