A BRIEF SURVEY ON BIO INSPIRED OPTIMIZATION ALGORITHMS FOR MOLECULAR DOCKING

Mayukh Mukhopadhyay
Tata Consultancy Services, Kolkata-700091, India

ABSTRACT
The idea in molecular docking is to design pharmaceuticals computationally by identifying potential drug candidates targeted against proteins. The candidates can be found using a docking algorithm that tries to identify the bound conformation of a small molecule ligand to a macromolecular target at its active site, which is the region of an enzyme where natural substrate binds. Mathematically, molecular docking can be formulated as an optimization problem in which the objective is to minimize the intermolecular bound conformational energy of two interacting molecules. In this brief survey, after a concise introduction on nature inspired computing, the application of bio inspired optimization algorithms in molecular docking has been studied.

KEYWORDS: *Bio-Inspired Algorithm, Differential Evolution, Protein-Ligand Docking Problem*

I. NATURE INSPIRED COMPUTING

1.1 Introduction

Nature Inspired Computing (NIC) is one that aims to develop new computing techniques after getting ideas by observing how nature behaves in various situations to solve complex problems. A typical NIC system is based on self-organization and complex systems. It is a computing system operated by population of autonomous entities surrounded by environment.

Autonomous entity of the NIC system consists of two devices: detectors and effectors [25]. There may be one or more detector that receives information related to its neighbours and its environment. Obviously the information depends on the system to be modeled or problem to be solved. There may be multiple effectors that make changes to the internal state, exhibits certain behaviors and make changes to the environment. Basically the effector facilitates sharing of information among autonomous entities. NIC system has a repository of local behavior rules. The rules of behavior are crucial to autonomous entity. They are used to decide how autonomous entity must act or react to the information collected by the detector from the environment and neighbors. Autonomous entity should be capable of learning. It must respond to local changing conditions by modifying its rules of behaviour over time.

Autonomous entities have ADEAS (Autonomous, Distributed, Emergent, Adaptive, Self-organized) characteristics [25]. Autonomous entities are independent and rational. They use formal computing methods to describe how entities acquire and improve reactive behavior. They have decision making capabilities and are distributed in the environment. They follow predefined protocols and interact with each other to exchange their state information. New complex behaviors emerge when the entities act collectively. The entities respond to changes in the environment by changing their behavior. By interacting with each other the entities self-organize to fine tune their behaviour.

Environment may be static or dynamic. In static environment, autonomous entities are free to roam. Dynamic environment acts as notice board allowing autonomous entities post and read local information. The central clock helps synchronize the actions of the autonomous entities.

By far the majority of nature-inspired algorithms are based on some successful characteristics of biological system [22]. Therefore, the largest fraction of nature-inspired algorithms is biology-inspired, or bio-inspired for short.

Among bio-inspired algorithms, a special class of algorithms has been developed by drawing inspiration from swarm intelligence. Therefore, some of the bio-inspired algorithms can be called swarm-intelligence based. In fact, algorithms based on swarm intelligence are among the most popular. Good examples are ant colony optimization, particle swarm optimization, cuckoo search, bat algorithm, and firefly algorithm.

Obviously, not all algorithms were based on biological systems. Many algorithms have been developed by using inspiration from physical and chemical systems. Some may even be based on music. In the next section, we will briefly divide all algorithms into different categories, and we do not claim that this categorization is unique. This is a good attempt to provide sufficiently detailed references.

1.2 Classification

We can divide all existing nature inspired algorithms into four major categories: swarm intelligence (SI) based, bio-inspired (but not SI-based), physics/chemistry-based, and others. We will summarize them briefly in the rest of this section. However, we will focus here on the relatively new algorithms.

Well established algorithms such as genetic algorithms are so well known that there is no need to introduce them in this brief paper. It is worth pointing out the classifications here are not unique as some algorithms can be classified into different categories at the same time.

Loosely speaking, classifications depend largely on what the focus or emphasis and the perspective may be. For example, if the focus and perspective are about the trajectory of the search path, algorithms can be classified as trajectory based and population-based. Simulated annealing is a good example of trajectory-based algorithms, while particle swarm optimization and firefly algorithms are population-based algorithms.

If our emphasis is placed on the interaction of the multiple agents, algorithms can be classified as attraction-based or non-attraction-based. Firefly algorithm (FA) is a good example of attraction based algorithms because FA uses the attraction of light and attractiveness of fireflies, while genetic algorithms are non-attraction-based since there is no explicit attraction used.

On the other hand, if the emphasis is placed on the updating equations, algorithms can be divided into rule-based and equation-based. For example, particle swarm optimization and cuckoo search are equation based algorithms because both use explicit updating equations, while genetic algorithms do not have explicit equations for crossover and mutation. However, in this case, the classifications are not unique. For example, firefly algorithm uses three explicit rules and these three rules can be converted explicitly into a single updating equation which is nonlinear.

