International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 3 (2011) pp. 248-255

Improving Protein Docking Using Sustainable Genetic Algorithms

Emrah Atilgan[1], Jianjun Hu[2]

[1] Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, 29169, USA
atilgan@email.sc.edu

[2] Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, 29169, USA
Jianjunh@cse.sc.edu

***Abstract*: AutoDock is a widely used automated protein docking program in virtual screening of structure-based drug design. Several search algorithms such as simulated annealing, traditional genetic algorithm (GA), and Lamarckian genetic algorithm (LGA) are implemented in AutoDock to find optimal conformation with the lowest binding energy. However, the docking performance of these algorithms is still limited by the local optima issue of simulated annealing and traditional evolutionary algorithms (EA). Due to the stochastic nature of these search algorithms, users usually need to run multiple times to get reasonable docking results, which is time-consuming. We have developed a new docking program AutoDockX by applying a sustainable GA named ALPS to the protein docking problem. We tested the docking performance over three different proteins (pr, cox and hsp90) with more than 20 candidate ligands for each protein. Our experiments showed that the sustainable GA based AutodockX achieved significantly better docking performance in terms of running time and robustness than all the existing search algorithms implemented in the latest version of AutoDock. AutodockX thus has unique advantages in large-scale virtual screening.**

***Keywords*:** autodock, protein docking, genetic algorithm, HFC, sustainable evolutionary algorithms

I. Introduction

Computational docking of ligands to protein structures is a key step in identifying potential drug candidates. The docking problem has been formulated into a ligand-protein binding energy optimization problem. Dozens of programs have been developed for molecular docking [1-9]. In any docking scheme, two requirements must be balanced: to get better precision with lower binding energy and to minimize the computational time. Recently, there is significant progress in computational protein docking [10]. The first type of new docking programs utilizes Fast Fourier Transformation (FFT) for efficient sampling of the conformation spaces [1;11-14]. The second type of programs exploits biochemical knowledge of the docking process to improve the docking performance. These include docking algorithms that use shape or physicochemical complementarily information [4;15;16] and predicted binding site information [17]. Another category of new docking algorithms are focused on improving the energy/potential function used for docking [18-22]. Finally, since the first successful application of genetic algorithms for protein docking, there has been significant progress in the global optimization field. New optimization algorithms such as spider-search [23], hybrid GA [24;25], differential evolution [26] have all been successfully applied to a variety of engineering problems. This paper followed this trend to apply new sustainable genetic algorithms to the protein docking problem.

One of the most widely used automated docking programs is AutoDock, which predicts how small molecules bind to a receptor of known 3D structure [27]. AutoDock uses three different conformation search algorithms: simulated annealing (SA), traditional genetic algorithm (GA), and Lamarckian genetic algorithm (LGA). However, all three search algorithms are subject to the local optima issue. And due to the stochastic nature of the search algorithm, users usually need to run multiple (such as 10-15) independent runs to get reasonable results. This practice thus significantly increases the computational time needed (10 times or more) for protein docking, which becomes a major issue for large-scale virtual screening experiments with millions of ligands to be docked to a given protein. To overcome these limitations of AutoDock and get better docking performance, we proposed to apply sustainable evolutionary algorithms [28;29], a new type of genetic algorithms, to protein docking. We integrated the ALPS sustainable evolutionary algorithm [28;30;31] into AutoDock and compared its performance with those of current algorithms using the lowest binding energy and computational time criteria. Our results showed that the main advantage of sustainable evolutionary algorithms is their capability to address the premature convergence problem typical in traditional genetic algorithms [32;33]: an evolutionary algorithm cannot improve the quality of the best identified solution after some number of evaluations or generations. Our experiments showed that sustainable evolutionary algorithms can help to address the premature convergence problem of traditional GAs and have achieved significantly better binding conformation using less running time. As the number of generations increases, sustainable

evolutionary algorithms are able to find better results while traditional genetic algorithms get stuck in local optima. According to a recent survey of protein docking algorithms [1], there are more than 50 protein-ligand docking softwares, most of which still used traditional GAs for conformation search optimization. Our experiments implied that other modern protein-ligand docking programs can also be potentially improved by the sustainable genetic algorithms.

II. Background

A. *Protein-Ligand Docking*

Protein docking is a method that predicts the bound conformation of one protein to another protein or a ligand. A docking algorithm aims to find the best orientation of these two molecules such that they have the minimum binding energy as scored by a predefined scoring function. There are two key components in a docking algorithm: a good scoring function with high selectivity and efficiency that distinguishes between correctly or incorrectly docked structures and a search algorithm that can efficiently do global minimization of the scoring function [34-36].

Protein-ligand docking algorithms can be classified into two methods. In early docking algorithms, both protein and ligand are considered as rigid bodies and they have only six degrees of translational and rotational freedom to search for best orientations. Since the number of degrees of freedom is large if the proteins are modeled as flexible, it is impractical to perform exhaustive conformational search. Most of current docking algorithms consider the flexibility of ligands to find the best binding position between small molecules (ligands) such as substrates or drug candidates and structurally known target proteins (see Figure 1). Interaction between proteins produces no change in conformation. Flexibility of ligands comes from the rotatable bonds (also called *torsions*) of a ligand (see Figure 2). The number of optimization variables is composed of six degrees of freedom for rotation and translation plus the number of torsion angels. The ligand finds its position into the protein's active site after a certain number of moves (searches) in its conformational space. Flexibility modeling allows the ligand to change its structure with the torsions angles. Each move costs energy, and after moves are completed, total energy is computed by the system. Our goal is to minimize this binding energy to find the best conformation.

