Soft Computing Techniques for the Protein Folding Problem on High Performance Computing Architectures.

Antonio Llanes[a], Andrés Muñoz[b], Andrés Bueno-Crespo[a], Teresa García-Valverde[b], Antonia Sánchez[a], Francisco Arcas-Túnez[b], Horacio Pérez-Sánchez[a], José M. Cecilia*[a]

[a]*Computer Science Department, Bioinformatics and High Performance Computing Research Group (BIOHPC), Universidad Católica San Antonio de Murcia (UCAM). Campus de los Jerónimos, s/n Guadalupe 30107 (Murcia) - Spain.*
[b]*Computer Science Department, Universal Knowledge Enhancement by Multidisciplinary Implementation (UKEIM), Universidad Católica San Antonio de Murcia (UCAM). Campus de los Jerónimos, s/n Guadalupe 30107 (Murcia) - Spain.*

Abstract: The protein-folding problem has been extensively studied during the last fifty years. The understanding of the dynamics of global shape of a protein and the influence on its biological function can help us to discover new and more effective drugs to deal with diseases of pharmacological relevance. Different computational approaches have been developed by different researchers in order to foresee the three-dimensional arrangement of atoms of proteins from their sequences. However, the computational complexity of this problem makes mandatory the search for new models, novel algorithmic strategies and hardware platforms that provide solutions in a reasonable time frame. We present in this revision work the past and last tendencies regarding protein folding simulations from both perspectives;hardware and software. Of particular interest to us are both the use of inexact solutions to this computationally hard problem as well as which hardware platforms have been used for running this kind of *Soft Computing* techniques.

Keywords: Soft Computing, Protein FoldingProblem, Protein Structure Prediction, Parallel Computing, Distributed Computing, Metaheuristics, High Performance Computing.

1. INTRODUCTION

1.1. Protein folding problem

A noteworthy interrelation exists at the molecular level between the structure of a protein and its biological function, and in biochemistry we can find a diversity of such functionalities. It is well known that the mechanism by which a protein exerts its biological function is directly related to its native three-dimensional structure, which is precisely codified on its sequence of aminoacids[1].

Being able to solve this problem is of outstanding importance since having access to the information related to the structure of these biomolecules, allows for being able to explain how bioactive compounds can modulate their biological activity and therefore paves the way to the drug discovery process.

In addition, one can find many more sequences than structural information, mainly due to the last advances in high-throughput sequencing and personalized medicine efforts [1-2]. Thus, a noticeable interest exists in the development of methodologies that, exploiting only information extracted from sequences, can predict in detail the structure of proteins.

1.2. The simulation problem

Finding accurate solutions of the PSP problem is very challenging, and researchers have developed many different approaches in order to solve it by means of computer simulation. These simulation methods receive as input a protein sequence and outputtheir predictions for the protein structures.

Existing computer simulation methods for the PSP problem can be classified depending on:

a) the degree of details used in the protein model that undergoes the computer simulation:there are detailed all-atom models that try to accurately represent and describe bonded and non-bonded interactions present in the folded protein structure. From the other side, coarse grain models can also be considered. In the last decades, the first theoretical hypotheses concerning protein folding, such as those stated by Dill et al.[2] were proposed. Main underlying ideas indicated that forces implied in the protein folding process were related with the intercommunication between their aminoacids. But recently, a theory that states that non-bonded interactions significantly contribute to the dynamics of this mechanism, is being accepted, and researchers are showing interest to the use of very simple models of proteinsand other biological macromolecules. In this context, the study of these coarse grain models through computer simulation techniques can yield interesting results when their predictions are contrasted with empirical measurements.

b) the scoring function used for the estimation of the interactions between the elements of the protein model:thismathematical function will mainly depend on the type of protein model used, and for a given model, it might contain different sets of parameters that describe the relative intensity of the interactions between the different elements of the protein model. Its derivation or construction depends usually on physical theories or statistical analyses performed on previously available protein structures.

c) the algorithm used for the global optimization problem of the scoring function: once a given protein model and scoring function have been chosen, a optimization methodis selected for working on the global optimization problem. It concerns the search of the most optimal value of the scoring function, since we assume that this value will correspond to the native protein fold [1]. Here it is possible to use methods that take into account the dynamics of the system, such as Molecular Dynamics [3], or stochastic methods that try to solve the optimization problem not taking into account the dynamics of the system [4]. The former is more realistic, but at the same time it is more computationally demanding, whereas the latter is much faster, but by using it we lose information about the evolution of the system.

Once we have chosen a model, scoring function and optimization algorithm, we can still consider what is the fastest way to carry out the required simulations depending on the available hardware architectures.

1.3. Combination of models, algorithms and HPC.

The choice of model and its associated algorithm is mainly motivated by the required objectives, but it is also constrained by the computer hardware characteristics attainable in the relevant time frame. One of the most widely studied models of protein folding is the hydrophobic-hydrophilic (HP) model introduced by Dill [2]. In the description of the HP model, the different amino acids that form the macromolecular chain can be seen as a discretized conformation in a three-dimensional grid or lattice. Here, one of the most relevant underlying assumptions is that hydrophobic forces contribute considerably to the folding process, and the protein chain is modeled as an array of hydrophobic or hydrophilic chains (H or P for nonpolar and polar, respectively). Then, the most optimal protein conformation is the one that augments that number of nonpolar residues that are contiguous. In this case the folding process can be described as a minimization of the free-energy of the system, and it can be considered as NP-hard problems [5]. This implies that such problems can not be efficiently processed by a computer (for insights we refer the reader to [6,7]).

Models and their associated algorithms should not be selected in isolation though. They must be evaluated in the context of the computer hardware environment they are going to run on. Algorithms that are designed to leverage maximum performance on a particular hardware architecture could become less effective on a different hardware. Therefore, the selection must be made carefully, and may change over time [8]. This issue even grows exponentially nowadays as we are witnessing the consolidation of heterogeneous systems (i.e., systems that use more than one kind of processors), mainly motivated for the exacerbated power consumption in current microprocessors, and trying to follow the wake of Moore's law. Such heterogeneity is found at different levels from laptops to large-scale computers like supercomputers, clouds, etc, and also where it emerges naturally is in the low-power devices market such as smartphones, tablet and so on. [9]. This emergent landscape of computation in the high performance computing market offers new opportunities in the simulation of protein structure prediction. However, the recent 2014 United States Department of the Energy (DoE) report on top ten exascale research challenges [8] shows as one of the main challenges for next years the design of *Exascale algorithms*. It will require redesigning, or even reinventing the algorithms used in current scientific and engineering codes, and potentially reformulating the science problems to leverage billion-way parallel architectures.

In this sense, S*oft Computing* techniques are designed to deal with the difficulties which arise in real problems by including several factors like several levels of imprecision into the calculation and taking this into account to even change the granularity of the problem or somehow relaxing the goal of optimization at some point[10]. The source of inspiration of *Soft Computing* is based on the natural processes, trying to formalize such processes to solve a particular task. Techniques within this field include neural networks, genetic algorithms (GA), evolutionary algorithms, etc., having many of them a common ingredient in their definition: parallelism as the way of speeding-up simulations and providing practical implementations for a feasible search of a single, unified and parameterized solution.

This review article shows the last tendencies on the prediction of protein structure by computer simulation and our perspectives for the forthcoming years. We focus on both the *Soft Computing* techniques that have been applied to coarse-grain protein models, such as the HP-model since it is one of the most widely used coarse-grain models in the literature, and also the underlying hardware and programming models that have been used to execute those algorithms. The paper is structured as follows: Section 2 briefly introduces the reader into the main concepts underlying this review. Section 3 shows the *Soft Computing* techniques applied to protein folding methods before discussingin Section 4 about new trends in novel algorithms and architectures related to this problem. The paper finishes with some conclusions on the current state of the art for this topic.

2. BACKGROUND

2.1. Benchmarks in protein structure prediction.

In order to test the accuracy and convenience of PSP methods it is necessary to have control data (benchmarks) so that we can check whether our predictions are reliable or not. If our particular PSP model, scoring function and algorithm can reproduce the structure of proteins for which experimental structural data is available, we can continue forward and start to make predictions for sequences for which structures are still unknown. It is therefore of outstanding importance to test our PSP methods against all possible available benchmarks.

The field of PSP benchmarks can be usually divided into experimental and synthetic ones. When working with detailed atomic models, we will be able to compare them with structural data from online public databases such as Protein Data Bank (PDB) [11]. In order to test the accuracy of protein structure prediction methods, the current "gold standard" rule is to compare the predicted structure with the experimental one, and calculate the RMSD (Root Mean

Structure Deviation) between them. This is only possible when protein structures have been obtained by experimental methods such as X-ray crystallography, nuclear magnetic resonance, or cryo electron microscopy, and deposited in public access databases such as PDB.

In the case of coarse grain models we have two options. The first one is to convert them to all-atom models and then compare with experimental structures from PDB, and the second one is (when the first possibility does not exist) to compare them with synthetic data obtained previously from other researchers who have performed an exhaustive search of the solution space of the problem.

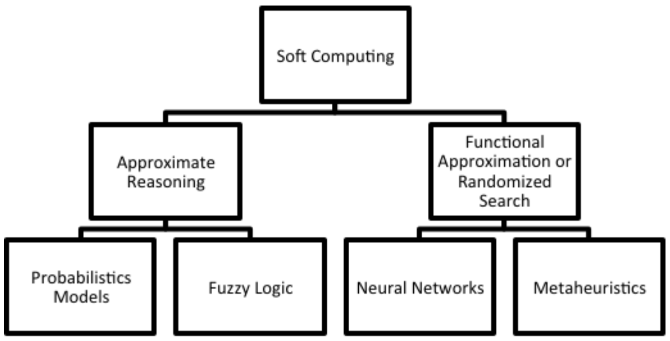

Figure 1 - Classification of Soft Computing techniques

Lastly, and independently of the detail of the method used, we might be also interested in benchmarking the computational speed of our PSP method, depending on its hardware implementation, programming language used, etc. This is also very relevant since the computational performance of the method, and the availability of computational resources the researchers have access to, will dictate the size of the systems we want to study.

2.2. Soft Computing techniques

From the algorithmic point of view, traditional hard computing techniques are based on three main objectives: precision, certainty and rigor. These requirements make the computational cost of such algorithms very costly, particularly to deal with real problems where the input size grows exponentially. Actually, this is the departure point of Soft Computing that tries to overcome the main difficulties in real problems, with the thesis that precision and certainty are sometimes unapproachable, and thus it may include the tolerance for imprecision and uncertainty [12,13]. Therefore, Soft Computingcan be defined as the antithesis of what we have called Hard Computing. We refer the reader to [10,14] for a more detailed definition of Soft Computing.

Although several classification of *Soft Computing* techniques have been proposed in the literature [12,13], Figure 1 shows a consensus among all of them. Since the fuzzy boom at the beginning of 90's, many methodologies based on these techniques have been proposed in the literature [15,16]. Although *Soft Computing*is a term introduced by Zadeh in 1994 [17], previous work was done by the definition of fuzzy sets [18]. Fuzzy sets are the pioneer paradigm in *Soft Computing*.They have been included in many other *Soft Computing*methods to provide hybrid methods. Among these new methods we may highlight Neural Networks [19], Support Vector Machines [20], Fuzzy Logic [12], Metaheuristics [21] (including techniques such as Evolutionary Computation [22, 23] or Swarm Intelligence [24]), to name just a few. There are a large number of algorithms within the umbrella of *Soft Computing*. They are applied to different fields such as symbol and pattern representation to enrich knowledge representation, machine learning for flexible knowledge acquisition, and inference by flexible knowledge processing. Moreover, *Soft Computing*techniques can be offered as a tool to interact with or they can be integrated in a larger framework where they provide unified and hybrid architectures.*Soft Computing* has been successfully applied to solve problems within the field of bioinformatics [25-27]. However, the large data sets generated from biological experiments and new high-throughput technologies make mandatory that modern *Soft Computing* approaches will be scalable across large-scale problems. In Section 3, we briefly introduce the *Soft Computing*techniques that have been applied to the protein folding problem.With that in mind, this paper focuses on the functional approximation or randomized search part of *Soft Computing*(see Figure 1) as it is gaining popularity during the last few years.

2.3 HPC platforms and programming models

In what follows, we reprise and update our vision of the High Performance Computing (HPC) arena, which was first given in [28]. HPC techniques and platforms are being applied for addressing many scientific challenges that would be otherwise very difficult to solve. The number of calculation required for this kind of scientific applications requires large computing resources. Just to mention an example, Anton is a supercomputer specially designed to simulate protein movements that could aid the drug design process [29].

However, we are witnessing a revolution in this areaas the Moore's law that has driven the development of new microprocessors in the last years [30,31], which is based on the idea that the number of transistors in an microprocessor would be doubled every two years, is running up against the laws of physics [32,33]. While a new microprocessor technology come up into the market, the industry has taken the steady transition to heterogeneous computing systems [34], with heterogeneity representing systems where nodes combine traditional multicore architectures (CPUs) and accelerators (mostly represented by GPU computing movement [35]or Intel Xeon Phi cards [36]).Heterogeneity limits system growing as it cannot be performed in an incremental way anymore. In particular, concepts like energy consumption, programmability, scalability, data location, and reliability become challenges for tomorrow's cyberinfrastructure [37]. This Section summarizes current trends in HPC platforms that are commonly used within the field of Bioinformatics. Of particular interest to us are,manycore architectures like Graphics Processing Units (GPUs), clusters of computers also known as

Supercomputers and cloud and distributed computing architectures.

2.3.1 GPU computing

Motivated by the computational demand of the videogame industry, Nvidia introduced in 2006 a graphics processing unit (GPU), codenamed CUDA (Compute Unified Device Architecture), which made available the computational power of those novel computing architectures to the scientific community. Nowadays, they have become a compelling alternative to the traditional architectures as they deliver high rates of floating point performance and massively parallelism at a very low cost, and thus democratizing the high performance computing (HPC) arena [38, 39].This movement was termed “GPGPU” which stands for *General-Purpose computation on Graphics Processing Units*. The GPGPU has promoted the use of this novel and massively parallel architecture in a wide range of applications, particularly in Bioinformatics, where parallelism and arithmetic intensity are common denominators in almost every application (we refer the reader to GPU application catalog provided by Nvidia[40]).

Following this trend almost all microprocessor company(e.g. ATI/AMD, Intel, etc) have developed their own hardware alternatives designed specifically foraccelerating general purpose applications.Among them, we may highlight Tesla-based GPUs from Nvidia, Firestream is ATI/AMD alternative and finally the new Intel Xeon Phicoprocessor which is based on Many Integrated Core (MIC) architecture. Along with these hardware components, those companies have also provided new programming models to easily leverage the horsepower of these emergent technologies.The first programming model for GPGPU was CUDA [35] (Compute Unified Device Architecture) provided by Nvidia that is specifically developed for programming Nvidia’s GPUs. Nvidia has a wide scientific community behind CUDA, and it offers several educational and research communities to promote the development of scientific applications with CUDA on Nvidia's GPUs. ATI/AMDfirst offered a programming model called Stream Computing which is not supported anymore and Intel relies on vectorization instructions based on X86programming. In 2008 the Khronos Group developed an open standard for parallel programming on cross-platform heterogeneous systems, called OpenCL[41]. OpenCL is an attempt to provide a standard programming language that allows multiplatform development on different devices like GPUs, accelerators, multicore systems, etc.