This clearly shows that classifications depend on the actual perspective and motivations. Therefore, the classifications here are just one possible attempt, though the emphasis is placed on the sources of inspiration.

1.2.1 Swarm intelligence based

Swarm intelligence (SI) concerns the collective, emerging behaviour of multiple, interacting agents who follow some simple rules [22]. While each agent may be considered as unintelligent, the whole system of multiple agents may show some self-organization behaviour and thus can behave like some sort of collective intelligence. Many algorithms have been developed by drawing inspiration from swarm-intelligence systems in nature.

All SI-based algorithms use multi-agents, inspired by the collective behaviour of social insects, like ants, termites, bees, and wasps, as well as from other animal societies like flocks of birds or fish. A list of swarm intelligence algorithms is presented in Table 1.

The classical particle swarm optimization (PSO) uses the swarming behaviour of fish and birds, while firefly algorithm (FA) uses the flashing behaviour of swarming fireflies. Cuckoo search (CS) is based on the brooding parasitism of some cuckoo species, while bat algorithm uses the echolocation of foraging bats. Ant colony optimization uses the interaction of social insects (e.g., ants), while the class of bee algorithms are all based on the foraging behaviour of honey bees.

SI-based algorithms are among the most popular and widely used. There are many reasons for such popularity; one of the reasons is that SI-based algorithms usually sharing information among multiple agents, so that self-organization, co-evolution and learning during iterations may help to provide the high efficiency of most SI-based algorithms. Another reason is that multiple agent can be parallelized easily so that large-scale optimization becomes more practical from the implementation point of view.

1.2.2 Bio-inspired but not SI-based

Obviously, SI-based algorithms belong to a wider class of algorithms, called bio-inspired algorithms. In fact, bio-inspired algorithms form a majority of all nature-inspired algorithms. From the set theory point of view, SI-based algorithms are a subset of bio-inspired algorithms, while bio-inspired algorithms are a subset of nature-inspired algorithms [22]. That is

$$\text{SI-based} \subset \text{bio-inspired} \subset \text{nature-inspired} \quad (1)$$

Conversely, not all nature-inspired algorithms are bio-inspired, and some are purely physics and chemistry based algorithms as we will see below. Many bio-inspired algorithms do not use directly the swarming behaviour. Therefore, it is better to call them bio-inspired, but not SI-based. For example, genetic algorithms are bio-inspired, but not SI-based. However, it is not easy to classify certain algorithms such as differential evolution (DE). Strictly speaking, DE is not bio-inspired because there is no direct link to any biological behaviour. However, as it has some similarity to genetic algorithms and also has a key word 'evolution', we tentatively put it in the category of bio inspired algorithms. These relevant algorithms are listed in Table 1.

For example, the flower algorithm or flower pollination algorithm developed by Xin-She Yang in 2012 is a bio-inspired algorithm, but it is not a SI-based algorithm because flower algorithm tries to mimic the pollination characteristics of flowering plants and the associated flower consistency of some pollinating insects.

1.2.3 Physics and Chemistry based

Not all metaheuristic algorithms are bio-inspired, because their sources of inspiration often come from physics and chemistry. For the algorithms that are not bio-inspired, most have been developed by mimicking certain physical and/or chemical laws, including electrical charges, gravity, river systems, etc. As different natural systems are relevant to this category, we can even subdivide these into many subcategories which are not necessary. A list of these algorithms is given in Table 1.

Schematically, we can represent the relationship of physics and chemistry based algorithms as the follows [22]:

$$\frac{\text{Physics algorithms}}{\text{Chemistry algorithms}} \begin{cases} \notin \text{bio-inspired algorithms} \\ \in \text{nature-inspired algorithms} \end{cases} \quad (2)$$

Though physics and chemistry are two different subjects, however, it is not useful to subdivide this subcategory further into physics-based and chemistry. After all, many fundamental laws are the same. So we simply group them as physics and chemistry based algorithms.

1.2.4 Other algorithms

When researchers develop new algorithms, some may look for inspiration away from nature. Consequently, some algorithms are not bio-inspired or physics/chemistry-based, it is sometimes difficult to put some algorithms in the above three categories, because these algorithms have been developed by using various characteristics from different sources, such as social, emotional, etc. In this case, it is better to put them in the other category, listed in Table 1 [22].

1.2.5 Some Remarks

Though the sources of inspiration are very diverse, the algorithm designed from such inspiration may be equally diverse. However, care should be taken, as true novelty is a rare thing. For example, there are about 28,000 living species of fish; this cannot mean that researchers should develop 28000 different algorithms based on fish. Therefore, one cannot call their algorithms trout algorithm, squid algorithm or shark algorithm[22].