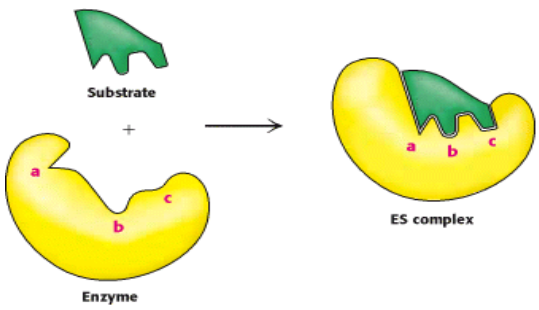

Figure 1. An example of protein-ligand docking

B. *Search Algorithms in AutoDock*

AutoDock (Automated Docking Software for Predicting Optimal Protein-Ligand Interaction) is a suite of automated docking tools. AutoDock is widely used as a docking engine

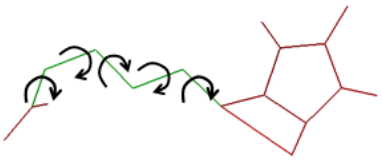

Figure 2. A ligand with rotatable bonds (torsions).

in virtual screening [37-39] for predicting how small molecules bind to a receptor of known 3D structure. In AutoDock [40], a ligand and a protein are defined by a set of values describing the translation, orientation and conformation of the ligand with respect to the protein. The target protein is represented as a grid. This three dimensional grid surrounds all atoms of the protein. Each atom in a protein has its own points in the space. The representation of a ligand consists of 3 coordinates of the location of the ligand (x, y, z), followed by the 4 quaternion parameters q_x, q_y, q_z, q_w, which define the orientation of the small molecule, and followed by the number of torsions $n_1, n_2, \dots, n_m$, depending on how many rotatable bonds the ligand has [8] (see Figure 3). These are the state variables of the ligand, and each state variable corresponds to a gene. The ligand's state corresponds to the genotype, and the atomic coordinates of the state corresponds to the phenotype [27].

$$(\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{q_x}, \mathrm{q_y}, \mathrm{q_z}, \mathrm{q_w}, \mathrm{n_1}, \mathrm{n_2}, \dots \dots, \mathrm{n_m})$$

Figure 3. Representation of a ligand as a vector

Autodock implements three conformation search algorithms for docking including simulated annealing (SA), traditional genetic algorithm (GA), and Lamarckian genetic algorithm (LGA).

1) *Simulated Annealing*

In early versions of AutoDock, Simulated Annealing (SA) was used as the major optimization method [41-43]. Simulated annealing is a generic probabilistic method for global optimization. The algorithm starts from a random or specific state with an initial temperature parameter (T0) and a specific cooling scheme [41]. At each step of the simulation, the ligand explores the conformation space by adding a small random displacement in each degree of freedom and evaluating the binding energy for the new conformation, which is composed of the intermolecular energy between the protein and the ligand and the intra-molecular energy of the ligand. It repeatedly searches the neighborhood and selects a neighbor as a new state. New energy is compared to the energy of the previous step. If the new energy is lower, the step is accepted. Otherwise, if the new energy is higher, the decision is made probabilistically based on a temperature (T) parameter. Because simulated annealing is a kind of a Monte Carlo method, different runs may produce different solutions

[40]. However, it does not guarantee to find the global minimum conformation [41].

2) Genetic Algorithm

A genetic algorithm is a population-based search technique used to find appropriate solutions to optimization and search problems. In Autodock, a random population of individuals is generated by initializing each individual as a vector composed of a set of uniformly distributed random values between the minimum and maximum x, y, and z values [27]. Also, the genes representing torsion angles are given random values between -180 and +180. The fitness value of an individual is the binding energy between ligand and the target protein [27]. Two-point crossover is used. *Mutation* operator is performed by adding a random real number that has a Cauchy distribution to the variable, where α and β are parameters that affect the mean and spread of the distribution. *Elitism* operator is used to keep top individuals in the population.

3) Hybrid Global-Local Search Algorithm: Lamarckian Genetic Algorithm (LGA)

Lamarckian genetic algorithm is the best search algorithm used in AutoDock so far. LGA in Autodock uses Solis-Wets local search after each generation of genetic algorithm search for energy minimization. The result of the local search is used to update the fitness value and its representation associated with an individual. Even though Solis and Wets local search operator searches through the genotypic space, it can still be qualified as Lamarckian, because any environmental adaptations of the ligand acquired during the local search will be inherited by its offspring [27].

C. Limitations of Current Search Algorithms in AutoDock

The major limitation of the search algorithms in current version of Autodock is that they can get trapped in local optima when the number of torsion angles increases. For example, SA performs well with the ligands that have roughly 8 rotatable bonds or less. But the algorithm becomes ineffective with ligands that have more than 8 rotatable bonds [27].