All of those novel programming models provide an easier way to leverage massively parallel architectures. However, programmers still have to deal with a new programming paradigm, which is rather different to the traditional sequential-basedarchitectures [42]. Moreover, those computing architectures are nowadays plugged into the motherboard through PCI Express bus. This fact provides heterogeneous computers that may have a traditional CPU and other computing devices like GPUs or accelerators. Each of these processors have their own memory spaces, different instruction set architectures and communication latencies. Therefore, programmability here is not an easy task.

Currently, the scientific community is looking for new programming models and tools that hide those inherently hardware particularities and provide an easier and faster way to develop application on this new landscape of computation. There are two different trends to provide such abstraction layer. First, the execution of a given program efficiently on different devices from a single source code [43,44]. Second, the API development to extent traditional programming languages like OMPSs for OpenMP [45],or OpenACCAPI [46], which establishes several directives to specify loops and regions of code in standard programming language such as FORTRAN, C++, C.

2.3.2 Supercomputers

High performance computer (also known as Supercomputers) are those computers that are developed to deal with great challenges within the industry and academia. Statistics on supercomputers are provided in the TOP500 list [34], where information about the number of systems installed, the performance of each system or their location among others is provided to manufactures and (potential) users. Supercomputers within TOP500 are highly involved in Bioinformatics research. For instanceTianhe-II and Titan, two top supercomputers in this list, are heavily involved in developing bioinformatics domain problems. Tianhe-II is addressing the needs of genetic engineering and biopharmaceutical simulations. Moreover, Titan is being used for molecular similarity to provide a description of membrane fusion. This is actually one of the main ways for molecules to enter or exit from living cells. Other leading examples are the supercomputer installed at the Leibniz Supercomputer Center in Monaco (SuperMUC) and the Piz Daint the CSCS/Swiss Bioinformatics Institute. The former supercomputer is commonly used for running bioinformatics applications like analysis of linkage disequilibrium in genotyping. The later has been successfully applied to run a challenge of evolutionary genomics based on calculating selection events in genes achieving several orders of acceleration.

Supercomputers are adopting the use of accelerators to speedup arithmetic intensive parts of the applications. Actually, five of the ten fastest supercomputers in Top 500 list [34] include accelerators in their designs. Those accelerators are basically limited to Intel Xeon Phi and Nvidia GPUs architectures. However, these accelerators increase the overall power consumption of the system which is actually a big issue, particularly for large-scale datacenters where Total Cost of Ownership is mainly influenced by the power supply [47]. Indeed, the inclusion of these accelerators can increase the power consumption of a cluster node up to 30%.

However, the total cost of ownership is not the only concern to reduce overall power consumption in supercomputers. Actually, this is now becoming mandatory as the carbon footprint of those systems is actually very high, and the reduction of carbon emissions is one of the main challenges in the last 2015 United Nations Climate Change Conference where the International Trade Union Confederation has called for the goal to be "zero carbon, zero poverty".For instance, the power consumption of TI supercomputers companies such as Google or Facebook, consumed about 0.5% of the overall power consumption in the world during 2005. If the cooling and power distribution were also taken into account then the power consumption increases up to 1% [48]. The high performance computing community is trying to develop supercomputers and infrastructures that reduce power consumption. Actually, the GREEN500 list [49]shows the 500 most power efficient supercomputers in the world. Indeed, we are envisioning a shift from the traditional metrics like FLOPS (FLoating point Operations Per Second) to FLOPS per watt.

Virtualization techniques are placed as the main way to reduce the overall power consumption in supercomputers, as they enable to have several virtual machines running at the same time in the same real hardware. Actually, datacenters are adopting this new trend for several applications.Of course, virtualization may have a performance impact. For instance, Amazon Elastic Cloud Computing EC2offers a virtual infrastructure of 26496 cores, achieving484,2TeraFLOPS for the High Performance Linpack benchmark, placing the cluster at position 101 in the November 2014 Top500 list but this is actually a tradeoff the scientific community has to deal with.

2.3.3 Cloud and distributed computing

As previously explained, the TCO of having an in-housesupercomputer is very high and it is not affordable for small institutions [50]. Cloud computing is ubiquitous and energy-efficient computer organization by its definition [51], in which virtualization is the main ingredient to obtain great energy reduction. In cloud computing platforms, services run remotely in a ubiquitous and distributed computing set of computers (a.k.a cloud) that may provide scalable and virtualized resources. In this way, heavy workloads can be migrated to other virtual nodes of the cloud, providing higher levels of hardware utilization [52]. Cloud providers offer their resources in a pay as you go fashion. Actually, it can be seen as an alternative to physical infrastructures but this is only useful for a specific amount of data and target execution time.

Cloud computing propose an on-demand scenario where users only pay for the computational time usersutilize for running their applications. There are several cloud computing models: infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS), and Data as a service (DaaS). Among them, IaaS is the most commonly used model while the other may provide other level of abstraction [53]. In the cloud, developers may use several instances and thus they can create a parallel cluster on demand. Like real hardware scenarios, those clusters can be programmed using libraries such as the Message Passing Interface (MPI). Those instances can be also used in a batchprocessing mode, launching several instances of a program and so on.

Cloud computing platforms are very interesting for bioinformatics practitioners mainly for the flexibility and the cost-effectiveness. Truth be told, this actually depends on the workloads they expect to run on the cloud but, in general, small-medium bioinformatics laboratories, which may perform bioinformatics analysis are moving to this technology as they avoid cost and issues of having an in-house computer infrastructure [54]. An alternative solution is represented by Hybrid Clouds that have both the scalability offered by cloud computing and the control and ad-hoc customizations supplied by in-house computers [55].

Those distributed solutions are evolving in the era of Big Data to frameworks like Hadoopthat allows distributed access to files. These frameworks are well suited for distributed algorithms such as MapReduce [56]. MapReduce is a programming environment to manage large data sets with a parallel, distributed algorithm on a cluster. For example, the PSIPRED [57] protein analysis workbench leverages the Hadoop implementation of MapReduce to launch several services to perform the execution of prediction methods in a large-scale system. Moreover, MapReduce has been also applied to provide an enhanced framework where parallel genetic algorithms target the protein folding in distributed environment [58].

Finally, some efforts have been done in the volunteer computing arena that is noteworthy to remark. Among them, we may highlight Folding@Home [59] which is a volunteer computing project that tries to solve the protein folding problem by means of collective human knowledge. Folding@Homehas been used in several medical researcheslike to cure Alzheimer's disease, Huntington's disease, and many forms of cancer, among other diseases. This project is pioneer in the use of many novel computing platforms such as Graphics Processing Units, CellBe processor, multi and many core systems through MPI and OpenMP language, as well as some smartphones for distributed computing and scientific research [60].

Kondow and Berlich [61] runs particle swarm optimization (PSO) on cloud for the simulation of proteins three-dimensional structure. They simulate all-atom force field using ArFlock library, aimed at finding the folded state of two proteins of different sizes starting from completely extended conformations.

2.3.4. Multiagent systems

Multiagent systems (MAS) can be also considered as a platform to tackle Bioinformatics problems such as protein folding. As defined in [62], they combine a flexible and high-level paradigm with a technology developed at the intersection between artificial intelligence and distributed computing. A typical MAS is composed of several autonomous entities –agents— that can communicate and interact among them in a competitive or cooperative manner. MAS are especially useful for simulation tasks, including the behavior of biological systems [63], where the different parts of the system have some individual features that distinguish it from the rest.

There are several works in the literature that have adopted MAS to address the protein folding problem. For example, a MAS using an independent energy model where every amino acid is identifying with an agent is presented in [64]. These amino-agents lay at the bottom level of the MAS architecture, their positions being coordinated by a set of cooperative agents in a higher level. Amino-agents movements are based on Monte Carlo-like criterion and hill-climbing strategy (to avoid local minimum). Coordination agents act as orchestra director suspending amino-agents movements when they are not improving a global strategy. These coordination agents offer the possibility of designing complex heuristics depending on external information and on the search history. Thus, external knowledge from databases can be injected to coordination agents to force amino-agents to make movements oriented to improve the energy results. Experimental tests performed in this MAS show that the proposed coordination level always introduces a better performance, but the energy function used is too coarse to provide good biological model.

Moreover, a MAS based on reinforcement learning for solving bidimensional protein folding is showed in [65]. In this case there are several basic agents trying to solve the problem using the Q-learning algorithm [66] based on their local knowledge and a reduced set of supervisor agents that synchronize and coordinate the basic agents according to the current best solution. Basic agents are distributed across multiple processes/machines and they use a blackboard to communicate with their supervisor agents. Authors claim that this distributed proposal greatly reduces the computational time employed in the training phase of the Q-learning algorithm with respect to a non-distributed approach. However, it must be further investigated how to preserve the accuracy of the results using a MAS. Finally, in [67] a competitive approach among agents is taken to implement an architecture named Discovery Bus aimed at modeling molecular design workflows. This MAS follows the quantitative structure–property relationships (QSPR) model to predict the properties of novel proteins.

An excellent discussion of pros and cons when using MAS in protein folding is given in [68]. The main advantage of this approach resides in its flexibility: addition and removal of agents could be done at run-time and therefore it is possible to change the structure of the experiment (e.g., the protein’s structure). In practice, not only may the limit conditions and the simulation constraints be changed dynamically, but also elements from the structure could be added and removed during the simulation. This fact augments the potentialities of simulated experiments, enabling a virtual manipulation of the system simulating the protein folding, even when this is not possible in reality. This property extends in silico experiments to in virtuo experiments, i.e., not only enabling the change of values of the parameters characterizing simulations, but also the structure of the experiment during run-time in an easy manner thank to MAS features. As for the main disadvantage of the use of MAS in this topic, it has been criticized that simulations performed by means of multiagent systems are not totally validated against real data, diminishing their credibility. Thus far, works in this area have focused on the reliability of MAS proposals from a qualitative point of view, showing that multiagent-based simulations are tantamount to other approaches. However, a quantitative validation must be performed to take MAS as a prominent alternative to protein folding.

3. Implementation of protein folding methods.

This section summarizes main contributions on the field of *Soft Computing*applied to the protein folding simulation. Particularly, we focus on the functional approximation or randomized search part of *Soft Computing*; i.e. Artificial Neural Networks and Metaheuristics, applied to the protein folding problem.

3.1. Artificial Neural Networks and SVM.

Artificial Neural Networks (ANNs) have been widely used in the protein folding field. Specifically, the most relevant types of ANNs are the feedforward neural networks [69] and recurrent neural networks [70]. ANN can learn tasks without needing much prior knowledge, and moreover they are tolerant to errors and noisy data. While the most common use of ANNs in protein folding has been devoted to detect secondary structures [71-73], they have been also employed in other tasks such as predicting the posttranslational modifications [74-76]; to identify disordered regions [77]; to predict metal binding sites [78,79]; to assign sub-cellular localization [80-82];classification of proteins into functional classes [83]; reconstructing protein structures [84] and protein class prediction [73,85], among others.

Regarding the prediction task, classifying secondary structure is an easy job for a neural network, as for example to learn to distinguish between alpha-helices and beta-strands models. This classification allows detecting the most three-dimensional structures as they are based on secondary ones. Although the alpha-helices and beta-strands is the main approach in the prediction task, there are some other papers that propose classifications among more than two classes [70,86,87]. Regarding the databases used by neural network, the most popular are the Protein Data Bank (PDB) [11] and the Structural Classification of Proteins dataset (SCOP) [88].

A major advance in the way in which the datasets are treated is to add sequences that are homogeneous to those that are being studying [89]. For example, given the same family of proteins, they share similar structural and functional features. For ANNs, this fact provides additional information in the inductive learning process that improves the task learning. This method is known as Evolutionary Information, however to find these homogeneous sequences is not trivial. For this research line, it is very popular the PSI-BLAST program [90].

Support Vector Machine (SVM) can be focused on the same field of work than ANNs for protein folding [86,88,91], although SVM presents a much better performance for regression against classification in protein folding recognition [92]. Furthermore, they have been used to estimate the significance of the sequence-template alignments [93] and protein secondary structure prediction [94].

It is worth mentioning that although neural networks have been widely used for protein folding, they have not been combined with high performance computing because the prediction of secondary structure do not imply a large computational complexity. However, new trends in neural networks such as Deep Learning [95] have called for reconsidering high performance in the field of neural networks due to its computational complexity. In this sense, Deep Learning has been proposed to make use of graphical processing units (GPUs) and CUDA parallel computing. Hence, Deep Learning has been used for sequence-based residue–residue contact prediction [96] and later for protein secondary structure prediction [97]. These proposals have been implemented using CUDAMat [98], a Python library that provides methods of fast matrix calculations on CUDA-enabled GPUs, providing high-level access to computing cores of graphics processing units.

3.2 Metaheuristics.

There are different approaches to classify metaheuristic algorithms in the literature. A good review of metaheuristic classification can be found in[99], depicted here inFigure 2. This classification takes into account five different features of such algorithms, namely their origins;the number of solutions used at the same time; the way the objective function is used; the neighborhood structure; and the use they make of the search history.

Depending on their origins, a new trend in designing metaheuristics concerns nature-inspired methods. These methods take as a source of inspiration biological or physical principles. Nature-inspired methods are very attractive for practitioners in high performance computing, as they are inherently parallel in definition (e.g.they may be inspired by a “swarm”-like schema that uses several agents to optimize a function). Ants, bees and fireflies are only some examples of populations that inspired algorithms based on their social behavior. Those algorithms rely on swarm to deal with complex problems [100,101]. Despite of this trend, in the last part of this sectionare introduced the most important non-nature inspired algorithms applied to the PSP problem, such as local search methods.

Regarding the number of solutions used at the same time, we can find algorithms working with a single solution or trajectory (e.g.,Tabu Search) or with the evolution of a set of solutions (e.g., Genetic algorithms). On the other hand, some metaheuristics define static objective functions that do not change during the algorithm execution (e.g., Genetic algorithms), whereas others may be modified during the search trying to escape from local minima (e.g., Guided Local Search).

Metaheuristics may be also classified depending on their neighborhood structure. The one-neighborhood structure does not change the fitness landscape topology during the execution, while in the various neighborhood search it is possible to expand the search among different fitness landscapes. Finally, the use of memory in the metaheuristic is another discriminative feature, separating into algorithms that take into account previous states to perform the next action orthose that use a Markov process to decide the next action only based upon the current state.

In this paper we have adopteda classification of metaheuristics based on origins as it is one of the most used and easy to understand.

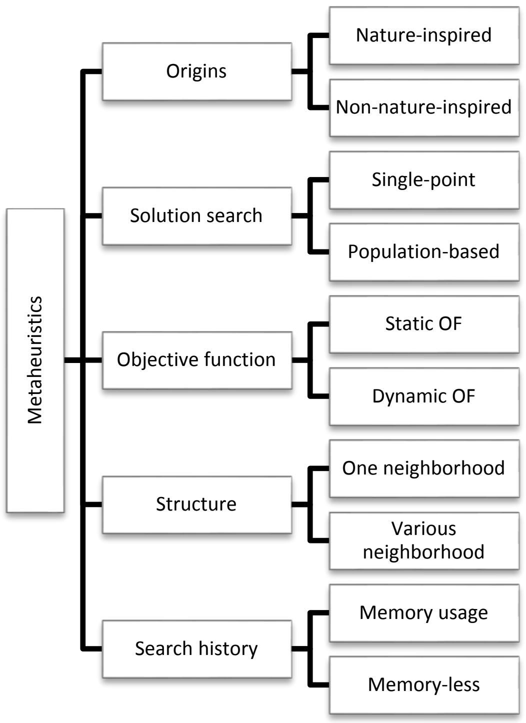

Figure 2Classification of metaheuristics techniques

Next sections review the main metaheuristics employed in protein folding.