It is worth pointing out that studies show that some algorithms are better than others which are still not quite understood why. However, if one looks at the intrinsic part of algorithm design closely, some algorithms are badly designed, which lack certain basic capabilities such as the mixing and diversity among the solutions. In contrast, good algorithms have both mixing and diversity control so that the algorithm can explore the vast search space efficiently, while converge relatively quickly when necessary. Good algorithms such as particle swarm optimization, differential evolution, cuckoo search and firefly algorithms all have both global search and intensive local search capabilities, which may be partly why they are so efficient.

Table 1. A list of algorithms

Swarm intelligence based algorithms		Bio-inspired (not SI-based) algorithms	
Algorithm	Author	Algorithm	Author
Accelerated PSO	Yang et al.	Atmosphere clouds model	Yan and Hao
Ant colony optimization	Dorigo	Biogeography-based optimization	Simon
Artificial bee colony	Karaboga and Basturk	Brain Storm Optimization	Shi
Bacterial foraging	Passino	Differential evolution	Storn and Price
Bacterial-GA Foraging	Chen et al.	Dolphin echolocation	Kaveh and Farhoudi
Bat algorithm	Yang	Japanese tree frogs calling	Hern´andez and Blum
Bee colony optimization	Teodorovi´c and Dell'Orco	Eco-inspired evolutionary algorithm	Parpinelli and Lopes
Bee system	Lucic and Teodorovic	Egyptian Vulture	Sur et al.
BeeHive	Wedde et al.	Fish-school Search	Lima et al.
Wolf search	Tang et al.	Flower pollination algorithm	Yang
Bees algorithms	Pham et al.	Gene expression	Ferreira
Bees swarm optimization	Drias et al.	Great salmon run	Mozaffari
Bumblebees	Comellas and Martinez	Group search optimizer	He et al.
Cat swarm	Chu et al.	Human-Inspired Algorithm	Zhang et al.
Consultant-guided search	Iordache	Invasive weed optimization	Mehrabian and Lucas
Cuckoo search	Yang and Deb	Marriage in honey bees	Abbass
Eagle strategy	Yang and Deb	OptBees	Maia et al.
Fast bacterial swarming algorithm	Chu et al.	Paddy Field Algorithm	Premaratne et al.
Firefly algorithm	Yang	Roach infestation algorithm	Havens
Fish swarm/school	Li et al.	Queen-bee evolution	Jung
Good lattice swarm optimization	Su et al.	Shuffled frog leaping algorithm	Eusuff and Lansey
Glowworm swarm optimization	Krishnanand and Ghose	Termite colony optimization	Hedayatzadeh et al.
Hierarchical swarm model	Chen et al.	**Physics and Chemistry based algorithms**	
Krill Herd	Gandomi and Alavi	Big bang-big Crunch	Zandi et al.
Monkey search	Mucherino and Seref	Black hole	Hatamlou
Particle swarm algorithm	Kennedy and Eberhart	Central force optimization	Formato
Virtual ant algorithm	Yang	Charged system search	Kaveh and Talatahari
Virtual bees	Yang	Electro-magnetism optimization	Cuevas et al.
Weightless Swarm Algorithm	Ting et al.	Galaxy-based search algorithm	Shah-Hosseini
Other algorithms		Gravitational search	Rashedi et al.
Anarchic society optimization	Shayeghi and Dadashpour	Harmony search	Geem et al.
Artificial cooperative search	Civicioglu	Intelligent water drop	Shah-Hosseini
Backtracking optimization search	Civicioglu	River formation dynamics	Rabanal et al.
Differential search algorithm	Civicioglu	Self-propelled particles	Vicsek
Grammatical evolution	Ryan et al.	Simulated annealing	Kirkpatrick et al.
Imperialist competitive algorithm	Atashpaz-Gargari and Lucas	Stochastic difusion search	Bishop
League championship algorithm	Kashan	Spiral optimization	Tamura and Yasuda
Social emotional optimization	Xu et al.	Water cycle algorithm	Eskandar et al.

1.3 Applications

Nature Inspired Computing techniques are so flexible that they can be applied to wide range of problems, so adaptable that they can deal with unseen data and capable of learning, so robust that they can handle incomplete data. They have decentralized control of computational activities.

Biological inspired computing is a subset of nature inspired computing. There are three key differences between traditional computing systems and biological information processing systems: components of biological systems respond slowly but implement much higher-level operations. The ability of biological systems to assemble and grow on their own enables much higher interconnection densities. The implementation of biological systems is not a planned one.

One who expects solutions from nature for complex problems has to first observe the nature's behaviour carefully. The next step is to use models and list all the behaviours observed so far. The

above steps should be repeated till a near perfect working model is obtained. As a by-product some unknown mechanisms may be found. Based on the observation from nature a problem-solving strategy is formulated. Two main computational applications of NIC are Clustering and Optimization.

1.3.1 Clustering

Clustering is the unsupervised classification of patterns (observations, data items or feature vectors) into groups (clusters). A loose definition of clustering could be the process of organizing objects into groups whose members are similar in some way.