A common issue of genetic algorithms (for both traditional GAs and LGA used in AutoDock) is that after some generations, the algorithm is no longer able to improve the best fitness of the population. This problem is called *premature convergence* problem [28;33]. If a sub-optimal individual dominates the population, *selection* tends to keep it around and prevents further adaptation. This is because the average fitness of the population increases as the evolutionary process continues, and then only new individuals with similar genotype and similarly high fitness tend to survive. Very different new individuals usually have low fitness since their beneficial characteristics have not been expressed into fitness values until some exploration and exploitation. Thus, a traditional genetic algorithm tends to concentrate its search effort near one peak, thus getting stuck in local optimum [33]. One possible approach to avoid from premature convergence may be to increase mutation rate or population size. Increasing mutation rate will keep diversity high and not allow loosing good individuals; however, it is just as likely as to replace good alleles and building blocks as bad ones. If the mutation rate is too large, mutation operator cannot create offspring near its parent. Increasing population size takes much longer time than necessary even for a single run [28].
A common practice to address local optima issue is to run the above stochastic search algorithm multiple times (e.g. 10) and then select the best solution. This approach, however, significantly increases the running time. A recent sustainable evolutionary algorithm called HFC (Hierarchical Fair Competition) has been shown to be much more efficient than the multi-run method and can achieve significantly better solutions.

III. Methods

A. Sustainable Evolutionary Algorithms

Evolutionary algorithms usually fall prey to the local optima problem. Recently, a new type of evolutionary algorithms has emerged focused on sustainability, which refers that the capability to make constant progress given more computation resource unless the optimum is reached.

The first such algorithm is the HFC algorithm framework [29]. The main idea of HFC is to keep the diversity of the population and maintain a pipeline for generating individuals at all fitness or age levels. The pipeline structure of HFC refers to the hierarchical organization of the subpopulations by different fitness levels. HFC reduces the selection pressure within each subpopulation to encourage exploration while maintaining the global selection pressure to exploit good individuals. Because of its structure, HFC does not allow the convergence of the population to the vicinity of any set of sub-optimal solutions. HFC achieves sustainable searching by ensuring continuous supply and the incorporation of individuals in the hierarchical levels. HFC continually changes the populations of individuals of intermediate fitness levels. Another form of sustainable evolutionary algorithms similar to HFC uses a parameter 'age' instead of fitness value, to set up the hierarchy levels.

B. ALPS – Age Layered Population Structure

Another sustainable GA called Age-Layered Population Structure (ALPS) was recently proposed [28], which was shown to have better performance than HFC for some genetic algorithm test problems. ALPS follows HFC algoriths' hierarchical organization of individuals but defines a new attribute of an individual, *age,* which is used to restrict competition and breeding among individuals of the population. The '*age'* refers to a measure of how long the individual has been in the population. ALPS segregates individuals into different layers according to their ages, and regularly replaces all individuals in the bottom layer with randomly generated ones. Thus, the genetic algorithm will never completely converge and is always examining new areas of the fitness landscape. This allows genetic algorithms to develop promising young individuals without being dominated them by older ones [28].

In ALPS, the population consists of a sequence of layers with increasing upper-limits on the maximum age of individuals that a layer can contain (Figure 4). ALPS uses two restrictions for evolution of individuals. First, individuals can only breed with individuals in their own layer or one layer below. Second, the last layer is replaced with randomly

generated individuals at regular intervals. New individuals start with the age of 0, since their genetic material has just been through the evolution process. Other individuals that are created by mutation or recombination get the age of their parents plus 1 since their genetic material comes from their parents. An individual's age is incremented by 1 if it is used as a parent to create an offspring. If an individual is not used as a parent, its age will not be changed. Even if an individual produces offspring multiple times in one generation, its age still incremented by 1.

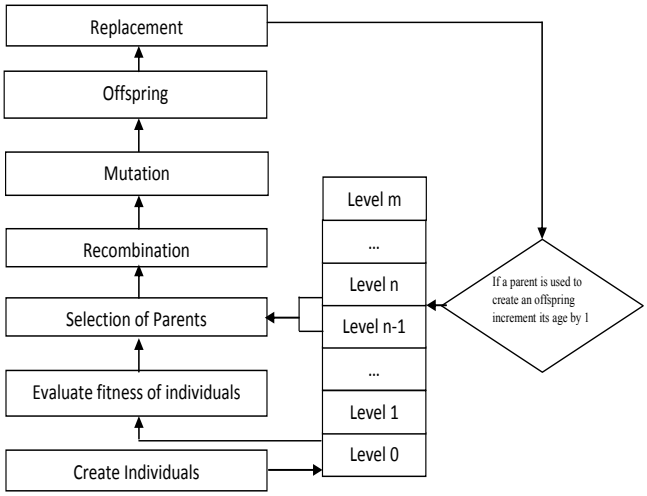

Figure 4. ALPS algorithm in AutoDockX

The population keeps individuals within age-layers to control competition and breeding. Each layer has a maximum age limit for individuals, only last level can have individuals of any age. Different schemes can be used for setting the age-limits for each age layer such as allocating equal number of individuals to each layer, or allocating more individuals to higher levels using polynomial or exponential distributed layer sizes. Then, these age-limits are multiplied by an age-gap parameter (Table 1). This allows younger individuals to be able to find and move into a good basin of attraction before they are pushed into next layer [28].