3.2.1.Nature-inspired metaheuristics

Ant Colony Optimization

One nature-based method that is proving to be increasingly popular is *ant colony optimization* (ACO) [102,103].This algorithm is based on foraging behavior observed in colonies of real ants, and it has been applied to a wide variety of combinatorial problems [104, 105], including vehicle routing [106], feature selection [107] and protein function prediction [108]. The method generally uses simulated “ants” (i.e., mobile agents), which first construct tours or paths on a network structure (corresponding to solutions to a problem), and then deposit “pheromone” (i.e., signaling chemicals) according to the quality of the solution generated. The algorithm takes advantage of emergent properties of the multi-agent system, in that positive feedback (facilitated by pheromone deposition) quickly drives the population to high-quality solutions.

ACO algorithms have been extensively applied to the protein folding although most of them are based on the coarse-grain HP model. For instance, Shmygelska and Hoos [109] applied ACO to optimize the protein folding based on the HP model in both 2 and 3 dimensions. There are also other ACO-based implementations that have been applied to this problem in the literature. Song et al [110] provides a rapid transfer pheromone matrix method, a scheme to avoid deadlock folding problems, adynamic method of pheromone updating and also three different local search methods. This work uses the tortilla 3D benchmark [111] for the experimental evaluation.

Thalheim et al [112] combine the ACO with a branch and bound algorithm to enhance the protein folding simulation. For the experimental evaluation, they use proteins that are based on the bibliography and some of them come from PDB. Hu et al. [113] develop four different mechanismsto improve ACO algorithm, concretelyincludinga path retrieval method, the path construction, some folding heuristics and the pheromone attraction. These new mechanisms provide interesting results for solving protein folding problems with the HP square lattice model. Other hybrid approaches can be found at Chen et al. [114], where an ACO with genetic ideas was developed.

Some parallelization strategies have been applied to ACO solving the protein folding. [115] uses MPI to implement the parallel version of ACO.And in [116,117] OpenMPis used. It is noteworthy to highlight that only these few versions of parallelism have been implemented to solve the protein folding problem with ACO. From the High Performance Computing point of view, these parallel implementations use hardware clusters to evaluate their results.In [115]an IBM Blade center composed of 9 nodes, each node comprised of 2 2.4 Ghz Intel processors with 1 Gbyte of shared RAM is used. In [116]authors use a single PC to evaluate the sequential algorithm results, and an IBM pServer with eight 1.6GHz Power(gr) CPUs and 6GB RAM to run the parallel ones, which it seems not too fair. In [117], authors run the CASP8 benchmark on a multicore PC, specifically an IBM p550 server with an 8-core 64-bit 1.6-GHz PowerPC CPU, and the CASP9 benchmark is run on a cluster with 20 nodes of 16-core 1.6-GHz AMD CPU per node.

Although it has been demonstrated that this algorithm can take advantage of the GPU massively parallelism [118], to the best of our knowledge we could not find any work in this direction for the protein structure prediction using coarse-grain models.

ArtificialBeeColony

Artificial bee colony (ABC) algorithm is an optimization algorithm based on the behavior of honeybee swarms [119]. It provides a population-based search procedure in which the communication between bees is emulated to discover the best places with high nectar amount. Contrary to ACO, where only the HP model was targeted, ABC has been applied to different protein models such as HP, HP-SC, AB or ECEPP/3. There are several implementations of ABC applied to the protein folding problem. Zhang and Wu [120] use the HP-2D model to simulate the protein folding.However, authors use four Fibonacci sequences simulating proteins to test the algorithm instead of using a well-known benchmark like PDB or CASP.Another example of this algorithm can be found in [121], where synthetic sequences are created using Fibonacci sequences. Authors obtain experimental results with some PDB structures, though.

There are also parallel implementations of ABC that could be found in the literature. For example, in Benítez et al. [122-124], a complete study of different algorithm implementations can be found. Firstly, authors start implementing two parallel approximations of ABC algorithm in [122]: a master-slave implementation and a hybrid-hierarchical one, both of them implemented using ANSI C with MPI.They continue with the same two parallel approximations with genetic algorithm in [123],and finally authors conclude with the same parallel implementations of a hybrid algorithm merging an ABC with a Genetic algorithm (ABC-GA algorithm) in [124]. These authors remark that in future work they will consider the use of alternative computing technologies, such as reconfigurable computing and General-Purpose Graphics Processing Units, to accelerate processing. Nonetheless, no further papers in this sense have been found, at least applied to the protein folding problem with these algorithms. Finally, Bahamish et al. [125] develop a modified ABC that optimizes the Marriage in Honey Bee Optimization algorithm.

All the experimental environments in [120], [121]and [125]are based on single or multicores PCs.Benítez et al. [12-124] run their implementations on a 124 processing cores cluster.

Other papers considered in this area are Wang et al [126], where the Chaotic Artificial Bee Colony (CABC) algorithm, which combines the ABC algorithm with the chaotic search algorithm, is applied to 3D protein structure prediction; Li et al [127], where a balance-evolution artificial bee colony (BE-ABC) is presented and an AB off-lattice model is adopted, testedby Fibonacci sequences and proteins from the PDB as well; and [128], whereanother version of ABC is presented. These papers do not include any kind of HPC environment, and all the experiments run on a single PC.

Particle swarm optimization.

The third kind of algorithm that is shown in this section is Particle Swarm Optimization (PSO).PSO is a stochastic population-based optimization technique that is based on the social behavior of fish schooling or bird flocking. Applied to the protein folding problem, in [129] the authors implement PSO with an algorithm to avoid local minimums named levy flight. Like other algorithms, a parallel approach is performed by authors in [130] implemented using MPI, which is the most common way to parallelize the algorithms reviewed in this field. None of these papers, neither Chen et al. [129] nor Hernández et al.[130,131],give details about the environment for running the experiments on. Solely in [130] authors say that experiments are implemented in a "*dual-core PC and a Cluster*".

Other PSO algorithmscan be found in Liu et al.[132] and Mansour et al.[133]. The latterhave also developed a genetic algorithm for protein structure prediction. Both papers adopt the HP model with no HPC environments.

Genetic Algorithms

Genetic algorithms have been very used to address a broad range of combinatorial optimization problems that are NP-complete [134,135]. Genetic algorithms start from an initial randomly generated population of individuals. Over this initial population different selection, recombination and mutation operators are applied in order to evolve toward better solutions. In each iteration (generation), a function evaluates each individual, namely fitness function. On the one hand, the selection operator removes those individuals with worse fitness from a probabilistic point of view. On the other hand, the recombination and mutation operators generate variations of the individuals in order to produce new individuals. [136].

One of the first proposals of evolutionary algorithms to the PSP problem was presented by Unger and Moult [137]. In this work, a genetic algorithm is applied as an extension of a traditional Monte Carlo method to include information exchange between a set of parallel simulations. This method proves to find better solutions in the bidimensional HP lattice model than the traditional Monte Carlo methods. Some years later, an improved version of the basic GA [138] was presented using a new crossover operator and a new search strategy to avoid the homogenization of the population. Since then, several works following this idea has been proposed using different operators and strategies [139-145].

Genetic algorithms constitute a good alternative in several optimization problems. Nevertheless, one of the disadvantages of the genetic algorithm in optimization problems is the slow convergence. Concretely, in problems like PSP, they can suffer from excessively slow convergence rate due to the high number of needed calculations.

In order to avoid such problem, there is an opportunity in the hybridization of evolutionary algorithms with other heuristics, machine learning techniques, etc. The hybrid genetic algorithms can improve the performance of the basic algorithm and the quality of the solutions. For instance, the algorithms proposed in [146-148] combine a GA with tabu search algorithm, showing better results for the PSP than a genetic algorithm alone. Other works have proposed GA combined with other techniques, like backtracking [149], hill-climbing [150] and simulated annealing [151] or Particle Swarm Optimization [130].

As a result, since the PSP problem presents a large and complex search space, algorithms that combine local search methods with GA show significant improvements. In this sense, the combination of GA and local search using domain-specific knowledge, i.e. memetic algorithms [152] can help to find better solutions. Memetic algorithms (MA) use the concept of meme. A meme can be defined as a unit of cultural evolution which is able to local refinements. Some works have explored this mechanism for the PSP, resulting in that MAs are robust for finding structures across a range of models and difficulty [153-159].

The described proposals define the PSP as a single-objective optimization problem. This approach gets good results when one of the objective should be optimized or when all the objectives are not in conflict among them. Nevertheless, if several objectives should be optimized, a better approach is to consider the objectives separately, i.e., as a multiobjective optimization problem (MOP). A common problem in MOPs is the fact that usually there is no solution able to optimize all objectives at the same time. Therefore, the idea of optimum should be redefined and it is searched a solution that satisfies all the objectives in an acceptable manner. Some of the best well-known multiobjective evolutionary algorithms (MOEAs) are PAES-II, NSGA-II and MOEA/D. [160].

In this sense, some works propose the formulation of the PSP problem as an MOP to be solved by an MOEA. For example, [160] considers the PSP problem as the problem of minimizing free Potential Energy (PE) and minimizing Solvent Accessible Surface area (SAS). Authors solve this MOP using a modified version of the popular NSGA-II. In a similar way, the work of Day et al. [161] proposes a multiobjectivization for the HP model which scores better results in most of the cases than using a single-objective. Another example of this approach is the work of Brasil et al. [162,163]. In this work a new MOEA based on tables, called MEAMT, is presented. MEAMT is able to use four objectives based on tables to solve the PSP problem. In MEAMT, each table stores a subset of solutions with the best found solutions according to one of the objectives. More recently, several works have been proposed following this line of research. Some examples can be found in [164-167].

A great deal of the GA's popularity lies in its parallel nature and the inherent efficiency of parallel processing. MOEAs are a clear example of this parallelization, since their different objectives can be processed in parallel in an easy way. Despite the parallelization of MOEAs has been studied in several real-world problems, less work has been done in the parallel multiobjective approaches to PSP.

One of the works in this field has been developed by Calvo et al. [168-171]. They propose different parallel MOEAs approaches to the PSP problem reducing the complexity of the problem by the minimization of the set of variables involved in the process. Authors use 14 processors to execute parallel algorithms. They show that, although the quality of the solutions is not significantly improved, the process requires less time and presents a better parallel efficiency.

Tantar et al. [172] also propose a solution for the PSP using multiobjective parallel hybrid GAs (Hill Climbing local search [173] and simulated annealing [174] combined with GA) using computational grid. They use the ParadisEO-CMW framework, which combine the PAradisEO framework and the Condor-MW middleware. ParadisEO [175] is an open source framework dedicated to distributed and parallel models and the design of a broad range of metaheuristics. The Condor3 system [176] provides mechanisms that support High Throughput Computing (HTC). The underlying support the experiments was GRID5000 (2500 processors, 2.5TB of cumulated memory and 100 TB of non-volatile storage capacity). The tests were addressed using the tryptophan-cage (Protein Data Bank ID 1L2Y) and α-cyclodextrin proteins. Their studies show that, although the multiobjective GA increases the complexity, it provides more accurate solutions.

A different approximation for the PSP is proposed by Benítez et al. [177]. They present a parallel GA using the 3DHP-Side Chain model. In their approach the parallelism is reached by the division of the load into several processors (slaves) that are coordinated by a master processor. While the slaves have to compute the individual's fitness function, the master is in charge of the initialization the population and performing the rest of the GA operators. Since there is not dataset for the used model, the proposal was tested with a benchmark of synthetic sequences. Authors show that, although the results obtained are not the optimal, they are the best results found for the 3DHP-SC model. Finally, authors show that parallel processing accelerates significantly the process, but they propose other hardware-based approaches in order to get a better performing.

Unfortunately, this technique can suffer from a bottleneck in the master processor. In order to avoid this problem, in [178] it is proposed a mesh NoC-based multicore architecture in which the single-master multi-slave design is partitioned in small islands where an island has slaves and a master processor. In order to avoid GA falling in local minimal within each island, authors define a GA which is able to migrate between the islands. The experiments are performed using 9 proteins from a benchmark of synthetic sequences for the lattice protein model. Results show an overall 310X speedup gain compared to the design of the single-master /slave.

Others works have proposed modified GAs in order to parallelize the problem. For example, Narayanan et al. [58] propose a simple GA in which the mutation and selection strategies are parallelized using the MapReduce [179] architecture. Authors pursue to obtain the optimal conformation of a protein using the two dimensional square HP model. The proposal is validated against benchmarks of synthetic sequences, showing that the convergence of the algorithm to the optimal is faster than the obtained with traditional techniques.

Another modified version of an evolutive algorithm inspired by the biological immune systems, namely the clonal selection algorithm (CSA), is presented in [180] for PSP on AB Off-Lattice model. Experiments are performed using sequences of Fibonacci for simulating the AB model. The interesting aspect of this work is that the algorithm is parallelized using the CUDA platform and GPUs. In fact, authors show that the speed can be improved effectively, but they do not measure the quality of obtained solutions. There are also other hybrid GAs with bioinspired algorithms like Scalabrin et al. [181], but no more discussion is necessary because this paper has been also considered in the bioinspired algorithms section.

To summarize, although more works should be done in this direction, in the last years the parallelization of MOEAs is getting more attention and several works are including it as their future works [182,183].

Other nature-inspired algorithms

Other bioinspired algorithms also worth mentioning are gathered in this section.Firstly, a **Firefly Algorithm** (FA) [184] has been tested in the protein folding problem. Firefly Algorithm is a new algorithm that is based on the flashing behaviors of firefly swarms. The main purpose of the flash of fireflies is to attract other fireflies. The FA’s assumptions consist in three basic rules: (1) sex of fireflies does not mind at all as all fireflies are unisex. Each firefly flashes in order to attract other fireflies regardless their sex; (2) the intensity of the flash is mainly due to attract a prey and to share food; (3) the more a firefly shines, the more attractive it is to others. Therefore, each firefly firstly moves toward a neighbor whom glow is brighter. In this paper, two dimension HP lattice model is tested in a single PC, a P4 IBM with 3.1 GHz processor and 2 GB of RAM.

Only one approach to GP-GPU implementation has been found for bioinspired algorithms. Scalabrin et al. [181] (same authors of [122-124])have implemented a new algorithm named **Population-Based Harmony Search**, (PBHS). The Harmony Search is inspired by the improvisation process of a musician searching for the best harmony. The solution is represented by a harmony and the method of improvisation guides the balance between deep search and wide exploration. The results of this paper show that the implementation in CPU could be better when few data are used, but the GP-GPU is clearly better when data grow. The hardware experimental environment in this paper is an Intel processor (Core2-Quad at 2.8 GHz) and a NVIDIA GeForce GTX280.