A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs.

For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties ("natural" data types), in finding useful and suitable groupings ("useful" data classes) or in finding unusual data objects (outlier detection).

1.3.2 Optimization

An optimization problem is the problem of finding the best solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete.

Classification of optimization algorithm can be carried out in many ways. A simple way is to look at the nature of the algorithm, and this divides the algorithm into two categories: deterministic algorithms, and stochastic algorithms.

Deterministic algorithms follow a rigorous procedure, and its path and values of both design variables and the function are repeatable. On the other hand, stochastic algorithms always have some randomness and each individual path towards a feasible solution is not exactly repeatable.

II. MOLECULAR DOCKING

2.1 Purpose and Motivation

Proteins found in nature have evolved by natural selection to perform specific functions. The biological function of a protein is linked to its three-dimensional structure or conformation. Consequently, protein functionality can be altered by changing its structure.

The idea in molecular docking is to design pharmaceuticals computationally by identifying potential drug candidates targeted against proteins. The candidates can be found using a docking algorithm that tries to identify the bound conformation of a small molecule, also referred to as ligand, to a macromolecular target at its active site, which is the region of an enzyme (large proteins that catalyse chemical reactions) where the natural substrate (specific molecule an enzyme acts upon) binds.

Often, the active site is located in a cleft or pocket in the protein's tertiary structure. The structure and stereochemistry (spatial arrangement of the atoms) at the active site complements the shape and physical/chemical properties of the substrate so as to catalyse a particular reaction. The purpose of drug discovery is thus to derive drugs or ligands that bind stronger to a given protein target than the natural substrate. By doing this, the bio-chemical reaction that enzyme catalyses can be altered or prevented.

Until recently, drugs were discovered by chance via a trial-and-error manner using a high-throughput screening methods that experimentally test a large number of compounds for activity against the target in question. This process is very expensive and time consuming. If a three- dimensional structure of the target exits, simulated molecular docking can be a useful tool in the drug discovery process because it allows for many possible lead candidates to be tested before committing expensive resources for wet lab experiments (synthesis), toxicological testing, bioavailability and clinical trials.

The focus of the below sections will be on realizing the complexity of simulating such a docking of ligand into the active site of target protein and a brief review of the most representative bio-inspired metaheuristic algorithms for doing the same.

2.2. Problem Definition

Molecular docking may be defined as an optimization problem, which would describe the "best-fit" orientation of a ligand that binds to a particular protein of interest. Figure 1 below illustrates docking of a small molecule into a protein's active site where solvent molecules are indicated by filled circles [23]:

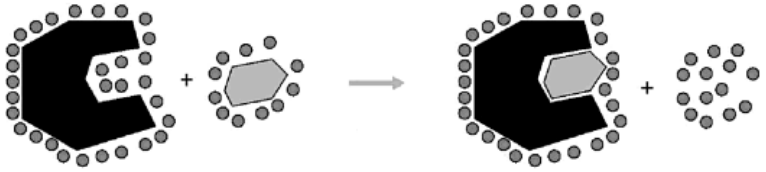


Figure 1: Protein Ligand docking in solvent

Protein-ligand docking can also be defined as a search problem where the task is to find the best ligand conformation (low energy binding mode) within the active site of protein. Since the relative orientation and conformation of both the protein and the ligand is taken into account it is not an easy problem. Usually, the protein is treated as fixed in three-dimensional coordinate system while the ligand is allowed to be flexible. The flexibility of the ligand is measured in terms of the number of rotatable bonds in the ligand. Following are the four types of Docking Problem based on increasing complexity:

(i) Protein-Rigid & Ligand-Rigid
(ii) Protein-Rigid & Ligand-Flexible
(iii) Protein-Flexible & Ligand-Rigid
(iv) Protein-Flexible & Ligand-Flexible

Category (iv) can be further divided in two cases:

(iv.a) Rigid protein backbone, selected side chains are flexible
(iv.b) Fully flexible protein backbone and side chains.

It can be thought of as a problem of "lock-and-key", where one is interested in finding the correct relative orientation of the "key" which will open up the "lock" (where on the surface of the lock is the key hole, which direction to turn the key after it is inserted, etc.). Here, the protein can be thought of as the "lock" and the ligand can be thought of as a "key". Molecular docking may be defined as an optimization problem, which would describe the "best-fit" orientation of a ligand that binds to a particular protein of interest. However, since both the ligand and the protein are flexible, a "hand-in-glove" analogy is more appropriate than "lock-and-key". During the course of the process, the ligand and the protein adjust their conformation to achieve an overall "best-fit" and this kind of conformational adjustments resulting in the overall binding is referred to as "induced-fit" as illustrated in figure 2 below:

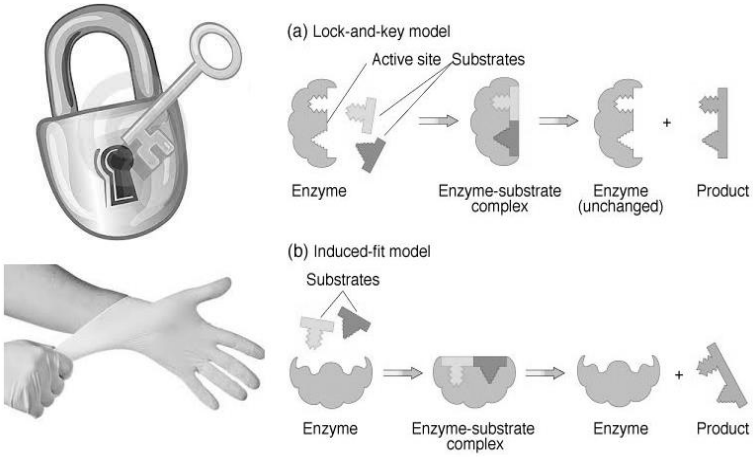


Figure 2: Analogy for docking problem

2.2.1 The Docking Problem

Let A and B be a ligand and protein molecule, respectively. Further, let f be a scoring function(energy function) that ranks solutions with respect to binding energy and let C be the conformational search

space of all possible conformations(docking solutions) between A and B. Then we can define docking problem as an optimization (minimization) problem in which the task is to find the following [23]:

$$\text{a } x \in C \text{ satisfying } f(x) \leq f(y) \forall\, y \in C \quad (3)$$

A docking problem requires [24]:
1) A Scoring function to score protein-ligand complex by measuring the binding strength of the ligand at a specific position of the protein.
2) A search algorithm to search for the protein-ligand combination that corresponds to the lowest energy conformation.

2.2.2 Complexity of Docking Problem

Docking problems have huge search spaces, and the number of possible conformations to consider increases dramatically when flexibility is taken into account. Unfortunately, the search space is not unimodal but rather highly multimodal due to the numerous local optima caused by the energy function used. Moreover, the complexity of the docking problem is influenced by the size, shape and bonding topology of the actual ligand being docked.

Despite great improvements in computational power, docking remains a very challenging problem, generally considered to be NP-hard (although no formal proof exists) [23]. Thus, a brute-force approach looking at all possible docking conformations is impossible for all but the simplest docking problems. To handle docking flexibility in an efficient manner, search heuristics that sample the search space of possible conformations are required.

2.3. Survey of Bio Inspired Algorithms on Docking Problem

In most of the protein- ligand docking problems in literature, which has been solved by evolutionary algorithms, the protein is kept rigid and ligand's 3 translational, 3 rotational and r torsional degrees of freedom is optimized. Thus the total number of variables, the dimension of the optimization problem equals 6+r. A wide variety of optimization strategies is used to find the global minimum corresponding to a complex structure.

One of the first applications of EAs to molecular docking was introduced by Dixon [3] .He used a simple GA with binary encoding representing a matching of ligand atoms and spheres placed inside the active site similar to DOCK (Kuntz et al., 1982. He used incremental construction algorithms.) Only limited experimental results were presented by him. Later on, Oshiro, Kuntz and Dixon [16] introduced a similar GA using a DOCK scoring function. This time the experiments resulted in the low root mean square deviation (RMSD) values.

Genetic Algorithm for Minimization of Energy (GAME) by Xioa and Williams [18] was another early attempt for molecular docking. GAME was designed to dock one or more rigid ligand to rigid protein. Solutions were encoded as binary strings representing three rotational angles and three translational coordinates. The algorithm was tested on one protein ligand complex. It was done with different search strategies firstly; ligand was allowed to move with fixed rotation. Secondly, ligand was allowed to rotate with fixed translation. At last all six degrees of freedom have been taken into account.

Clark and Ajay [1] in 1995 introduced DIVALI. It used Gray-coded binary representation, bit-flip mutation, and an elitism scheme. Two point crossover was used as it was believed to perform better. The protein remained rigid during docking process whereas the ligand was flexible. Its key feature was masking operator. It was evaluated on four protein ligand complexes using an AMBER-like force field as scoring function and obtained good docking results.

Genetic Optimization for Ligand Docking, GOLD was developed by Jones et al [9].They used a mixed encoding combining binary string and real valued numbers. Binary strings were used to represent torsion angles of the ligand and selected side chain angles of the protein thus giving partial flexibility to protein. Integer strings were used to represent mappings of hydrogen bonding sites between ligand and protein. Both number and strength of hydrogen bonds and Van der Waals energy contributed in the fitness function there. Standard one point crossover and bit-flip mutation were used.