Table 1. Different age schemes with the age-gap 10

num_generations	Default	Description
num_evals	250000	No. of generations
pop_size	200	Population size
alps_number_layers	10	No. of layers
alps_age_gap	3	Age gaps for migration
alps_age_scheme	5	Age allocation scheme
alps_elitism	5	No. of elitism individuals
alps_tourn_size	5	Tournament selection size
alps_prob_select_prev	0.25	Selection probability
alps_recomb_prob	0.8	Crossover probability
alps_rec_rand2_prob	1.0	mutation probability

C. *Integrating ALPS into AutoDock*

To integrate a new search algorithm into Autodock, we have modified the global search algorithm of Autodock package and replaced the default traditional GA with ALPS to create a new docking program called AutodockX. Parameters of ALPS are defined in a parameter file. Some of ALPS's default parameters are defined in Table 2.

Table 2. Default ALPS parameters

	Age-Range of Layers				
Aging Scheme	L_0	L_1	L_2	L_3	L_4
Linear	0-10	11-20	21-30	31-40	41-50
Fibonacci	0-10	11-20	21-30	31-50	51-80
Polynomial (n^2)	0-10	11-40	41-90	91-160	161-250
Exponential (2^n)	0-10	11-40	41-80	81-160	161-320

IV. Experimental Results

A. *Test Data Preparation*

We tested the search algorithms in Autodock with ALPS on three different proteins, *pr*, *hsp90* and *cox,* from ZINC database. We tested 22 ligands for protein *cox*, 24 ligands for protein *hsp90*, and 27 ligands for protein *pr*. The ligands have different degrees of freedom and different types of atoms leading to different dimensions for global optimization by the search algorithms.

All the algorithms were tested on Optimus which is one of the high performance computing systems of the University of South Carolina. The specifications of Optimus are: 64 nodes, dual CPU, 2.0 GHz Dual-Core AMD Opterons, totaling 256 cores, 8GB RAM per node and 1 Terabyte of Storage in head node.

B. *Comparing Performance of ALPS versus GA and LGA*

To compare the docking performance of ALPS against traditional GA and Lamarckian GA, we calculated the docking energy of the best docking result for each algorithm for docking 22 ligands to cox protein. Figure 5 shows the binding energies of these docked conformations. It clearly shows that ALPS algorithm has achieved the lowest (best) docking energy over all 22 ligands. LGA works worse than ALPS but better than traditional GA, which sometimes obtained much worse results.

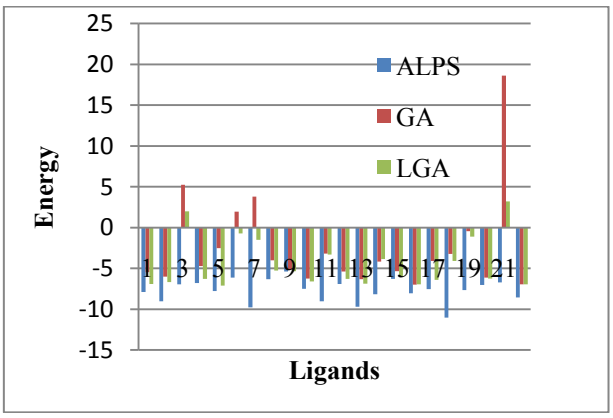

Figure 5. Comparison of three algorithms of 22 ligands with protein cox docking process. The population size was 50; maximum number of generations was 16000 for 10 runs.

To fully show that sustainable evolutionary algorithms such as ALPS can help to address the premature convergence problem of traditional GAs, we compared the performance of ALPS, GA, and LGA in docking ligands to 3 proteins using different numbers of evaluations. We set the population size to 50, and varied the number of generations as 500, 1000, 2000, 4000, 8000, 16000, 32000, which makes the total number of evaluations 25000, 50000, 100000, 200000, 400000, 800000, $1 * 10^6$, respectively. This allows testing whether a search algorithm can find better solutions given more computational time. The mutation rate was set to 0.02, crossover rate was set to 0.8 for GA and LGA. For ALPS, we set the number of layers to 10, age gap to 20, age scheme to exponential. Recombination probability was set to 0.8, probabilistic selective rate was set to 0.25 for ALPS. The number of runs for each experiment is set as 10. All three GAs used the same real-value crossover operator as defined in Autodock.

At the end of a docking process, Autodock output multiple conformation solutions organized into clusters. For simplicity, we only consider the best solution (lowest binding energy) for each protein-ligand pair. For each protein, we calculated the average of the binding energy for the given set of ligands docked to that protein.