Another bioinspired algorithm is the one developed by Cai et al. [185], where authors proposea new algorithm inspired by the plant growth process called **Artificial Plant Optimization Algorithm** (APOA). Photosynthesis operator, phototropism operator and apical dominance operator are designed in this paper.Another version of this algorithm can be found in [186], where authors implement the gravitropism mechanism that is neglected in the standard version. In this paper, authors employ this phenomenon to enhance the performance. To test the efficiency, they apply this new variant to solve protein structure prediction problem, including short sequences, Fibonacci sequences and real protein sequences, showing effective simulation results. The authors of these papers also present another bioinspired algorithm in [187] called **Social Emotional Optimization Algorithm** (SEOA). It is a new swarm intelligent methodology by simulating the human social behaviors. In this algorithm, each individual represents one virtual person in the searching space, all of them trying to promote to a high society position by collaboration and competition. In this paper, it is applied to predict the structure of toy model proteins. To test the performance, short sequence, Fibonacci sequence and real protein sequences are selected to compare. Simulation results show that this approach is valid. Authors do not use HPC environments in any of these papers commented in this paragraph.

Several hybrid approximations have been implemented, as for example in Benitez et al. [122], discussed above. Other papers with this point of view areNemati et al. [108], showing an implementation that combines a hybrid genetic and ACO algorithm; and also in Lin and Su [188], where authors implement a hybrid genetic and PSO algorithm. Moreover, although several modifications in algorithms have been tested, no improvements in hardware environments are found, since in [108] authors run the algorithm in a 3.0 GHz CPU and 512 MB of RAM, and no specification was found about hardware in [188].

To summarize, these papers give us the idea that several implementations of different algorithms have been tested during last years. Perhaps the more common algorithms at this point are ACO and ABC, although some other algorithms with different implementations have been found, for example hybrids algorithms. On the other hand, too little parallel implementations have been developed for these algorithms, and the exploitation of High Performance Architectures is reduced to the executions of parallel implementations based on MPI and OPENMP. Other types of more intensive data parallelism, like GP-GPU implementations, are expected to be widely developed, but unfortunately, the implementation in[181] byScalabrin et al. has been the only one found in this direction.

It is worth mentioning other reviews on this area, such as [189,190], that show the same point of view of different algorithms applied to the protein folding problem, although none of them elaborate a review from the High Performance Computing view.

3.2.2. Non-nature-inspired metaheuristics

Non-nature-inspired algorithms are mainly based on local search methods. They are a family of metaheuristic algorithms aimed at solving NP-hard optimization problems. Applied to protein folding, they try to obtain the minimum energy structure in polynomial time from a set of candidate solutions sampled from the search space. The main idea is to start from a folded protein deemed as a potential solution and then modify it (i.e., move to a neighbor solution in the search space) trying to obtain a slight improvement in the energy structure. Local search methods possess the main advantage of rapid convergence to better quality solutions, if not optimal, when efficient neighborhood functions are employed. However, an optimal solution cannot be guaranteed since the candidate solutions are randomly selected and the optimal one could not be included nor reached from the selected ones. Another drawback to take into account is that these methods get locked in a local optimum very often and may revisit the same set of solutions repeatedly.

Among the local search methods for protein folding simulation, Tabu search[191] is the most frequently found in the literature. The basic feature of this method is the use of memory structures to save solutions already explored. Then, if a potential solution is explored again in a specific period of time, it is considered tabu (i.e., forbidden) and therefore it is not expanded in order to promote the exploration of new regions in the search space. Tabu search algorithms applied to protein folding are also based on this feature, and they differ in the moves definition and how to avoid local optima.

Apart from Tabu search, hill climbing[192] and simulated annealing (SA)[193,194] are other two local search algorithms applied to protein folding. Hill climbing consists in starting with a random solution and changing a single element of the protein structure iteratively and incrementally while each change produces a better solution, until no further improvements can be made. On the other hand, simulated annealing uses a probabilistic heuristic to change from one random solution to another random solution with the aim of moving to a state of lower energy, but it still possible to change to a worse solution, i.e., a state of higher energy (and in this manner avoid local optima). The probability to move from a state *s* to a state'*s* depends on the energy of each state and on a global dynamic variable called temperature (T), which is initiated to a high value. As usual, if s' is considered better than s, then the movement is performed. However, if s' is considered worse than s, it is still possible to make that movement depending on T. For higher values of T, the probability of making this "worse" movement is higher. As T decreases through iterations, this probability also decreases, simulating the annealing process in metal. In this manner, it is possible during the initial phase of the process to move towards less promising solutions so as to avoid local optima, but at the end of the process --when T has values next to 0-- the probability of selecting worse solutions is almost inexistent. It is worth mentioning that both algorithms are normally used in combination with genetic or stochastic algorithms, as an alternative to improve the efficiency in the latter.

In the next paragraphs we review some of the most relevant works on protein folding for each local search algorithm.

Tabu Search.

[195] describes a generic tabu search plus a set of new moves for named "pull moves", that modifies the basic Tabu search by moving one aminoacid a small distance and then pull the chain along, stopping as soon as possible. These moves are complete (all existing configurations can be reached from the initial one), reversible and local (displace as few vertices as possible). As a result, authors propose small adjustments to a given configuration in order to improve the effectiveness of Tabu search in protein folding for HP-2D models. [196] also addresses HP-2D models. Moves are defined as changes of single angles of consecutive positions in the vector representing the protein, whereas the tabu list consists of forbidden angle moves to avoid reverse moves in a specific number of iterations. Authors claim to find optimal conformations for all short sequences from 5 to 12 aminoacids.

[197] explores on HP energy models on 3D FCC lattices. The Tabu method is composed of a function to initialize the model in a randomized, structured manner; a fitness function to guide the search; and efficient data structures to avoid cycles.. Authors obtain the first foldings in the well-known "Harvard instances"[198], 10 different proteins on the cubic lattice. This work has been revisited in [199], where the tabu algorithm is combined with constraint programming. Results show to be promising and reliable for proteins consisting in less than 100 aminoacids. Eventually, all the previous results on HP energy models on 3D FCC lattices have been outperformed by the work in[200]. This paper defines a hydrophobic-core centric local search algorithm named SS-Tabu. Movements are defined as a coil spinning around a dynamic hydrophobic-core center (HCC) by means of a diagonal move to build the cores. In order to avoid local minima, two different techniques named random-walk (based on the pull moves defined in[195] and relay-restart are defined. Another appealing approach on 3D HP lattices is proposed in[201] where authors develop an hybrid search algorithm that combines an enhanced particle swarm algorithm with an enhanced tabu search algorithm. The former appends the operation of crossover (single-point and two-point crossover) whereas the latter adds the operation of mutation. The main idea resides in using the tabu search algorithm to "help" the swarm algorithm to avoid local minimum. This hybrid algorithm has been implemented by MATLAB R2009b under a Windows XP system and tested through Fibonacci sequences and some PDB real proteins. Results show that it is superior to other 3D HP algorithms up

to sequences no longer than 48 aminoacids. A different approach for obtaining minimum energy in oligopeptides is presented in[202]. Moves are based on the dihedral angles in the protein's skeleton and the cost function is the empirical energy function ECEPP/3. It is aimed at working in angle space while keeping bond length and bond angle values constant. The algorithm is parallelized by executing several moves simultaneously. Hence, it is created a partitioning of the set of possible movements on p subsets of approximately the same size, and every partition is evaluated in p different processors. In this manner every processor finds its best move, and the best between these is eventually selected. The main drawback in this approach is the extensive communication requirement among processors. It has been tested using the Met-enkephalinpentapetide, showing a real speed-up compared to related techniques due to the parallelization process. As a result, Tabu search is considered valid for conformational searches of peptides when an optimal combination of tabu parameter values can be found.

Xiaolong et al. proposes a tabu algorithm whose main feature is the generation of the initial solution for 3D AB off-lattice models [203]. Instead of using a random function, a better-informed method is defined by locating hydrophobic residues at the center of three-dimension space and locating hydrophilic residues surrounding hydrophobic ones. In[204] a similar heuristic for the initial solution is employed and a new one is defined for conformation updating in 2D AB off-lattice models. The conformation updating heuristic consists in picking out hydrophilic monomers squeezed among hydrophobic monomers and placing them in certain spots in 2D space to speed up the search for lower-energy states.

Hill Climbing.

Regarding hill climbing works in the protein folding area, we have found that this technique is usually combined with genetic algorithms to improve the results of the latter. Thus, in [205] a hybrid of hill-climbing and ERS-GA (genetic algorithm with elite-based reproduction strategy), named HHGA, is proposed for protein structure prediction on the HP-2D triangular lattice. Two hill climbing strategies are proposed: In the first one, the algorithm selects its neighbour residues from the current solution. These residues are generated as in mutation operations, i.e., randomly changing its direction. In the second one, the neighbour residues are generated following a method similar to the crossover operation. Hence, five neighbours are generated 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 neighbour will replace the current solution. A benchmark composed of eight HP-2D protein sequences up to 64 aminoacids is evaluated and compared to simple genetic algorithms [206] and tabu search [207], demonstrating that HHGA produces a similar outcome to the those algorithms, but at the cost of incrementing the running time. Another work adopting hill climbing along with a genetic algorithm can be found in [208], which relies on hill-climbing recombination and mutation to support the search process of the evolutive algorithm for HP proteins. Here, the crossover operation is dynamically performed, allowing offspring to be added in the population during the same generation in an asynchronous manner. In this model, the proposed mutation operator is problem-specific and it is applied in a steepest-ascent hill-climbing manner. Moreover, to avoid local optima, redundant individuals may be replaced with new genetic material thanks to an explicit diversification stage which is carried out periodically during the population evolution. Standard S1-S8 HP proteins are employed as a benchmark and they are evaluated by the hybrid model presented in the paper and compared to other three simpler models, namely a simple evolutionary algorithm, an evolutionary algorithm with diversification stage and an evolutionary algorithm with hill climbing but without diversification. Results show that using hill climbing to support evolutive algorithm is clearly beneficial with respect to other models neglecting its use and it could compete with other algorithms such as memetics. Another hybrid GA-hill climbing algorithm, this time to fold proteins from knowledge of the primary sequence and predictions of its secondary structure, can be found in[209]. Dihedral angles are used to represent the protein's structure augmented with a four-helix bundle to improve the folding simulation conditions. According to the obtained results, the inclusion of a hill climbing algorithm to execute local searches in the GA outperforms 20% and 50% the execution of the pure, original GA in [210]. In conclusion, it can be stated that hill climbing algorithms are not practical by their own in protein folding, but they are rather combined with genetic algorithms to improve the latter.

Simulated annealing.

Like hill climbing, SA is mainly adopted for improving other global search algorithms. For example, [211] introduces a protein folding simulation procedure on FCC lattice that employs a constraint satisfaction problem (CSP) solver to generate neighbourhood states for a simulated annealing-based local search method. This proposal has been evaluated using three basic proteins for tuning (namely, 4RXN, 1ENH, 4PTI) and then several proteins selected from PDB, with length varying from 54 to 74 aminoacids. Results show that the hybrid approach outperforms CSP alone both in accuracy and efficiency, and outperforms local search alone in accuracy but not in time.

Another approach consisting in a combination of Bayesian and SA functions is described in [212]. It uses Bayesian scoring functions to assemble native-like structures from fragments of unrelated protein structure with similar local sequences. The simulated annealing contributes to generate native-like structures for small helical proteins in a rapid manner. Finally, it is worth mentioning the approach in[213] based on a pure SA algorithm in 3D HP protein folding simulations aimed at experimentally determining upper bounds for the maximum depth of local minima of the underlying energy and for the stopping criterion. Tests on the well-known ten benchmarks

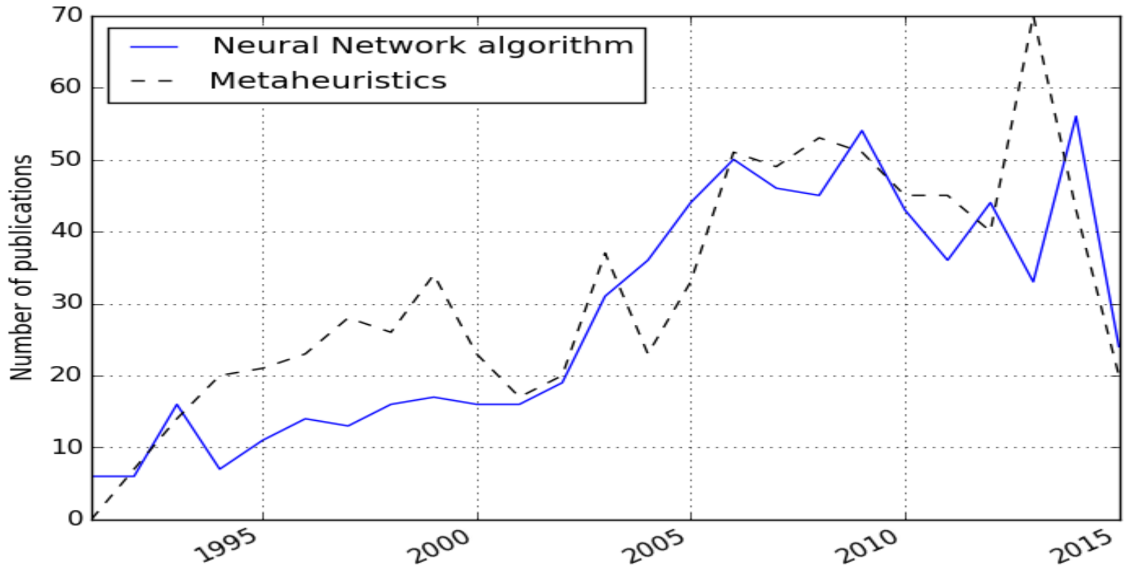

Figure 3 - Number of publications in protein folding, protein structure prediction or HP model

given by [214] show that the maximum escape height from local minima can be upper bounded by n^(⅔) whereas the stopping criterion complies with the number of Markov chain transitions that lead to minimum conformations. Further tests must be carried out on real foldings of short protein sequences to validate these results, which could serve as appropriate starting conformations for folding simulations of real protein sequences and realistic energy functions.

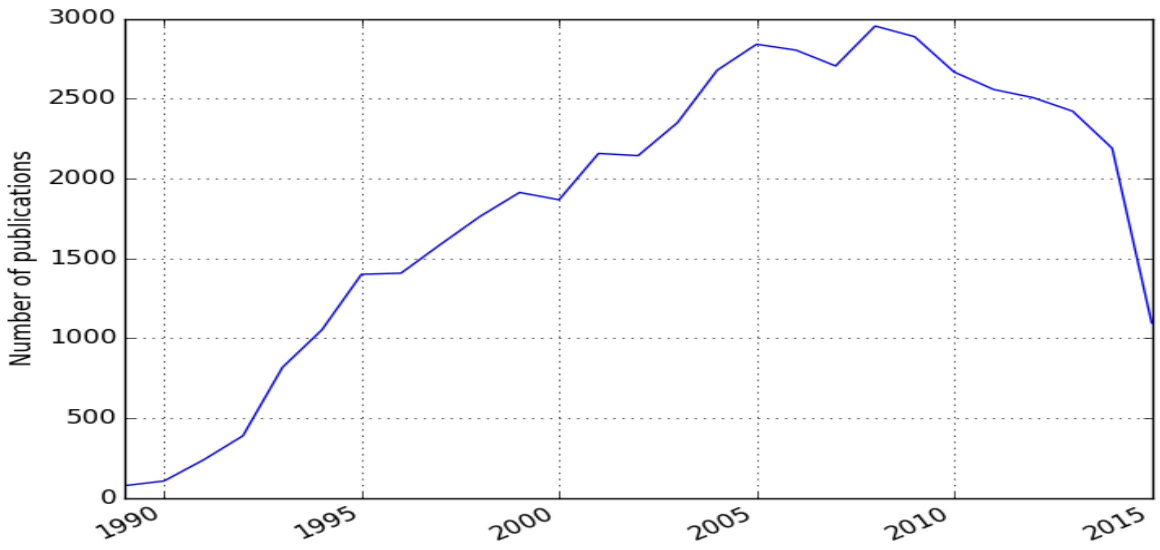

Figure 4 - Number of publications for Neural Networks and Metaheuristics techniques applied to protein folding

4. TRENDS IN DESIGNING NOVEL ALGORITHMS AND ARCHITECTURES

This section provides quantitative information about the main contributions in the field of Metaheuristics applied to PSP, mainly based on coarse-grain models. Moreover, we show what kind of hardware architectures have been used to run these novel algorithms on. Our deep search literature review follows a methodology that is firstly described to let the reader reproduce the experiments.