Levine et al [12] developed Stalk system in which they applied new technologies to molecular docking prediction. They combined the concepts of parallel and distributed computing, high speed

networking, genetic algorithms and virtual reality for the study of molecular interactions. In Stalk they have used CAVE (CAVE Automatic Virtual Environment) as the virtual reality system. Stalk considered a rigid body docking in which it is assumed that no modification of proteins backbone or side chain occurs. Each string of GA contained six parameters three translational and three rotational. The fitness function included non-bonded interactions only. Stalk was run on one protein-ligand complex and showed good docking results.

AutoDock is another popular docking program that originally used Simulated annealing which was later replaced by Genetic Algorithm to search for promising docking conformations. GA was then replaced by LGA, Lamarckian Genetic Algorithm (Morris et al [14] which was a hybrid GA combining a genetic algorithm and local search. LGA used a real valued representation, two point crossover and Cauchy mutation with fixed value of, defining the mean and spread of Cauchy distribution. In their representations each chromosome (solution) was composed of a string of real valued genes: three Cartesian coordinates for the ligand translation; four variables defining a quaternion specifying the ligand orientation; and one for flexible torsion angle. Further, local search was applied on each generation on a user defined proportion of population. When local search is applied the genotype of the individual is replaced with the new best solution found by evaluating its fitness, which is the sum of intermolecular interaction energy between the ligand and the protein and the intramolecular interaction energy of the ligand. The implementations of AutoDock on various protein-ligand complexes and corresponding numerical results have been presented (Morris et al, 1998). There they concluded that out of three search algorithms (SA, GA, LGA) which they have applied in AutoDock, the most efficient, reliable and successful is Lamarckian Genetic Algorithm LGA.

Vieth et al.[17] compared the efficiency of molecular dynamics (MD), Monte Carlo (MC) and Genetic Algorithms (GA) for docking five representative ligand- receptor complexes. All the three algorithms employed CHARMM-based energy function. The results were also compared with AutoDock. The receptor was kept rigid while flexibility of ligand was permitted. They concluded that MD was most efficient in case of large search spaces while GA outperformed others in small search spaces.

In 1999, Wang [7] et al. applied GA combined with random search. Steric complementarity and energetic complementarity of ligand with its receptor have been separately considered in two stage automated docking. Eight complexes have been randomly selected. For most of the cases the root mean square (RMS) of the GA solutions was smaller than 1.0. In the first stage rough searching of a set of bound sites based on steric complementarity was done whereas in the second stage detailed searching of the locally associated sites based on energetic complementarity was performed.

With the passage of time several other algorithms have been developed, changes in variation operators have been made to improve the efficiency of docking algorithms. Multi-Objective, multi-population genetic algorithms came into picture. Of recent improvements C-l Li et al. [13] proposed an information entropy based evolution model for molecular docking. The model was based on a multi-population genetic algorithm. The GA proposed to solve this was binary coded. In each generation there were three operators: selection, crossover and mutation. Selection was performed by an integer decimal method, crossover was two-point and mutation used was uniform. An elitist maintaining mechanism was designed in the proposed GA.

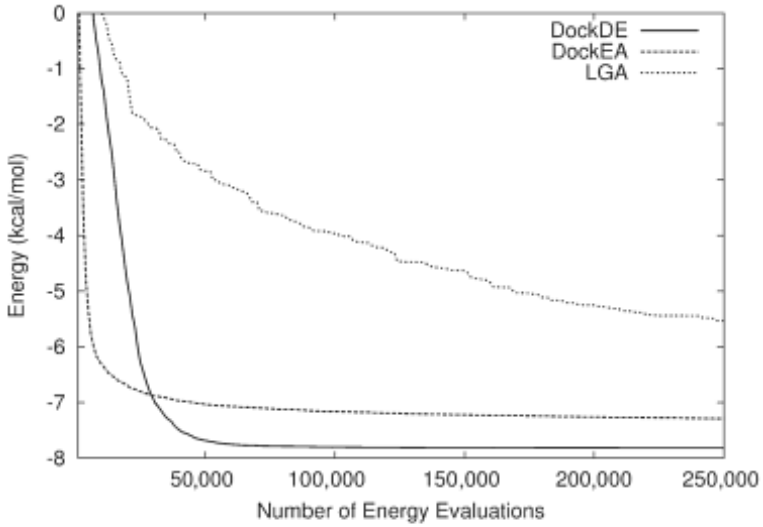
Krob et al.[15] applied Ant Colony Optimization to structure based drug design. They introduced a new docking algorithm PLANTS (Protein-Ligand Ant System) based on ACO metaheuristic. An artificial Ant Colony was employed to find a minimum energy conformation of the ligand in the proteins active site. They presented the effectiveness of PLANTS for several parameter settings as well as a direct comparison to GOLD, which is based on a GA. PLANTS, treated the ligand as flexible while almost whole receptor except rotatable hydrogen bond donors were kept rigid. The continuous variables have been discretized so that ACO can directly be applied, as it is designed to tackle combinatorial optimization problems.