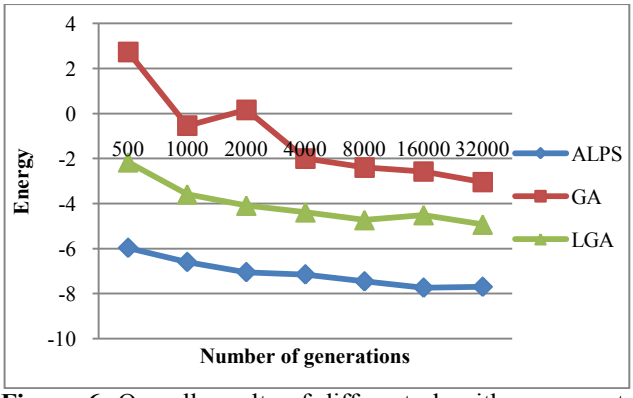

Figure 6. Overall results of different algorithms on protein cox using fixed population size 50 and varying maximum number of generations. Each algorithm was run 10 times for each protein-ligand pair. The averages of the lowest binding energy for all protein-ligand pairs are then calculated.

Figure 6 shows the results of 3 algorithms for docking 22 ligands to cox protein using different number of maximum generations ranging from 500 up to 16000. Note that these are NOT the average binding energy of a single run. Instead, for each allowed max generation number, we restarted the docking algorithms and calculated the average of lowest binding energy. Figure 4 showed that for each given maximum generation number, ALPS always gives lower (better) binding energies than the traditional GA and Lamarckian GA and in general, LGA worked better than basic GA. We obtained similar conclusions for the other two docking experiments on protein pr and hsp90 even though the performance gap between ALPS and the other GAs varies. Due to the premature convergence issue, it shows that when the number of generations reaches 16000, doubling the generations to 32000 can only help GA and LGA obtain slightly better solutions, which are still worse than the solutions obtained by ALPS using only 500 generations or 25,000 energy function evaluations.

To check whether the population size biased to the ALPS algorithm, we did another set of experiments by fixing the maximum generation number to 10,000 while varying the population sizes for the three algorithms. Results in Figure 7 showed that when the population size increases, the traditional GA has severe premature convergence problem leading to significantly worse (higher binding energy) results. Again, the ALPS algorithm gave the best result for all population sizes.

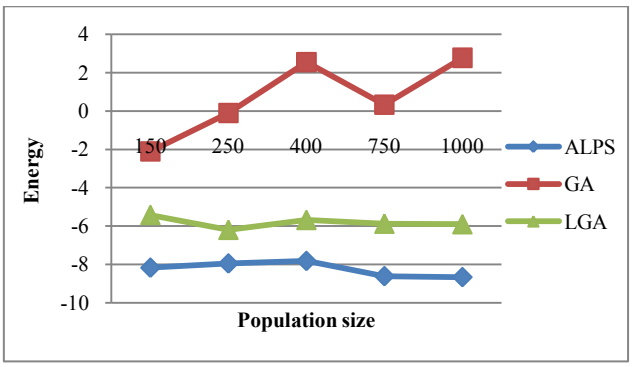

Figure 7. Docking results of three algorithms with fixed generations and varying population sizes. We tested two ligands for protein *cox*, and got the average of the lowest binding energies. We set the number of generations to 10000, and varied the population size from 150 to 1000.

C. Comparing Performance of ALPS, GA, LGA and SA

In this experiment, we compared SA (Simulated Annealing) with ALPS and the other GAs of AutoDock. Because SA ONLY works well for ligands with 8 or less torsion angles, we have chosen the ligands with at most 8 torsions to be able to compare this algorithm with others. When we choose the ligands with 9 torsions, the SA algorithm always got stuck in local minima and cannot obtain reasonable binding energy. Thus, we have only evaluated this algorithm with one ligand-protein pair.

One critical parameter of SA is the number of accept-reject steps for each temperature, which indirectly determine the total number of evaluations. Since it is not possible to predict how many accepted or rejected steps will be made at a given temperature, the number of evaluations will be different for different problems. In the past experiments [27] of SA search in Autodock, the range is between $1.19 * 10^9$ and $2.33 * 10^6$, if the accepted and rejected steps initially set to 25000. The initial temperature is $616\ cal\ mol^{-1}$. We use same termination criteria. For the other three algorithms, we set the population size to 50, and the maximum number of generations to 32000. This means that the total number of evaluations will be approximately $1.6 * 10^6$ for all three population-based search algorithms. Figure 8 shows the results of four algorithms on three different proteins. For all three proteins, sustainable ALPS achieved the lowest binding energies and simulated annealing is the worst.

D. Robustness of AutoDockX

Sustainable GA such as ALPS has a unique advantage which is their robust search performance –their search result depends much less on the starting random population and thus does not require multiple runs (e.g. 10) of GA and LGA as is usually

done by Autodock users. To show the robust docking performance, we run GA, LGA, and ALPS to dock the cox-ZINC00012342 pair each running 10 times. Table 3 shows the lowest binding energies in each result cluster after 10 runs for GA, LGA, and ALPS. A cluster is defined as the group of solutions that have a RMSD distance lower than a given threshold. Table 2 clearly indicates that GA and LGA obtained widely varying results for different runs, each run