4.1. Experimental methodology

For this experimental study, we have used the Web of Knowledge (WOK, formerly known as ISI Web of Knowledge) [215]. WOK belongs to Thomson Reuters Corporation and it is an academic citation indexing and search service to provide bibliographic content and tools to access, analyze and manage multiple research information.

A particular interest to us is the WOK advanced search tool. This tool offers a very powerful search tool to look for different research articles using formal rules based on field tags, Boolean operators, parentheses, and query sets to create your own query. Booleans operators include AND, OR, NOR, SAME and NEAR. The following field tags are the most interesting for our searches purposes:

- TS = Topic. Searches the Topic fields in all databases in your institution subscription. Topic fields include Titles, Abstracts, Keywords and Indexing fields such as Systematics, Taxonomic Terms and Descriptors.
- SU = Research Area. Searches the Research Areas field within a Full Record.
- GP=Group Author. Searches the Group Author(s) and Book Group Author(s) fields within of a record.
- AU=Author. Searches for author names of journal articles and books in the Author(s) field and the Corporate Author(s) field.

The most interesting filed tag for our data mining purpose is TS as we are looking for articles related to protein folding, different Metaheurtistics techniques and particular hardware implementations. For instance, the following pattern searches for articles in which either “Protein folding” or “Protein Structure Prediction” or “HP model” are included in the article’s Title, Abstract or Keywords.

$$\text{TS} = (\text{"protein folding"} \\ \text{OR "protein structure prediction"} \\ \text{OR "HP model"})$$

However, the information obtained from this tool may have some inaccuracies as we are dealing with unstructured data. For instance, the terms “Neural Networks” and “Protein Folding” may be included in chemistry research articles about the brain, which clearly is not our scope. Therefore, after searching for some keywords we dida carefully review on ambiguous papers and checked whether they were related to the topics we are really looking for. Moreover, the WOK does not have very up-to-date information. Some recent papers are not included in their databases, and therefore, the quantitative information of the last couple of years may be incomplete. This issue mayaffect our conclusions regarding to the hardware trends as hardware platforms have evolved very rapidly in the last five years. As a result, we have also included articles from other databases such as Google Scholar, arXiv, CiteSeer(X), DBLP and IEEEXplore, to name just a few.

4.2. Trends in Soft Computing for the protein folding

In the first place, Figure 3shows the number of publications within the field of protein folding, protein structure prediction or coarse-grain HP model available in the WOK. The rule to perform this search is the following:

$$TS = ("proteinfolding" \\ OR\ "proteinstructureprediction" \\ OR\ "HPmodel")$$

From Figure 3 we can state that the Protein Structure Prediction is a very active field of research that began in eighties and it is still an object of continuous research with approximately 2.500 published papers per year.

Next, Figure 4shows the number of publications related to the protein folding that use *Soft Computing* techniques. Here we have grouped *Soft Computing* techniques into two different categories: Neural Networks and Metaheuristics. According to this figure, Neural Networks have been the most active research topic from the nineties. However, Metaheuristics has recently attracted interest in the protein folding community. In the last few years the number of articles published in Metaheuristics is at the top of the *Soft Computing* techniques applied to the protein folding. Local Search techniques, however, are almost always combined with other global techniques such as genetic algorithms, swarm intelligence or ant colony optimization to provide hybrid *Soft Computing* techniques. They improve the optimization process of those global search techniques to avoid stalling in local optimum, as noted in Section 3.2.2. The following rule is only an example of how we have obtained the number of publications for Neural Networks:

TS = ("protein folding" OR "protein structure prediction" OR "HP model")

$$and\ TS = ("Neural\ Network*"\ or\ "deep\ learning*"$$

$$or\ "support\ vector\ machine*"\ or\ "random\ forest*"$$

or "extreme learning machine" or "multilayer perceptron*")*

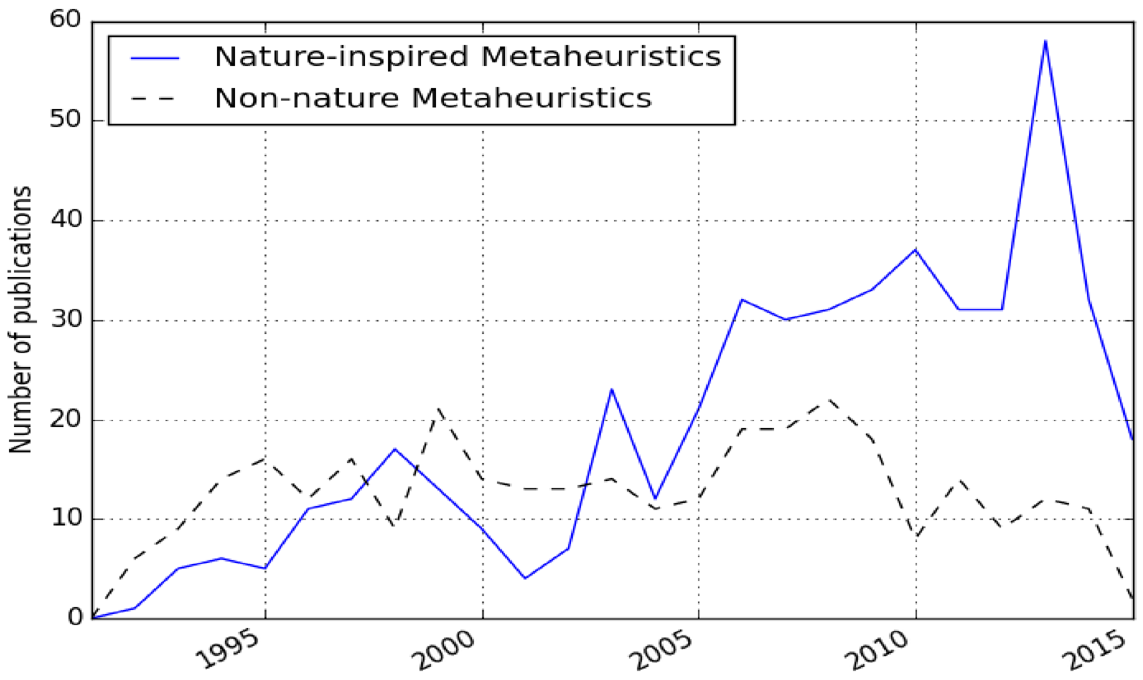

Figure 5 Number of publications that use Metaheuristics according to their origin.

Figure 5shows the number of publications in WOK related to both:the protein folding and different kind of Metaheuristics that area classified depending on their origins.The keywords used to do this search include for the nature-inspired metaheuristics:Genetic algorithm, Ant Colony Optimization, Artificial Bee Colony, Particle Swarm Optimization, Firefly Algorithm, Population-Based Harmony Search, memetic algorithm, Artificial Plant Optimization Algorithm and Social Emotional Optimization Algorithm. For non-nature-inspired metaheuristics the keywords are hill climbing, simulated annealing and tabu search. Some issues come up with this search as these keywords may belong to the same algorithmic family. For instance, ACO and ABC are population based methods which is also another keyword in Figure 5. Therefore, the number of publications depends on what keywords have been included in the article. Finally, those Metaheuristics that we could not find any work related to protein folding have not been included in Figure 5.

Figure 5places Genetic Algorithms are widely used in this area as they are one of the pioneer in Metaheuristic research. Particle Swarm and Ant Colony Optimization techniques are at the second place of the techniques used for protein structure prediction. Some variations of these Metaheuristics like memetics, firefly or Artifical Bee Colony are also applied in the literature but their use is marginal.

Figure 5shows the number of publication for different kind of non-nature metaheuristics that are mainly local search techniques. As previously described, local search techniques are used along with other global techniques to provide hybrid search method that improve simulation's quality and performance. The methods used in the protein folding arena are Tabu search, simulated annealing and hill climbing. The latter is widely used to improve the search provided by Metaheuristics. Although Tabu search is very close to hill climbing, the computational cost of tabu is higher than hill climbing, and thus it is not so convenient to integrate it in a hybrid method. Simulated Annealing is, however, a very powerful local search and it is actually the most studied in the literature.

4.3. Trends in hardware architectures for Soft Computing techniques applied to the protein folding

A common computational feature shared by many *Soft Computing* methods is their inherent massive parallelism. Most of them are population-based, that is, a collection of agents "collaborate" to find an optimal (or at least a satisfactory) solution. Because of this inherently parallel nature, these methods are well-suited to leverage parallel, distributed or even GPU architectures. Table 1summarizes the hardware platforms that have been used to improve the execution of different *Soft Computing* techniques.Neural Networks are basically executed on single core processors. Although there are some efforts in parallelizing neural networks applied to other problems, to the best of our knowledge there is only a work that cares about performancein this kind of algorithms applied to coarse-grain protein folding. Moreover, this algorithm is based on deep learning, which has many layers and thus the computational requirements increase drastically. Genetic algorithms are, however, very tied to parallel architectures. They are based on a population of entities where the island-model is very attractive to improve the solution.

In the parallel island-model of genetic programming, the population for a given run is divided into semi-isolated subpopulations. Each subpopulation is assigned to a separate processor or node of computing system and it proceeds independently to each other. Once each instance of the genetic algorithm finishes (or other interval), a relatively small percentage of the individuals in each subpopulation are probabilistically selected (based on fitness) for migration from each processor to various neighboring processors. This idea has been implemented on different platforms from clusters of computer nodes to grid computing environments. There are also other different parallel algorithms based on data approach that are better suited to GPUs. ACO, ABC and PSO also use the island model to leverage cluster computing architectures. Population Based Harmony Search has been implemented on GPUs as well. Finally, local search techniques have been also improve with some ways of parallelism in different architectures. Nonetheless, as previously mentioned, these methods are always combined to other methods, and therefore, they are also involved in other rows of the Table 1.

4.4. Summary

This section briefly summarizes the strengths and weaknesses of the reviewed algorithms grouped into main categories we have used throughout the paper. First of all, Artificial Neural Networks (ANNs) have been successfully applied to the protein's secondary structure prediction. The ANN computational cost of learning, applied to this problem, is affordable for sequential architectures, and thus it does not require the use of high performance computing.

Moreover, the ANNs offer an abstraction layer that provides solutions without having deep-knowledge of the problem domain that is very appreciated for non-domain experts within this area. However, we have only found few works using ANN that target more complex protein structure. This actually limits the successful of these techniques. Indeed, new trends in neural networks, such as deep learning, are demonstrating very good results in other domain fields [216]. They demand the use of high performance computing. The search for the ANN optimal architecture; i.e. the number of neurons within the hidden layer or even the number of layers, can be a very time consuming process.

This paper divides Metaheuristics for their origins into two main groups; nature and non-nature-inspired. Nature-inspired metaheuristics provide very good solutions in a reduced time-frame but they do not guarantee optimal solutions. Algorithms like ACO, ABC, PSO and so on, are based on swarm intelligence to solve problems. They are inherently parallel, and therefore, theoretically well-suited for parallelization on emergent architectures. This feature has been explored in few papers, but indeed, we still see many remaining work in this area.Moreover, genetic algorithms have the advantage that they could escape from suboptimal local maximum/minimum. They are population-based and they use stochastic operators that allow searching in different regions, thus if the population finds a better fitness value can move away from the suboptimal solutions. Genetic algorithms are also inherently parallel as population-based algorithms and therefore they are also well-suited for parallelization. Nevertheless, genetic algorithms also have some disadvantages whenever they target problems like protein folding. Sometimes genetic algorithms may converge very slowly, especially near an optimum. Some hybrid approximations have been presented for the protein folding problem in order to solve such problem. In that sense, genetic algorithms could suffer of the opposite problem and they can converge prematurely to the suboptimal solutions if the operators are not efficient enough. Finally, another disadvantage inherently associated to genetic algorithms is finding the algorithm parameters; it is not straightforward at all and very problem-dependent.

Non-nature-inspired metaheuristics, which in this paper are basically focused onlocal search techniques, provide appealing solutions for 2D/3D HP models. They can be easily combined with other global algorithms such as Genetic Algorithms or ACO to improve their solutions. They can quickly converge to better quality solutions, even optimal, when efficient neighborhood functions are employed and they could serve as appropriate starting conformations for folding simulations of real protein sequences and realistic energy functions.However, local search algorithms by themselves cannot guarantee an optimal solution. The candidate solutions are randomly selected and the optimal one could not be included nor reached from the selected ones. Also they may get locked in a local optimum very often and may revisit the same set of solutions repeatedly.