Most recently Janson [8] et al. have done work on molecular docking with multi-objective particle swarm optimization (PSO). A new algorithm ClustMPSO was proposed which is based on PSO and follows a multi-objective approach for comparing the quality of solutions. Energy evaluations were done using AutoDock tools. In the second part of the paper a new approach for predicting the docking trajectory was proposed. It was shown empirically that ClustMPSO finds solution for docking

problem with lower energy than LGA and SA that are incorporated in AutoDock. ClustMPSO is also faster with respect to the solution evaluations.

Differential evolution (DE) was introduced by Storn and Price[19]. Compared to more widely known EA-based techniques (e.g. genetic algorithms, evolutionary programming and evolution strategies), DE uses a different approach to select and modify candidate solutions. The main innovative idea in DE is to create offspring from a weighted difference of parent solutions. The DE algorithm has recently demonstrated superior performance, outperforming related heuristics, such as GAs and PSO on both numerical benchmark functions and in several real world applications.

Based on the great success of DE, a novel application of DE to flexible ligand docking (termed DockDE) was introduced by Thomsen [20]. The comparison was performed on a suite of six commonly used docking benchmark problems where DockDE obtained the best energy for all of them compared to the DockEA and LGA algorithms as depicted in the below figure 3 extracted from [20].



Average energy of best observed ligand conformations for the 1stp complex using the DockDE, DockEA, and LGA algorithms (mean of 30 docking runs).

Figure 3. Energy Graph of DockDE against LGA and DockEA

In a recent study, a new docking algorithm called MolDock was introduced by Thomsen and Christensen [21] which is based on a hybrid search algorithm that combines the DE optimization technique with a cavity prediction algorithm.

Accuracy of Selected Docking Programs

Docking Program	Accuracy (%)
MolDock	87
Glide	82
Surflex	75
GOLD[a]	78
FlexX[b]	58

[a]The docking accuracy is based on 55 out of the 77 complexes.
[b]The docking accuracy is based on 76 out of the 77 complexes.

Figure 4. Comparative Survey of MolDock docking accuracy

The docking accuracy of MolDock on selected 77 complexes is 87%, outperforming the other docking algorithms on the benchmark data set which is depicted in above Figure 4 extracted from [21]. In conclusion, the results strongly suggest that the Differential Evolution algorithm has a great potential for protein-ligand docking.

III. CONCLUSIONS

Molecular docking is a valuable tool for pharmaceutical companies because they can be used to suggest promising drug candidates at a low cost compared to making real-world experiments. As a result of the need for methods that can predict the binding of two molecules when complexed, numerous algorithms have been suggested during the last three decades. In this survey, a review of the most representative bio-inspired optimization algorithms for protein-ligand docking was studied. In particular, docking methods utilizing differential evolution have been surveyed to have superior performance to other well studied approaches regarding accuracy and speed, issues that are important when docking hundreds of thousands of ligands in virtual screening experiments.

IV. FUTURE/PROPOSED WORK

Cancer is a serious and often fatal disease, widespread in the developed world. One of the most common methods of treating cancer is chemotherapy with cytotoxic drugs. These drugs, around 30 used regularly in chemotherapy, themselves can often have debilitating or life threatening effects on patients undergoing chemotherapy. Hence chemotherapy can be perceived as a complex control problem of how to balance the need for tumour treatment against the undesirable and dangerous side effects that the treatment may cause. Proposed work can be subdivided into two phases:

A) Literature Survey and Virtual screening for drug delivery

Here we survey current scientific literature to track potential cytotoxic Ligands. Then we use molecular docking tools to simulate novel protein-ligand complexes using these potential candidates for virtual drug on the basis of scoring function and optimized conformation.

B) Drug Administrating Optimization

Here we design a system, specifically a workbench, which supports the design of novel multidrug cancer chemotherapy treatments. The aim of the system is to allow clinicians to interact with models of tumour response and to use nature-inspired optimization algorithm to search for improved treatments. In simple words, the system should take in effective input of patient current status of tumour growth and provide output of administered dose of a combination of cytotoxic drug for optimal control.

ACKNOWLEDGEMENTS

The author would like to thank Department of Information Technology, Jadavpur University, Salt lake Campus for providing necessary access to scientific literature while conducting this survey. The author is grateful to Dr Parama Bhaumik, for guidance and valuable technical inputs. Finally, the author is indebted to constant motivation and grammatical editing from his better-half Mrittika Chatterjee.