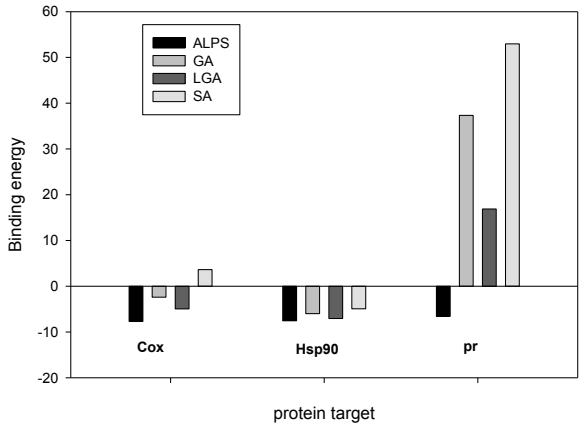

Figure 8. Comparison of binding energies of docked conformations for 4 algorithms: ALPS, GA, LGA, and SA. ALPS identified the lowest binding energy for all 3 target proteins.

generating a different cluster. And thus they all need to run multiple times to find good docking conformations. For example, one run of GA obtained a binding energy of -0.11 while another run gave 36.26.Lamarckian GA is more robust and obtained lower binding energy than GA but still much inferior to ALPS in terms of both binding energy and also the variation among the solutions of different runs. For ALPS' 10 runs, all runs generated binding energy superior to the best energy scores of both GA and LGA and the variance of these 10 runs is extremely small. This means that for ALPS, we only need to run a single docking search instead of 10 runs of traditional Autodock search algorithms to get high-quality results. The running time efficiency of ALPS is thus much better than GA or LGA due to its robust search capability.
GA and LGA generated 10 different clusters from 10 runs. ALPS generated 8 clusters with cluster 3 containing results of 3 runs with very similar conformations (only the result of the lowest energy is shown for each cluster).

Table 3. Docking results of GA, LGA, ALPS on cox-ZINC00012342 pair after 10 runs.

Cluster	Lowest Binding Energy by GA	Lowest Binding Energy by LGA	Lowest Binding Energy by ALPS
1	-0.11	-6.19	-7.95
2	3.15	-5.7	-7.77
3	10.48	-5.17	-7.36
4	13.91	-4.68	-7.29
5	16.54	-4.56	-7.23
6	18.02	-3.37	-7.14
7	25.67	-3.01	-6.98
8	28.01	-2.87	-6.75
9	32.8	-2.45	
10	36.26	0.36	

Finally, Figure 9 shows the energy ranges after 10 runs for three algorithms GA, LGA, ALPS. We calculated the mean energies and the standard deviations of 13 ligands with protein *cox.* Traditional genetic algorithms may give very different results for 10 runs. However; sustainable GA, ALPS always finds better and consistent solutions with much smaller quality variation. With AutodockX, there is no longer a need to run multiple times to get desired results.

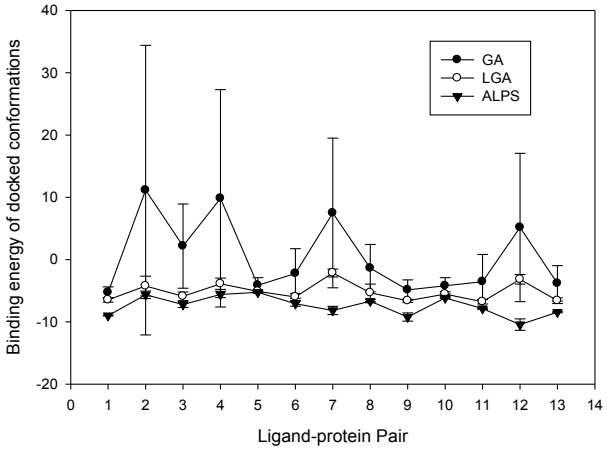

Figure 9. Binding energy variations for 10 runs of three algorithms on 13 protein-ligand pairs. The middle mark shows the mean value. GA has the largest variation among different runs and ALPS has the lowest variation or highest robustness in terms of search quality for multiple runs.

V. Conclusions

We have developed a new docking program AutoDockX by integrating the sustainable genetic algorithm ALPS to AutoDock, one of the most used tools in protein-ligand docking. We tested the docking performances over three different proteins (pr, cox and hsp90) with more than 20 candidate ligands for each protein. The results showed that our sustainable GA based AutodockX gives significantly better docking performance than all the existing search algorithms implemented in the latest version of AutoDock4. AutoDockX also has the benefits of less running time and higher robustness. A single run of AutoDockX gets better results than running traditional GA and LGA for multiple times (e.g. 10 runs). As a result, AutoDockX, has unique advantages in large-scale drug-candidate virtual screening in which millions of ligands need to be docked.

Acknowledgment

We thank Richard Porter for his help on proofreading. This research is supported by NSF CAREER AWARD DBI-8527821.

References

[1] C. Bajaj, R. Chowdhury, and V. Siddavanahalli, "F2Dock: fast Fourier protein-protein docking," *IEEE/ACM. Trans. Comput. Biol. Bioinform.*, vol. 8, no. 1, pp. 45-58, Jan.2011.

[2] A. J. Bordner and A. A. Gorin, "Protein docking using surface matching and supervised machine learning," *Proteins*, vol. 68, no. 2, pp. 488-502, Aug.2007.