SoftComputing Technique	Algorithm	Hardware Platform	Data Set	Model	Ref
Neural Networks	Deep Learning	CUDA	D329, SVMCON_TEST and CASP9	HP 2D	[96,97]
	NNPIF (Neural Network Pairwise Interaction Fields)	Single core	PDB	HP 2D	[84]
	MLP (Multilayer Perceptron)	Single core	PDB, SCOP	HP 2D	[72,73]
	MLP + tailored early-stopping	Single core	PDB	HP 2D	[85]
	MLP + Evolutionary information	Single core	PDB	HP 2D	[71]
	SVM (Support Vector Machine)	Single core	SCOP	HP 2D	[86,91,94]
Genetic algorithms	Multiobjective GA	14 processors	1CRN protein	Atomic model based on the dihedrals angle base between the Cα	[168-171]
	Hybrid Multiobjective GA (Simulated Annealing and Hill Climbing)	ParadisEO-CMW framework. GRID5000	tryptophan-cage (Protein Data Bank ID 1L2Y) and α-cyclodextrin proteins	Atomic model based on the dihedrals angle base between the Cα	[173,174]
	Simple GA	MapReduce architecture (cluster)	Benchmarks of synthetic sequences	HP model	[179]
	Parallel GA (single-master multi-slave)	Master-slaves processors	Benchmarks of synthetic sequences	3DHP-Side Chain model	[177]
	Parallel GA (multi-master multi-slave)	Mesh NoC-based multicore architecture	Benchmarks of synthetic sequences	Lattice protein model	[178]
	Clonal selection algorithm (CSA)	GPUs and CUDA platform	Fibonacci based sequences	AB Off-Lattice model	[180]
ACO	Parallel ACO	Cluster	http://www.cs.sandia.gov/tech reports/compbio/tortilla-hp-benchmarks.html	HP 3D	[115]
	Parallel ACO	Single PC and Cluster	-	HP 2D	[116]
	Parallel ACO - packBackbone	CASP 8 Multicore PC. CASP 9 run on a Cluster.	CASP 8/9	HP 3D	[117]
ABC	Parallel ABC. MPI	Cluster Networked computers with 124 processing cores	Sequences from bibliography	HP 3D Side-Chain	[122]
	Modified ABC. IF-ABC	Multicore PC (Matlab)	Fibonacci based sequences. PDB sequences.	AB	[121]
	Modified ABC. MHBO	Multicore PC Visual C++	Met-enkphaline	Atomic model based on the dihedrals angle base between the	[125]

				Cα	
PSO	Parallel PSO	Multicore PC and a cluster	Sequences from bibliography	Atomic model based on the dihedrals angle base between the Cα	[130]
PBHS	Population Based Harmony Search	Multicore PC. NVIDIA GeForce GTX280.	Benchmarks of synthetic sequences.	AB 2D	[181]
Tabu Search	Pull moves similar to de Gennesreptation model.	HuGS middleware (Human-Guided Search)	Sequences from bibliography	HP-2D	[195]
	Protein’s angles-based moves	Single core PC AMD Duron 700Mhz Linux	Sequences from bibliography	HP-2D	[196]
	Tabu search + Constraint programming	A cluster of Dell Power Edge 1950 4-core IntelE5430 processor with 2.66GHz and 16Gb RAM (no parallelism)	Sequences from bibliography	HP 3D FCC lattice	[199]
	Spiral Search Tabu	-	Sequences from bibliography CASP 8/9	HP 3D FCC lattice	[200]
	Particle Swarm Optimizer + Tabu Search	MATLAB R2009b under a Windows XP system.	Fibonacci based sequences PDB proteins: IBXL, IEDP, IAGT	HP 3D FCC lattice	[201]
	Empirical energy function ECEPP/3	SGI Origin 2000 computers parallelized (32 processors) Distributed memory MPI for interprocessor communication	Met-enkephalinpentapetide	Atomic model based on the dihedrals angle base between the Cα	[202]
	Well-informed initial solution	-	Fibonacci based sequences (13, 21, 34) PDB (1BXL, 1EDP, 1AGT)	3D AB off-lattice	[203]
	Heuristic for conformation updating	Intel Core2 Duo, 2.66 GHz processor and 2.0 GB of RAM	Fibonacci based sequences (13, 21, 34,55) PDB (1AGT, 1AHO)	2D AB off-lattice	[204]
Hill Climbing	Montecarlo + hill climbing	Linda Tuple Spaces (Agents) Multithread C Two Opteron dual core CPU at 2 GHz	Several proteins from PDB	1. coarse grained structures based on previous bibliography 2. Own model	[64]
	Genetic algorithm + hill climbing	Single core Intel i7-920	Sequences from bibliography	2D HP Triangular lattice	[205]

		machines			
	Genetic algorithm + hill climbing	-	S1-S8 standard HP proteins	2D HP	[208]
	Genetic algorithm + hill climbing	SGI Onyx2 12 × R10000 supercomputer	Folding of the alpha carbon atoms of 100 non-redundant test proteins	Dihedral angles to augmented with a four-helix bundle	[209]
Simulated annealing	Time-dependent cooling schedule	Gentoo Linux on a 2.4 GHz Intel Pentium IV processor	Sequences from bibliography	HP 3D	[213]

Table 1 Summary of the hardware platforms used to improve different Soft Computing techniques.

CONCLUSIONS AND FUTURE WORK

The protein folding problem is a very well-known topic that has been widely studied during the last fifty years. Indeed, this review article showsthat the protein structure prediction problemis still a very active field of research nowadays, where many novel techniques and algorithms have been applied by means of computer simulation, mainly due to their high computational requirements. Our review focuses on both computational aspects:

1.-From the *algorithmic point of view*, we center on novel algorithms within the *Soft Computing* fieldthat have been applied mainly to the coarse-grain protein-folding problem, and focusing mostly on the HP-model.A particular interest to us are Neural Networks and Metaheuristics, as they are increasing in popularity during the last decade.The combination of these methods with local search techniques produces very powerful search strategies that providesome remarkable and interesting solutions to this problem. In this sense, and to the best of our knowledge, we have not found any work that design a hyper-heuristic or parametrized metaheuristic schema for the problem of the prediction of protein structure. These techniques provides a high-level of abstraction to look for the best metaheuristic to be applied to a concrete problem. Basically, metaheuristics search solutions within the problem domain and hyper-heuristics do the search within the search space of heuristics. Future designs should not only consider a metaheuristic, they should design a hyper-heuristic to provide a wide search within the space solution though. Besides, new trends in neural networks, such as deep learning, are gaining popularity, and we envision them as a good alternative for the protein structure prediction problem. However, fruitful works in this area should be designed taking care of computational requirements they intrinsically have by its definition, and thus they should designed on massively parallel architectures.

2.-From the *hardware point of view*, there are also some relevant contributions in the literature. Most of *Soft Computing* techniques are inspired bynature and they are massively parallel by their definition.Therefore they are well suited for implementation on parallel or even massively parallel architectures. After a deep literature review, we concludethat the gap between hardware and software in the simulation of protein folding is still very wide. There are some works that combine novel hardware and software techniques but they representjust an incipient research line. We are witnessing a revolution in hardware platforms where massive and heterogeneous platforms are dominating the marketsuch as GPUs.There are many applications already working right on the scientific and engineering fields. Changing them to run with billion-way parallelism will require redesigning or even reinventing the algorithms used in them, and potentially reformulating the science problems.

The protein folding simulation is a multidisciplinary field of research where scientists from different areas work together in order to solve challenges of the next century. Although many success cases have been reported in this review, there are still many aspects on the scientific side that need improvement. Just to name a few, the focus of application of these techniques relies on the study of single systems such as isolated proteins, but an “out of the box” approach should be followed in order to exploit them in more complex systems such as the ones in study by systems biology, as the cell as a whole. Also, techniques reviewed in this paper for the PSP problem might be directly applied to other biological macromolecules such as disordered proteins, nucleic acids, polymers, and systems with relevant nanotechnological interest. However, solving the problem of the prediction of protein structure, it is not an easy task. The workflow in Bioinformatics to create efficient tools is a long pipeline where each stage may take several years. Once theoretical models have been defined by experts from fundamental research fields such as physics, biology and chemistry, computer scientists need to define algorithms to simulate such models in computers. Moreover, as we move to a sustainable world, there are also other important concerns to take into accountas performance and energy efficiency of such algorithms on particular hardware architectures. Understanding how to bridging the gaps between hardware and software will be the key to solve mission-critical science problems at exascale.

From our point of view, future developments in this area should be aware of this landscape of computation.First of all, the physical limitations of silicon-based architectures are threatening the evolution of processors. Heterogeneous computing including GPUs, multiprocessors, or low-power processors come to the rescue when no answer looms on the horizon.Particularly, GPUs are showing great benefits in terms of performance and power consumption. The ratios compared with CPUs are expected toincrease even more as long as the problem size keeps growing and GPU microarchitectures take the next step forward. Moreover, the novel interest of governments in green computing makes mandatory developsscientific power-aware applications that use all hardware resources at minimum power-budget.

ACKNOWLEDGEMENTS

This work is jointly supported by the FundaciónSéneca (Agencia Regional de Ciencia y Tecnología, Región de Murcia) under grants 15290/PI/2010 and 18946/JLI/13, by the Spanish MEC and European Commission FEDER under grant with reference TEC2012-37945-C02-02 and TIN2012-31345, by the Nils Coordinated Mobility under grant 012-ABEL-CM-2014A, in part financed by the European Regional Development Fund (ERDF). We also thank NVIDIA for hardware donation within UCAM GPU educational and research centers.Finally, we thank anonymous reviewers for their constructive and valuable comments.

REFERENCES

[1] Dill KA, MacCallum JL. The protein-folding problem, 50 years on. Science 2012; 338(6110): 1042-6.

[2] Dill KA, Bromberg S, Yue K, *et al.* Principles of protein folding a perspective from simple exact models. Protein Sci 1995; 4(4): 561-602.

[3] Shaw DE, Dror RO, Salmon JK, *et al.* In:Millisecond-scale molecular dynamics simulations on Anton. Proceedings of the Conference on: High Performance Computing Networking, Storage and Analysis. Portland, IEEE 2009; pp 1-11.

[4] Merlitz H, Wenzel W. Comparison of stochastic optimization methods for receptor–ligand docking. Chem Phys Lett 2002; 362(3): 271-7.

[5] Baker D. A surprising simplicity to protein folding. Nature. 2000; 405(6782): 39-42.

[6] Berger B, Leighton T. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. J Comput Biol 1998; 5(1): 27-40.

[7] Fraenkel AS. Complexity of protein folding. Bull Math Biol 1993; 55(6): 1199-210.

[8] U.S. Department of Energy. Report on Top Ten Exascale Research Challenges.2014. Available at: http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf

[9] Asanovic K, Bodik R, Catanzaro BC, *et al*. The landscape of parallel computing research: A view from Berkeley. California; 2006. Report No.: UCB/EECS-2006-183.

[10] Li X, Ruan D, van der Wal AJ. Discussion on soft computing at FLINS'96. Int J Intell Syst 1998; 13(2-3): 28-300.

[11] Berman H, Westbrook J, Feng Z, *et al.* The protein data bank. Nucleic Acids Res 2000; 28(1): 235-42.

[12] Zadeh LA. Soft computing and fuzzy logic. IEEE Software 1994; 11(6): 48-56.

[13] Verdegay JL., Yager RR, Bonissone PP. On heuristics as a fundamental constituent of soft computing. Fuzzy Set Syst 2008; 159(7): 846-55.

[14] Bonissone PP. Soft computing: the convergence of emerging reasoning technologies. Soft Comput 1997; 1(1): 6-18.

[15] Bonissone PP. In:Hybrid Soft Computing for Classification and Prediction Applications. Proceedings of the First International Conference on Computing in an Imperfect World. London: Springer-Verlag 2002; pp 352-3.

[16] Mitra S, Pal SK, Mitra P. Data mining in soft computing framework: a survey. IEEE T Neural Networ 2002; 13(1): 3-14.

[17] Zadeh LA. Fuzzy logic, neural networks, and soft computing. Commun ACM 1994; 37(3): 77-84.

[18] Zadeh LA. Fuzzy sets. Inform Control 1965; 8(3): 338-53.

[19] Haykin S, Network N. A comprehensive foundation. Neural Networks 2004; 2(2004).

[20] Hearst MA, Dumais ST, Osman E, Platt J, Scholkopf B. Support vector machines. IEEE Intell Syst App 1998; 13(4): 18-28.

[21] Glover F, Kochenberger GA. Handbook of metaheuristics. Springer Science & Business Media 2003.

[22] Back T, Fogel DB, Michalewicz Z. Handbook of Evolutionary Computation Bristol, UK: IOP Publishing Ltd. 1997.

[23] Davis L. Handbook of genetic algorithms, New York: Van Nostrand Reinhold 1991.

[24] Kennedy J, Kennedy JF, Eberhart RC. Swarm intelligence, Morgan Kaufmann 2001.

[25] Sánchez-Linares I, Pérez-Sánchez H, Cecilia JM, García JM. High-throughput parallel blind virtual screening using BINDSURF. Bioinformatics. 2012; 13.

[26] Jena RK, Aqel MM, Srivastava P, Mahanti PK. Soft computing methodologies in bioinformatics. Eur J Sci Res 2009; 26(2): 189-203.

[27] Mitra S, Hayashi Y. Bioinformatics with soft computing. IEEE T Syst Man Cy C 2006; 36(5): 616-35.

[28] Peréz-Sánchez, H, Cecilia, J M, Merelli, I. In:The role of High Performance Computing in Bioinformatics. Proceedings of International Work-Conference on Bioinformatics and Biomedical Engineering. Granada, Spain: 2014; pp 494-506.

[29] Shaw DE, Maragakis P, Lindorff-Larsen, K, *et al.* Atomic-level characterization of the structural dynamics of proteins. Science 2010; 330(6002): 341-6.

[30] Schaller RR. Moore's law: past, present and future. IEEE Spectr 1997; 34(6): 52-9.

[31] Schulz M. The end of the road for silicon? Nature 1999; 399(6738): 729-30.

[32] Pavlus J. The Search for a New Machine. Sci Am 2015; 312(5): 58-63.

[33] Huang A. Moore's Law is Dying (and that could be good). IEEE Spectr 2015; 52(4): 43-7.

[34] Top500 The List. Available at: http://www.top500.org/ [accesed June 24,2015].

[35] Kirk DB, Wen-mei WH. Programming massively parallel processors: a hands-on approach, MA, USA: Elsevier 2013.

[36] Jeffers J, Reinders J. Intel Xeon Phi coprocessor high-performance programming, MA, USA: Elsevier 2013.

[37] Carretero J, García-Blas J, Singh DE, *et al.* In:Optimizations to enhance sustainability of MPI applications. Proceedings of the 21st European MPI Users' Group Meeting. Kyoto, Japan: ACM New York 2014; pp 145.

[38] Garland M, Le Grand S, Nickolls J, *et al.* Parallel computing experiences with CUDA. IEEE micro 2008; (4): 13-27.

[39] Nickolls J, Buck I, Garland M, Skadron K. Scalable parallel programming with CUDA. ACM Queue 2008; 6(2): 40-53.

[40] nVIDIA. GPU Accelerated. Available at: http://www.nvidia.com/content/gpu-applications/PDF/GPU-apps-catalog-mar2015.pdf [accessed May 30,2015].

[41] Tsuchiyama R, Nakamura T, Iizuka T, Asahara A, Miki S, Tagawa S. The OpenCL programming book. Fixstars Corporation 2010.

[42] Garland M, Kirk DB. Understanding throughput-oriented architectures. Commun ACM 2010; 58-66.

[43] Kim HS, El Hajj I, Stratton J, Lumetta S, Hwu WM. In:Locality-centric thread scheduling for bulk-synchronous programming models on CPU architectures. Proceedings of the 13th Annual IEEE/ACM International Symposium on Code Generation and Optimization. San Francisco, CA, USA: IEEE Computer Society Washington 2015; pp 257-68.

[44] Chang LW, Dakkak A, Rodrigues CI, Hwu WM. In: Tangram: a High-level Language for Performance Portable Code Synthesis. Proceedings of Programmability Issues for Heterogeneous Multicores. Amsterdam, Netherlands, 2015.

[45] The OpenMP API specification for parallel programming. Available at: http://openmp.org/wp/ [accessed May 30,2015].

[46] OpenACC, Directives for accelerators. Available at: http://www.openacc-standard.org/ [accessed May 30, 2015].

[47] Fan X, Weber WD, Barroso LA. In: Power provisioning for a warehouse-sized computer. Proceedings of the 34th annual international symposium on Computer architecture. San Diego, CA, USA: ACM New York 2007; pp 13-23.

[48] Koomey JG. Worldwide electricity used in data centers. Environ Res 2008; 3(3).

[49] The Green 500 List. Available at: http://www.green500.org/ [accessed June 24, 2015].

[50] Guerrero GD, Wallace RM, Vázquez Poletti JL, *et al*. A performance/cost model for a CUDA drug discovery application on physical and public cloud infrastructures. Concurrency-Pract Ex 2014; 26(10): 1787-98.

[51] Hewwit C. ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing/Carl Hewitt. IEEE Internet Comput 2008; 12(5): 96-9.

[52] Berl A, Gelenbe E, Di Girolamo M, *et al.* Energy-efficient cloud computing. Comput J 2010; 53(7): 1045-51.