REFERENCES

[1]. Clark, K.P. and A.N. Jain (1995) "Flexible ligand docking without parameter adjustment across four ligand receptor complexes," J. Comput. Chem., Vol. 16, pp. 1210-1226.
[2]. Deb, K.(2005). Optimization For Engineering Design, Prentice-Hall of India Pvt. Limited, New Delhi.
[3]. Dixon, J.S.(1993). "Flexible docking of ligands to receptor sites using genetic algorithms," Proceedings of the 9th European Symposium on Structure-Activity Relationships: QSAR and Molecular Modelling, pp 412-413.
[4]. Dorigo, M., T. Stiitzle: Ant Colony Optimization.MIT Press, Cambridge, MA, USA 2004.
[5]. Goldberg, D.E. (1989). Genetic algorithms in search, optimization and machine learning, Addision – Wesley, New York.
[6]. Hart, W.E., C. Rosin, R.K. Belew, and G.M. Morris (2000). "Improved evolutionary hybrids for flexible ligand docking in AutoDock," Optim. Comput. Chem. Mol. Biol., pp. 209-230.
[7]. Hou, T., J. Wang, L. Chen, and X. Xu (1999) "Automated docking of peptides and proteins using a genetic algorithm combined with a tabu search," Pro. Engi., Vol. 12 No.8, pp. 639-647.
[8]. Janson, S., D. Merkle, and M. Middendorf (2008) "Molecular docking with multi-objective Particle Swarm Optimization," App. Soft Computing Vol. 8, pp. 666-675.
[9]. Jones, G., P. Willett, and R.C. Glen (1995). "Molecular recognition of receptor sites using a genetic

algorithm with a description of desolvation." J. Mol. Biol., Vol. 245, pp. 43-53.
[10]. Kennedy, J., R. Eberhart, Particle Swarm Optimization, IEEE International Conference on Neural Networks (ICNN'95), Vol. 4, 1995.
[11]. Kuntz, I.D., J.M. Blaney, S.J. Oatley, R. Langridge and T.E. Ferrin (1982). "A geometric approach to macromolecule-ligand interactions," J. Mol. Biol., Vol. 161, pp. 269-288.
[12]. Levine, D., M. Facello, P. Hallstrom, G. Reeder, B. Walenz, and F. Stevens (1997). "Stalk: An interactive system for virtual molecular docking," IEEE Comput. Sci. & Engi., Vol. 4, Issue 2, pp. 55-65.
[13]. Li, Chun-lian, Y. Sun, D. Long, and X. Wang (2005) "A genetic algorithm based method for molecular docking," ICNC: International conference on advances in natural computation. LNCS 3611, pp. 1159-1163.
[14]. Morris, G.M., D.S. Goodsell, R.S. Hallyday, R. Huey, W.E. Hart, R.K. Belew, and A.J. Olson (1998). "Automated docking using Lamarckian genetic algorithm and an empirical binding free energy function," J. Comput. Chem., Vol. 19, pp. 1639-1662.
[15]. Oliver, K., T. Stiitzle, and T.E. Exner (2006) "PLANTS: Application of Ant Colony Optimization to Structure-Based Drug Design," 5th International workshop on Ants (proceedings), vol. 4150, pp. 247-258.
[16]. Oshiro, C.M., Kuntz, and J.S. Dixon (1995). "Flexible ligand docking using a genetic algorithm," J. Comput. Aid. Mol. Des., Vol. 9, pp. 113-130.
[17]. Vieth, M., J.D. Hirst, B.N. Dominy, H. Daigler, and C.L. Brooks III (1998). "Assessing search strategies for flexible docking," J. Comput. Chem.,Vol. 19, pp. 1623-1631.
[18]. Xiao, Y.L. and D.E. Williams (1994), Proceedings of the 1994 ACM symposium on Applied computing. pp. 196-200.
[19]. Rainer Storn, Kenneth Price (1997), Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,December 1997, Volume 11, Issue 4, pp 341-359
[20]. Thomsen, R.(2003), Flexible ligand docking using differential evolution, Evolutionary Computation, 2003. CEC '03. The 2003 Congress on (Volume:4)
[21]. Thomsen R1, Christensen MH.(2006), MolDock: a new technique for high-accuracy molecular docking,J Med Chem. 2006 Jun 1;49(11):3315-21.
[22]. Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, Dušan FisterBbb(2013), A Brief Review of Nature-Inspired Algorithms for Optimization, Elektrotehni\v{s}ki vestnik, 80(3), 2013
[23]. Thomsen, R.(2007), Protein-Ligand Docking with Evolutionary Algorithms, Computational Intelligence in Bioiformatics, IEEE press 2007, pp. 169-196
[24]. Shashi, Kusum Deep, V.K Katiyar, C.K Katiyar(2009),A state of art review on application of nature inspired optimization algorithms in protein-ligand docking, Indian Journal of Biomechanics: Special Issue (NCBM 7-8 March 2009)
[25]. Jiming Liu, K.C. Tsui,Toward nature-inspired computing, Communications of the ACM, Vol. 49 No. 10, Pages 59-64

AUTHOR

The Author is presently working as a BI developer for British Telecommunications Retail Team in DSS platform and pursuing Jadavpur University-TCS collaborative Masters of Engineering in Software Engineering.