[3] J. I. Garzon, J. R. Lopez-Blanco, C. Pons, J. Kovacs, R. Abagyan, J. Fernandez-Recio, and P. Chacon, "FRODOCK: a new approach for fast rotational protein-protein docking," *Bioinformatics.*, vol. 25, no. 19, pp. 2544-2551, Oct.2009.

[4] T. Geppert, E. Proschak, and G. Schneider, "Protein-protein docking by shape-complementarity and property matching," *J. Comput. Chem.*, vol. 31, no. 9, pp. 1919-1928, July2010.

[5] S. Y. Huang and X. Zou, "MDockPP: A hierarchical approach for protein-protein docking and its application to CAPRI rounds 15-19," *Proteins*, vol. 78, no. 15, pp. 3096-3103, Nov.2010.

[6] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson, "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility," *J. Comput. Chem.*, vol. 30, no. 16, pp. 2785-2791, Dec.2009.

[7] E. Noy and A. Goldblum, "Flexible protein-protein docking based on Best-First search algorithm," *J. Comput. Chem.*, vol. 31, no. 9, pp. 1929-1943, July2010.

[8] C. D. Rosin, R. S. Halliday, H. Packard, W. E. Hart, and R. K. Belew, "A comparison of global and local search methods in drug docking," Morgan Kaufmann Publishers, Inc, 1997, pp. 221-228.

[9] A. Solernou and J. Fernandez-Recio, "Protein docking by Rotation-Based Uniform Sampling (RotBUS) with fast computing of intermolecular contact distance and residue desolvation," *BMC. Bioinformatics.*, vol. 11, p. 352, 2010.

[10] C. Pons, S. Grosdidier, A. Solernou, L. Perez-Cano, and J. Fernandez-Recio, "Present and future challenges and limitations in protein-protein docking," *Proteins*, vol. 78, no. 1, pp. 95-108, Jan.2010.

[11] D. W. Ritchie and V. Venkatraman, "Ultra-fast FFT protein docking on graphics processors," *Bioinformatics*, vol. 26, no. 19, pp. 2398-2405, Oct.2010.

[12] D. W. Ritchie, D. Kozakov, and S. Vajda, "Accelerating and focusing protein-protein docking correlations using multi-dimensional rotational FFT generating functions," *Bioinformatics*, vol. 24, no. 17, pp. 1865-1873, Sept.2008.

[13] D. Kozakov, R. Brenke, S. R. Comeau, and S. Vajda, "PIPER: an FFT-based protein docking program with pairwise potentials," *Proteins*, vol. 65, no. 2, pp. 392-406, Nov.2006.

[14] E. Sakk, "On the computation of molecular surface correlations for protein docking using fourier techniques," *J. Bioinform. Comput. Biol.*, vol. 5, no. 4, pp. 915-935, Aug.2007.

[15] A. Berchanski, B. Shapira, and M. Eisenstein, "Hydrophobic complementarity in protein-protein docking," *Proteins*, vol. 56, no. 1, pp. 130-142, July2004.

[16] V. Venkatraman, Y. D. Yang, L. Sael, and D. Kihara, "Protein-protein docking using region-based 3D Zernike descriptors," *BMC Bioinformatics*, vol. 10, p. 407, 2009.

[17] X. Gong, P. Wang, F. Yang, S. Chang, B. Liu, H. He, L. Cao, X. Xu, C. Li, W. Chen, and C. Wang, "Protein-protein docking with binding site patch prediction and network-based terms enhanced combinatorial scoring," *Proteins*, vol. 78, no. 15, pp. 3150-3155, Nov.2010.

[18] D. V. Ravikant and R. Elber, "PIE-efficient filters and coarse grained potentials for unbound protein-protein docking," *Proteins*, vol. 78, no. 2, pp. 400-419, Feb.2010.

[19] D. Tobi, "Designing coarse grained- and atom based-potentials for protein-protein docking," *BMC Struct. Biol.*, vol. 10, no. 1, p. 40, Nov.2010.

[20] G. Y. Chuang, D. Kozakov, R. Brenke, S. R. Comeau, and S. Vajda, "DARS (Decoys As the Reference State) potentials for protein-protein docking," *Biophys. J.*, vol. 95, no. 9, pp. 4217-4227, Nov.2008.

[21] I. C. Paschalidis, Y. Shen, P. Vakili, and S. Vajda, "Protein-protein docking with reduced potentials by exploiting multi-dimensional energy funnels," *Conf. Proc. IEEE Eng Med. Biol. Soc.*, vol. 1, pp. 5330-5333, 2006.

[22] D. Tobi and I. Bahar, "Optimal design of protein docking potentials: efficiency and limitations," *Proteins*, vol. 62, no. 4, pp. 970-981, Mar.2006.

[23] D. K. S. Chao Lin Chu, "Spider Search: An Efficient and Non-Frontier-Based Real-Time Search Algorithm," *International Journal of Computer Information Systems and Industrial Management*, vol. 1, no. 1, pp. 234-242, 2009.

[24] P. S. A. S. Ruhaidah Samsudin, "Combination of Forecasting Using Modified GMDH and Genetic Algorithm," *International Journal of Computer Information Systems and Industrial Management*, vol. 1, no. 1, pp. 170-176, 2009.