[53] Armbrust M, Fox A, Griffith R, *et al.* A view of cloud computing. Commun ACM 2010; 53(4): 50-8.

[54] D'Agostino D, Clematis A, Quarati A, *et al.* Cloud infrastructures for in silico drug discovery: economic and practical aspects. Biomed Res Int 2013; 19.

[55] D'Agostino D, Galizia A, Clematis A, Mangini M, Porro I, Quarati A. A QoS-aware broker for hybrid clouds. Computing 2013; 95(1): 89-109.

[56] Shvachko K, Kuang H, Radia S, Chansler R. In: The Hadoop distributed file system. Proceedings of the 26th Symposium on Mass Storage Systems and Technologies (MSST). Incline Village, NV: IEEE 2013; pp 89-109.

[57] Buchan DW, Minneci F, Nugent TC, Bryson K, Jones DT. Scalable web services for the PSIPRED Protein Analysis Workbench. Nucleic Acids Res 2013; 41(1): 349-57.

[58] Narayanan AH, Krishnakumar U, Judy MV. In: An Enhanced MapReduce Framework for Solving Protein Folding Problem Using a Parallel Genetic Algorithm. Proceedings of the 48th Annual Convention of Computer Society of India. Springer International Publishing 2014; pp 241-50.

[59] Beberg AL, Ensign DL, Jayachandran G, Khaliq S, Pande VS. In: Folding@ home: Lessons from eight years of volunteer distributed computing. Proceedings of Parallel & Distributed Processing. Rome: IEEE 2009; pp 1-8.

[60] Folding@Home. Available at: http://folding.stanford.edu/home/the-science [accessed June 15, 2015].

[61] Kondov I, Berlich R. In: Protein structure prediction using particle swarm optimization and a distributed parallel approach. Proceedings of the 3rd workshop on Biologically inspired algorithms for distributed systems. Karlsruhe, Germany: ACM New York 2011; pp 35-42.

[62] Wooldridge M. An Introduction to Multiagent Systems Chichester, UK: John Wiley and Sons 2002.

[63] Cannata N, Corradini F, Merelli E, Omicini A, Ricci A. In: An agent-oriented conceptual framework for systems biology. Proceedings of Transactions on computational systems biology III. Berlin, Springer 2005 pp. 105-22.

[64] Bortolussi L, Dovier A, Fogolari F. Agent-based protein structure prediction. Multiagent and Grid Systems 2007; 3(2): 183-97.

[65] Czibula G, Bocicor MI, Czibula IG. Solving the protein folding problem using a distributed q-learning approach. Int J Comput Inf Sci 2011;(5): 404-13.

[66] Czibula G, Bocicor M, Czibula I. A reinforcement learning model for solving the folding problem. Int J Comput Appl T 2011; 2: 171–82.

[67] Cartmell J, Enoch S, Krstajic D, Leahy DE. Automated QSPR through competitive workflow. J Comput Aid Mol Des 2005; 19(11): 821-33.

[68] Amigoni F, Schiaffonati V. In: Multiagent-based simulation in biology. Proceeding of the Model-Based Reasoning in Science, Technology, and Medicine. Berlin, Springer 2007; pp 179-91.

[69] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural networks 1989; 2(5): 359-66.

[70] Pollastri G, Przybylski D, Rost B, Baldi P. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins 2002; 47: 228–35.

[71] Rost B, Sander C. Combining evolutionary information and neural networks to predict protein secondary structure. Proteins 1994; 19(1): 55-72.

[72] Chandonia JM, Karplus M. Neural networks for secondary structure and structural class predictions. Protein Sci 1995; 4(2): 275-85.

[73] Chandonia JM, Karplus M. The importance of larger data sets for protein secondary structure prediction with neural networks. Protein Sci 1996; 5(4): 768-74.

[74] Blom N, Hansen J, Blaas D, Brunak S. Cleavage site analysis in picornaviralpolyproteins: discovering cellular targets by neural networks. Protein Sci 1996; 5(11): 2203-16.

[75] Nielsen H, Engelbrecht J, Brunak S, von Heijne G. A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. Int J Neural Syst 1997; 8: 581-99.

[76] Nielsen H, Brunak S, von Heijne G. Machine learning approaches for the prediction of signal peptides and other protein sorting signals. Protein Eng 1999; 12(1): 3-9.

[77] Li X, Romero P, Rani M, Dunker A, Obradovic Z. Predicting protein disorder for N-, C-and internal regions. Genome Inform 1999; 10: 30-40.

[78] Sodhi JS, Bryson K, McGuffin LJ, Ward JJ, Wernisch L, Jones DT. Predicting metal-binding site residues in low-resolution structural models. J Mol Biol 2004; 342(1): 307-20.

[79] Passerini A, Punta M, Ceroni A, Rost B, Frasconi P. Identifying cysteines and histidines in transitionmetalbinding sites using support vector machines and neural networks. Proteins 2006; 65(2): 305-16.

[80] Nair R, Rost B. Better prediction of subcellular localization by combining evolutionary and structural information. Proteins 2003; 53(4): 917-30.

[81] Emanuelsson O, Nielsen H, Brunak S, von Heijne G. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. J Mol Biol 2000; 300(4): 1005-16.

[82] Reinhardt A, Hubbard T. Using neural networks for prediction of the subcellular location of proteins. Nucleic Acids Res 1998; 26(9): 2230-6.

[83] Jensen LJ, Gupta R, Blom N, *et al*. Prediction of human protein function from post-translational modifications and localization features. J Mol Biol 2002; 319(5): 1257-65.

[84] Mirabello C, Adelfio A, Pollastri G. Reconstructing Protein Structures by Neural Network Pairwise Interaction Fields and Iterative Decoy Set Construction. Biomolecules 2014; 4(1): 160-80.

[85] Igel C, Gebert J, Wiebringhaus T. In: Protein fold class prediction using neural networks with tailored early-stopping. Proceedings of Neural Networks. IEEE International Joint Conference 2004; pp 1693-7.

[86] Ding CH, Dubchak I. Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics 2001; 17(4): 349-58.

[87] Karchin R, Cline M, Mandel-Gutfreund Y, Karplus K. Hidden Markov models that use predicted local structure for fold recognition: alphabets of backbone geometry. Proteins 2003; 51(4): 504-14.

[88] Andreeva A, Howorth D, Brenner SE, Hubbard TJ, Chothia C, Murzin AG. SCOP database in 2004: refinements integrate structure and sequence family data. Nucleic Acids Res 2004; 32(1): 226-9.

[89] Rost B, Sander C. Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol 1993; 232(2): 584-99.

[90] Altschul SF, Madden TL, Schaffer AA, *et al*. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997; 25(17): 3389-402.

[91] Khan MA, Jan Z, Ali H, Mirza AM. In: Performance of Machine Learning Techniques in Protein Fold Recognition Problem. Proceedings of Information Science and Applications IEEE 2010; pp 1-6.

[92] Sangjo H, Byung-chul L, Seung TY, Chan-seok J, Soyoung L, Dongsup K. Fold recognition by combining profile–profile alignment and support vector machine. Bioinformatics 2005; 21(11): 2667-73.

[93] Xu J. Fold recognition by predicted alignment accuracy. IEEE ACM T Comput Bi 2005; 2(2): 157-65.

[94] Hua S, Sun Z. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. J Mol Biol 2001; 308(2): 397-407.

[95] Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks 2015; 61: 85-117.

[96] Eickholt J, Cheng J. Predicting protein residue–residue contacts using deep networks and boosting. Bioinformatics 2012; 28(23): 3066-72.

[97] Spencer M, Eickholt J, Cheng J. A Deep Learning Network Approach to ab initio Protein Secondary Structure Prediction. IEEE ACM T Comput Bi 2014: 103-12.

[98] Mnih V. Cudamat: a CUDA-based matrix class for python. Department of Computer Science, University of Toronto, Tech Rep UTML TR, 4: 2009.

[99] Blum C, Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput Surv 2003; 35(3): 268-308.

[100] Das S, Abraham A, Konar A. In: Swarm intelligence algorithms in bioinformatics. Proceedings of Computational Intelligence in Bioinformatics. Springer Berlin Heidelberg 2008; pp 113-47.

[101] Bergholt MS, Zheng W, Lin K, *et al*. In vivo diagnosis of gastric cancer using Raman endoscopy and ant colony optimization techniques. Int J Cancer 2011; 128(11): 2673-80.

[102] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Comput Intell Mag 2006; 1(4): 28-39.

[103] Blum C. Ant colony optimization: Introduction and recent trends. Phys Life Rev 2005; 2(4): 353-73.

[104] Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperation agents. IEEE Trans Syst Man Cybern 1996; 29-41.

[105] Dorigo M, Caro G, Gambardella L. Ant algorithms for discrete optimization. Artif Life 1999; 5(2): 137-72.

[106] Dorigo M, Stützle T. In: Handbook of Metaheuristics; Springer US: 2003; pp 250-85.

[107] Sivagaminathan RK, Ramakrishnan S. A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Syst Appl 2007; 33(1): 49-60.

[108] Nemati S, Basiri ME, Ghasem-Aghaee N, Aghdam MH. A novel ACO–GA hybrid algorithm for feature selection in protein function prediction. Expert Syst Appl 2009; 36(10): 12086-94.

[109] Shmygelska A, Hoos HH. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. BMC bioinformatics 2005; 6(1): 30.

[110] Song J, Cheng J, Zheng T. In: Protein 3D HP model folding simulation based on ACO. Proceedings of Intelligent Systems Design and Applications. IEEE 2006; pp 410-5.

[111] Tortilla HP benchmark. Available at: http://www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html [accessed Jun 15, 2015]

[112] Thalheim T, Merkle D, Middendorf M. Protein folding in the HP-model solved with a hybrid population based ACO algorithm. Int J Comp Sci 2008; 35(3): 291-300.

[113] Hu XM, Zhang J, Li Y. In: Flexible protein folding by ant colony optimization. Proceedings of Computational Intelligence in Biomedicine and Bioinformatics. Berlin: Springer 2008; pp 317-36.

[114] Chen C, Tian YX, Zou XY, Cai PX, Mo JY. A hybrid ant colony optimization for the prediction of protein secondary structure. Chinese Chem Lett 2005; 16(11): 1551-4.

[115] Chu D, Zomaya A. In: Parallel Ant Colony Optimization for 3D Protein Structure Prediction using the HP Lattice Model. Nedjah N, de MacedoMourelle L, Alba E. Springer Berlin Heidelberg 2006; pp 177-198.

[116] Guo H, Lu Q, Wu J, Huang X, Qian P. In: Solving 2D HP Protein Folding Problem by Parallel Ant Colonies. Proceedings of 2nd International Conference Biomedical Engineering and Informatics. Tianjin: IEEE. 2009; pp 1 - 5.

[117] Lv Q, Wu H, Wu J, Huang X, Luo X, Qian P. A parallel ant colonies approach to de novo prediction of protein backbone in CASP8/9. Sci China Inform Sci 2013; 56(10): 1-13.

[118] Cecilia JM, García JM, Nisbet A, Amos M, Ujaldón M. Enhancing data parallelism for ant colony optimization on GPUs. J Parallel Distr Com 2013; 73(1): 42-51.

[119] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 2008; 8(1): 687-97.

[120] Zhang Y, Wu L. Artificial bee colony for two dimensional protein folding. AEES 2012; 1(1): 19-23.

[121] Li B, Li Y, Gong L. Protein secondary structure optimization using an improved artificial bee colony algorithm based on AB off-lattice model. Eng Appl Artif Intel 2014; 27: 70-9.

[122] Benítez CMV, Lopes HS. In: Parallel artificial bee colony algorithm approaches for protein structure prediction using the 3DHP-SC model. Proceedings of Intelligent Distributed Computing IV. Springer Berlin Heidelberg, 2010; pp 255-64.

[123] Benitez CMV, Lopes HS. In: Hierarchical Parallel Genetic Algorithm applied to the three-dimensional HP Side-chain Protein Folding Problem. Proceedings of Intenrantional Conference on Systems Man and Cybernetics. Istanbul: IEEE 2010; pp 2669-76.

[124] Benitez CMV, Parpinelli RS, Lopes HS. Parallelism, hybridism and coevolution in a multi-level ABC-GA approach for the protein structure prediction problem. Concurr Comput 2012; 24(6): 635-46.

[125] Bahamish HAA, Abdullah R, Abu-Hashem MA. In: A modified Marriage in Honey Bee Optimisation (MBO) algorithm for protein structure prediction. Proceedings of 2nd International Conference on Computer Technology and Development. Cairo: IEEE 2010; pp 65-9.

[126] Wang Y, Guo GD, Chen LF. Chaotic Artificial Bee Colony algorithm: A new approach to the problem of minimization of energy of the 3D protein structure. Mol Biol+ 2013; 47(6): 894-900.

[127] Li B, Chiong R, Lin M. A balance-evolution artificial bee colony algorithm for protein structure optimization based on a three-dimensional AB off-lattice model. Comput Biol Chem 2015; 54: 1-12.

[128] Li Y, Zhou C, Zheng X. Artificial Bee Colony Algorithm for the Protein Structure Prediction Based on the Toy Model. Fundam Inform 2015; 136(3): 241-52.

[129] Chen X, Lv M, Zhao L, Zhang X. An Improved Particle Swarm Optimization for Protein Folding Prediction. IJIEEB 2011; 3(1): 1-8.

[130] Pérez-Hernández LG, Rodríguez-Vázquez K, Garduño-Juárez R. In: Parallel particle swarm optimization applied to the protein folding problem. Proceedings of the 11th Annual conference on Genetic and evolutionary computation. New York: ACM 2009; pp 1791-2.

[131] Pérez-Hernández LG, Rodríguez-Vázquez K, Gorduño-Juárez R. In: Estimation of 3D protein structure by means of parallel particle swarm optimization. Proceedings of Evolutionary Computation. Barcelona: IEEE 2010; pp 1-8.

[132] Liu J, Wang L, He L, Shi F. In: Analysis of toy model for protein folding based on particle swarm optimization algorithm. Proceedings of First International Conference. Changsha, China: Springer Berlin Heidelberg 2005; pp 636-45.

[133] Mansour N, Kanj F, Khachfe H. Particle swarm optimization approach for protein structure prediction in the 3D HP model. Interdiscip Sci 2012; 4(3): 190-200.

[134] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Addison-Wesley 1989.

[135] De Jong KA, Spears WM. In: Using genetic algorithms to solve NP-complete problems. Proceedings of International Conference on Genetic Algorithms. California: Morgan Kaufmann Publishers, 1989; pp 124-32.

[136] Holland JH. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. Oxford, England: U Michigan Press 1975.

[137] Unger R, Moult J. Genetic algorithms for protein folding simulations. J Mol Biol 1993; 231(1): 75-81.

[138] König R, Dandekar T. Improving genetic algorithms for protein folding simulations by systematic crossover. BioSystems 1999; 50(1): 17-25.

[139] Patton AL, Punch III WF, Goodman ED. In: A Standard GA Approach to Native Protein Conformation Prediction. Proceedings of International Conference of Genetic Algorithms; 1995. pp 574-81.

[140] Pedersen JT, Moult J. Protein folding simulations with genetic algorithms and a detailed molecular description. J Mol Biol 1997; 269(2): 240-59.

[141] Lopes HS, Scapin MP. In: An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model. Talbi EG, Liardet P, Collet P, Lutton E, Schoenauer M. Artificial Evolution: Springer Berlin Heidelberg 2006. pp 238-46.

[142] Hoque MT, Chetty M, Dooley LS. In: A new guided genetic algorithm for 2D hydrophobic-hydrophilic model to predict protein folding. Proceedings of Evolutionary Computation. Edinburgh, Scotland: IEEE 2005; pp 259-66.

[143] Song J, Cheng J, Zheng T, Mao J. In: A novel genetic algorithm for HP model protein folding. Proceedings of Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies. IEEE 2005; pp. 935-37.

[144] Sun S. Reduced representation model of protein structure prediction: statistical potential and genetic algorithms. Protein Sci 1993; 2(5): 762-85.

[145] Dandekar T, Argos P. Folding the main chain of small proteins with the genetic algorithm. J Mol Biol 1994; 236(6): 844-61.

[146] Zhang X, Wang T, Luo H, *et al.* In: 3D Protein structure prediction with genetic tabu search algorithm. Proceedings of The ISIBM International Joint Conferences on Bioinformatics, Systems Biology and Intelligent Computing (IJCBS). Shanghai, China: 2009;

[147] Jiang T, Cui Q, Shi G, Ma S. Protein folding simulations of the hydrophobic–hydrophilic model by combining tabu search with genetic algorithms. J Chem Phys 2003; 119(8): 4592-6.

[148] Rashid MA, Hoque MT, Newton MH, Pham DN, Sattar A. In: A New Genetic Algorithm for Simplified Protein Structure Prediction. Proceedings of 25th Australasian Joint Conference. Sydney, Australia: Springer Berlin Heidelberg 2012; pp 107-19.

[149] Cotta C. In: Protein structure prediction using evolutionary algorithms hybridized with backtracking. Proceedings Artificial Neural Nets Problem Solving Methods. Springer Berlin Heidelberg, 2003; pp 321-8.

[150] Chira C. Hill-Climbing search in evolutionary models for protein folding simulations. Stud Univ Babe\c s-Bolyai Inform 2010; 55: 29-40.

[151] Zhang X, Lin X, Wan C, Li T. In: Genetic-annealing algorithm for 3D off-lattice protein folding model. Proceedings of Emerging Technologies in Knowledge Discovery and Data Mining. Springer Berlin Heidelberg 2007; pp 186-93.

[152] Moscato P, Cotta C. In: A gentle introduction to memetic algorithms. Glove F, Kochenberger GA. Handbook of Metaheuristics: Springer US 2003. pp. 105-44.

[153] Islam MK, Chetty M. In: Novel memetic algorithm for protein structure prediction. Proceedings of Advances in Artificial Intelligence. Springer Berlin Heidelberg 2009; pp 412-21.

[154] Krasnogor N, Blackburne BP, Burke EK, Hirst JD. In: Multimeme Algorithms for Protein Structure Prediction. Proceedings of 7th International Conference of Parallel Problem Solving from Nature. Granada, Spain: Springer Berlin Heidelberg 2002; pp 769-78.

[155] Smith JE. In: The co-evolution of memetic algorithms for protein structure prediction. Hart WE, Smith JE, Krasnogor N. Recent Advances in Memetic Algorithms. Springer Berlin Heidelberg 2005. pp 105-28.

[156] Bazzoli A, Tettamanzi AG. In: A memetic algorithm for protein structure prediction in a 3D-lattice HP model. Proceedings of Applications of Evolutionary Computing. Springer Berlin Heidelberg 2004; pp 1-10.

[157] Islam MK, Chetty M. Clustered memetic algorithm with local heuristics for ab initio protein structure prediction. IEEE T Evolut Comput 2013; 17(4): 558-76.

[158] Islam MK, Chetty M, Murshed M. In: Novel local improvement techniques in clustered memetic algorithm for protein structure prediction. Proceedings of Evolutionary Computation. New Orleans, LA: IEEE 2011; pp 1003-11.

[159] Smith JE. In: Protein structure prediction with co-evolving memetic algorithms. Proceedings of the congress on Evolutionary Computation. IEEE 2003; pp 2346-53.

[160] Coello CAC, Van Veldhuizen DA, Lamont GB. Evolutionary algorithms for solving multi-objective problems New York: Kluwer Academic 2007.

[161] Day RO, Zydallis JB, Lamont GB, Pachter R. Solving the protein structure prediction problem through a multiobjective genetic algorithm. Nanotechnology 2002; 2: 32-5.

[162] Brasil CRS, Delbem ACB, da Silva FLB. Multiobjective evolutionary algorithm with many tables for purely ab initio protein structure prediction. J Comput Chem 2013; 34(20): 1719-34.

[163] SoaresBrasil CR, BotazzoDelbem AC, FerrazBonetti DR. In: Investigating relevant aspects of MOEAs for protein structures prediction. Proceedings of the 13th annual conference on Genetic and evolutionary computation. Dubin, Ireland: ACM 2011; pp 705-12.

[164] Cutello V, Narzisi G, Nicosia G. A multi-objective evolutionary approach to the protein structure prediction problem. J R Soc Interface 2006; 3(6): 139-51.

[165] Garza-Fabre M, Rodriguez-Tello E, Toscano-Pulido G. In: Multiobjectivizing the HP model for protein structure prediction. Proceedings of 12th European Conference Evolutionary Computation in Combinatorial Optimization. Málaga, Spain: Springer Berlin Heidelberg 2012; pp 182-93.

[166] Handl J, Lovell SC, Knowles J. In: Investigations into the effect of multiobjectivization in protein structure prediction. Proceedings of 10th International Conference on Parallel Problem Solving from Nature. Dortmund, Germany: Springer Berlin Heidelberg 2008; pp 702-11.

[167] Garza-Fabre M, Toscano-Pulido G, Rodriguez-Tello E. In: Locality-based multiobjectivization for the HP model of protein structure prediction. Proceedings of the 14th annual conference on Genetic and evolutionary computation. New York: ACM 2012; pp 473-80.

[168] Calvo JC, Ortega J. In: Parallel protein structure prediction by multiobjective optimization. Proceedings of 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing. Weimar, Germany: IEEE 2009; pp 268-75.

[169] Calvo JC, Ortega J, Anguita M, Urquiza JM, Florido JP. In: Protein structure prediction by evolutionary multi-objective optimization: search space reduction by using rotamers. Proceedings of Bio-Inspired Systems: Computational and Ambient Intelligence. Springer Berlin Heidelberg 2009; pp 861-8.

[170] Calvo JC, Ortega J, Anguita M. Comparison of parallel multi-objective approaches to protein structure prediction. J Supercomput 2011; 58(2): 253-60.

[171] Calvo JC, Ortega J, Anguita M. PITAGORAS-PSP: Including domain knowledge in a multi-objective approach for protein structure prediction. Neurocomputing 2011; 2675-82.

[172] Tantar A, Melab N, Talbi EG. In: A comparative study of parallel metaheuristics for protein structure prediction on the computational grid. Proceedings of Parallel and Distributed Processing Symposium. Long Beach, CA: IEEE 2007; pp 1-10.

[173] Tantar AA, Melab N, Talbi EG, Parent B, Horvath D. A parallel hybrid genetic algorithm for protein structure prediction on the computational grid. Future Gener Comp Sy 2007; 23(3): 398-409.

[174] Tantar AA, Melab N, Talbi EG. A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction. Soft Comput. 2008; 12(12): 1185-1198.

[175] Cahon S, Melab N, Talbi EG. ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. J Heuristics 2004; 10(3): 357-80.

[176] Thain D, Tannenbaum T, Livny M. Distributed computing in practice: The Condor experience. Concurr Comp-Pract E 2005; 17(2-4): 323-56.

[177] Benítez CMV, Lopes HS. Protein structure prediction with the 3D-HP side-chain model using a master–slave parallel genetic algorithm. J Braz Comp Soc 2010; 16(1): 69-78.

[178] Xue Y, Qian Z, Bogdan P, Ye F, Tsui CY. In: Disease Diagnosis-on-a-Chip: Large Scale Networks-on-Chip based Multicore Platform for Protein Folding Analysis. Proceedings of 51st ACM/EDAC/IEEE Design Automation Conference. San Francisco, CA: IEEE 2014; pp 1-6.

[179] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM 2008; 107-13.

[180] Zhu H, Xiao H, Gu J. In: Parallelism of Clonal Selection for PSP on CUDA. Proceedings of 3rd International Conference on Intelligent Networks and Intelligent Systems Shenyang: IEEE 2010; pp 467-70.

[181] Scalabrin MH, Parpinelli RS, Benítez CM, Lopes HS. Population–based harmony search using GPU applied to protein structure prediction. gbs-ijcse 2014; 9(1): 106-18.

[182] Mansour N, Kanj F, Khachfe H. Enhanced genetic algorithm for protein structure prediction based on the HP model: intech 2011;

[183] Garcia-Martinez JM, Garzón EM, Cecilia JM, Perez-Sanchez H, Ortigosa PM. An efficient approach for solving the HP Protein Folding Problem based on UEGO. J Math Chem 2015; 794-806.

[184] Zhang Y, Wu L, Wang S. Solving two-dimensional HP model by firefly algorithm and simplified energy function. Math Probl Eng 2013;

[185] Cai X, Wu X, Wang L, Kang Q, Wu Q. Hydrophobic-polar model structure prediction with binary-coded artificial plant optimization algorithm. J Comput Theor Nanos 2013; 10(6): 1550-4.

[186] Cui Z, Liu X, Liu D, Zeng J, Shi Z. Using Gravitropism Artificial Plant Optimization Algorithm to Solve Toy Model of Protein Folding. J Comput Theor Nanos 2013; 10(6): 1540-4.

[187] Cai X, Liu D, Wang L, Kang Q, Wu Q. Using Social Emotional Optimization Algorithm to Solve Toy Model of Protein Folding. J Comput Theor Nanos 2013; 10(6): 1545-9.

[188] Lin CJ, Su SC. Protein 3 D HP Model Folding Simulation Using a Hybrid of Genetic Algorithm and Particle Swarm Optimization. Int J Fuzzy Syst 2011; 13(2): 140-7.

[189] Zhao X. Advances on protein folding simulations based on the lattice HP models with natural computing. Appl Soft Comput 2008; 8(2): 1029-40.

[190] Karami Y, Khakzad H, Arab S, Fathy M, Shirazi H. In: Protein structure prediction using bio-inspired algorithm: A review. Proceedings of 16th CSI International Symposium on Artificial Intelligence and Signal Processing. Shiraz, Fars: IEEE 2012; pp 201-6.

[191] Glover F, Laguna M. Tabu search. In Du DZ, Pardalos PM. Handbook of Combinatorial Optimization. Springer US 1999. pp 2093-229.

[192] Russell S, Norvig P. A modern approach. Artificial Intelligence: Prentice-Hall 1995.

[193] Kirkpatrick S, Gelatt JCD, Vecchi MP. Optimization by simulated annealing. Science 1983; 220(4598): 671-80.

[194] Eglese RW. Simulated annealing: a tool for operational research. Eur J Oper Res 1990; 46(3): 271-81.

[195] Neal L, Mitzenmacher M, Whitesides S. In: A complete and effective move set for simplified protein folding. Proceedings of the seventh annual international conference on Research in computational molecular biology. New York: ACM 2003; pp 188-95.

[196] Błażewicz J, Łukasiak P, Miłostan M. Application of tabu search strategy for finding low energy structure of protein. Artif Intell Med 2005; 35(1): 135-45.

[197] Cebrián M, Dotú I, Van Hentenryck P, Clote P. In: Protein structure prediction on the face centered cubic lattice by local search. Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. 2008; pp 241-6.

[198] Yue K, Fiebig K, Thomas P, Chan H, Shakhinovich E, and Dill K. In: A test of lattice protein folding

algorithms. Proceedings of the National Academy of Sciences. 1995. pp 325-9.

[199] Dotu I, Cebrián M, Van Hentenryck P, Clote P. On lattice protein structure prediction revisited. IEEE/ACM Tans Comput Biol Bioinf 2011; 8(6): 1620-32.

[200] Rashid MA, Newton MAH, Hoque MT, Shatabda S, Pham D, Sattar A. Spiral search: a hydrophobic-core directed local search for simplified PSP on 3D FCC lattice. BMC Bioinformatics 2013; 14.

[201] Zhou C, Hou C, Zhang Q, Wei X. Enhanced hybrid search algorithm for protein structure prediction using the 3D-HP lattice model. J Mol Model 2013; 19(9): 3883-91.

[202] Morales LB, Garduño–Juárez R, Aguilar–Alvarado JM, Riveros–Castro FJ. A parallel tabu search for conformational energy optimization of oligopeptides. J Comput Chem 2000; 21(2): 147-56.

[203] Xiaolong Z, Cheng W. In: An improved tabu search algorithm for 3D protein folding problem. Proceedings of 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence. Hanoi, Vietnam: Springer Berlin Heidelberg 2008; pp 1104-9.

[204] Liu J, Sun Y, Li G, Song B, Huang W. Heuristic-based tabu search algorithm for folding two-dimensional AB off-lattice model proteins. Comp Biol Chem 2013; 47: 142-8.

[205] Su SC, Lin CJ, Ting CK. An effective hybrid of hill climbing and genetic algorithm for 2D triangular protein structure prediction. Proteome Sci 2011; 9: 19.

[206] Hoque MT, Chetty M, Dooley LS. In: A hybrid genetic algorithm for 2D FCC hydrophobic-hydrophilic lattice model to predict protein folding. Proceedings of 19th Australian Joint Conference on Artificial Intelligence. Hobart, Australia: Springer Berlin Heidelberg 2006; pp 867-76.

[207] Böckenhauer HJ, Ullah AZMD, Kapsokalivas L, Steinhöfel K. In: A local move set for protein folding in triangular lattice models. Proceedings of 8th International Workshop. Karlsruhe, Germany: Springer Berlin Heidelberg 2008; pp 369-81.

[208] Chira C, Horvath D, Dumitrescu D. Hill-Climbing search and diversification within an evolutionary approach to protein structure prediction. BioData Min 2011; 4(1): 23.

[209] Cooper LR, Corne DW, Crabbe MJC. Use of a novel Hill-climbing genetic algorithm in protein folding simulations. Comp Biol Chem 2003; 27(6): 575-80.

[210] Dandekar T, Argos P. Identifying the tertiary fold of small proteins with different topologies from sequence and secondary structure using the genetic algorithm and extended criteria specific for strand regions. J Mol Biol 1996; 256(3): 645-60.

[211] Ullah AD, Steinhöfel K. In: A hybrid approach to protein folding problem integrating constraint programming with local search. Proceedings of the Eighth Asia Pacific Bioinformatics Conference. Bangalore, India: LaxmiParida and Gene Myers 2010;

[212] Simons KT, Kooperberg C, Huang E, Baker D. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. J Mol Biol 1997; 268(1): 209-25.

[213] Albrecht AAM, Skaliotis A, Steinhöfel K. Stochastic protein folding simulation in the three-dimensional HP-model. Comp Biol Chem 2008; 32(4): 248-55.

[214] Beutler TC, Dill KA. A fast conformational search strategy for finding low energy structures of model proteins. Protein Sci 1996; 5(10): 2037-43.

[215] Web of Knowledge. Available at:www.webofknowledge.com [accessed Jun 25, 2015].

[216] Deng, L., Yu, D. Deep learning: methods and applications. Fond T Sign Proc 2014; 7(3–4): 197-387.