[25] M. V. N. K. P. V. R. E.V.Gopal, "Fast and Accurate Watermark Retrieval Using Evolutionary Algorithms," *International Journal of Computer Information Systems and Industrial Management*, vol. 1, no. 1, pp. 121-136, 2009.

[26] C. S. V. G.Jeyakumar, "An Empirical Performance Analysis of Differential Evolution Variants on Unconstrained Global Optimization Problems," *International Journal of Computer Information Systems and Industrial Management*, vol. 1, no. 1, pp. 77-86, 2009.

[27] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function," *J. Comput. Chem.*, vol. 19, no. 14, pp. 1639-1662, Jan.1998.

[28] G. S. Hornby, "Alps: The age-layered population structure for reducing the problem of premature convergence.,". M. Cattolico, Ed. ACM, 2006, pp. 815-822.

[29] J. Hu, E. Goodman, K. Seo, Z. Fan, and R. Rosenberg, "The hierarchical fair competition (HFC) framework for sustainable evolutionary algorithms," *Evol. Comput.*, vol. 13, no. 2, pp. 241-277, 2005.

[30] G. S. Hornby, "Steady-State Alps for real-valued problems,". F. Rothlauf, Ed. ACM Press, 2009, pp. 795-802.

[31] G. S. Hornby, "A Steady-State Version of Age Layered Population Structure EA," in *Genetic Programming Theory and Practice VII*. R. Riolo, U. O'Reilly, and T. cConaghy, Eds. Springer, 2009, pp. 87-102.

[32] J. Hu, E. D. Goodman, K. Seo, and M. Pei, "Adaptive hierarchical fair competition (AHFC) model for parallel evolutionary algorithms,". M. Kaufmann, Ed. 2002, pp. 772-779.

[33] J. Hu and E. D. Goodman, "The Hierarchical Fair Competition (HFC) Model for Parallel Evolutionary Algorithms," IEEE Press, 2002, pp. 49-54.

[34] R. D. Taylor, P. J. Jewsbury, and J. W. Essex, "A review of protein-small molecule docking methods," *J. Comput. Aided Mol. Des*, vol. 16, no. 3, pp. 151-166, Mar.2002.

[35] M. L. Teodoro, G. N. Philips, and L. E. Kavraki, "Molecular docking: A problem with thousands of degrees of freedom,", 1 ed IEEE Press, 2001, pp. 960-966.

[36] D. S. Goodsell, G. M. Morris, and A. J. Olson, "Automated docking of flexible ligands: applications of AutoDock," *J. Mol. Recognit.*, vol. 9, no. 1, pp. 1-5, Jan.1996.

[37] X. Jiang, K. Kumar, X. Hu, A. Wallqvist, and J. Reifman, "DOVIS 2.0: an efficient and easy to use parallel virtual screening tool based on AutoDock 4.0," *Chem. Cent. J.*, vol. 2, p. 18, 2008.

[38] S. Zhang, K. Kumar, X. Jiang, A. Wallqvist, and J. Reifman, "DOVIS: an implementation for high-throughput virtual screening using AutoDock," *BMC. Bioinformatics.*, vol. 9, p. 126, 2008.

[39] S. Cosconati, S. Forli, A. L. Perryman, R. Harris, D. S. Goodsell, and A. J. Olson, "Virtual screening with AutoDock: theory and practice," *Expert Opin. Drug Discov.*, vol. 5, no. 6, pp. 597-607, Apr.2010.

[40] L. E. Kavraki, "Protein-Ligand Docking, Including Flexible Receptor- Flexible Ligand Docking," Connexions module, 2007.

[41] D. S. Goodsell and A. J. Olson, "Automated docking of substrates to proteins by simulated annealing," *Proteins*, vol. 8, no. 3, pp. 195-202, 1990.

[42] G. M. Morris, D. S. Goodsell, R. Huey, and A. J. Olson, "Distributed automated docking of flexible ligands to proteins: parallel applications of AutoDock 2.4," *J. Comput. Aided Mol. Des*, vol. 10, no. 4, pp. 293-304, Aug.1996.

[43] O. Trott and A. J. Olson, "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *J. Comput. Chem.*, vol. 31, no. 2, pp. 455-461, Jan.2010.

Author Biographies

Emrah Atilgan is a Ph.D. student in Computer Science at University of South Carolina. He was graduated from Eskisehir Osmangazi University in Turkey with B.S. in Mathematics. He received his M.S degree in Computer Science from University of South Carolina. His research interests include high-performance computing, parallel computing, data mining and bioinformatics.

Jianjun Hu is assistant professor in the Department of Computer Science and Engineering, University of South Carolina. He received the B.S. and the M.S. degrees in mechanical engineering, in 1995 and 1998, respectively, from Wuhan University of Technology, China. He received the Ph.D. in Computer Science in 2004 from Michigan State University. He was postdoctoral fellow of bioinformatics at Purdue University and University of Southern California. Dr. Hu is a member of ACM and IEEE. His current research interests include evolutionary computation, machine learning, and data mining as well as their application in engineering design and bioinformatics.