Efficient Protein-Ligand Docking Using Sustainable Evolutionary Algorithms

Emrah Atilgan
Machine Learning and Evolution Laboratory
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC, USA
atilgan@email.sc.edu

Jianjun Hu
Machine Learning and Evolution Laboratory
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC, USA
Jianjunh@cse.sc.edu

***Abstract*— AutoDock is a widely used automated protein docking program in structure-based drug-design. Different search algorithms such as simulated annealing, traditional genetic algorithm (GA) and Lamarckian genetic algorithm (LGA) are implemented in AutoDock. However, the docking performance of these algorithms is still limited by the local optima issue of simulated annealing or the premature convergence issue typical in 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, Age-Layered Population Structure (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- Protein docking, Autodock, 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. One of the most widely used automated docking programs is AutoDock, which predicts how small molecules bind to a receptor of known 3D structure [1]. 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 and report the best results. To overcome these limitations of AutoDock and get better docking performance, we proposed to apply sustainable evolutionary algorithms [2], a new type of robust genetic algorithms, to protein docking. We integrated the ALPS [3, 4, 5], one of the sustainable evolutionary algorithms into AutoDock and compared its performance with those of current algorithms using the lowest binding energy and computational time criteria. The main advantage of sustainable evolutionary algorithms is their capability to address the premature convergence problem typical in traditional genetic algorithms [6, 7]: 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 [18], there are more than 50 protein-ligand docking softwares, most of which still used traditional GAs for conformation search optimization. Our experiments here implied that other modern protein-ligand docking programs can also be potentially improved by the sustainable genetic algorithms.

II. BACKGROUND

A. Proten-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 [8-10].

Protein-ligand docking algorithms can be classified into two methods. In early docking algorithms, both the protein and the ligand are considered as rigid bodies and they have only six degrees of translational and rotational freedom to search for best orientations. Most of the 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). Consideration of the ligand flexibility leads to more degrees of freedom to search. 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.

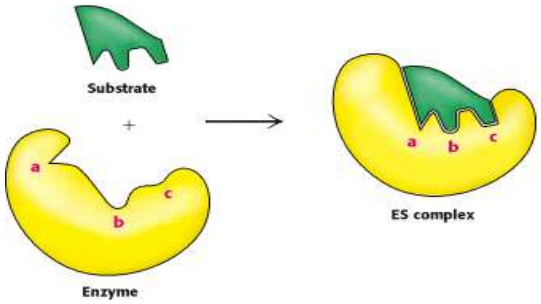

Figure 1: An example of protein-ligand docking

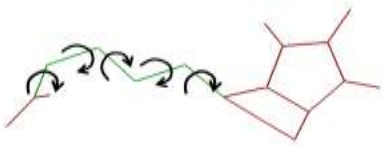

Figure 2: A ligand with rotatable bonds (torsions).

B. Search Algorithnms 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 in virtual screening [11, 12] for predicting how small molecules bind to a receptor of known 3D structure. In AutoDock [16], 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. 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 $(n_1, n_2, ...n_m)$, depending on how many rotatable bonds the ligand has [17]. 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 [1].

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 was used as the major optimization method [14, 15]. 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 [15]. 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 intramolecular energy of the ligand. 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 [16]. However, it does not guarantee to find the global minimum conformation [15].

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 [1]. 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 [1]. 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.

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

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 [1].

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. Even though SA performs well with the ligands that have roughly 8 rotatable bonds or less, the algorithm becomes ineffective with more than 8 rotatable bonds [1].

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 increase the best fitness of the population. This problem is called *premature convergence* problem [3, 7]. 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 [7].

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) [2] has been shown to be much more efficient than the multi-run method and can achieve significantly better solutions.

III. SUSTAINABLE GENETIC ALGORITHM FOR DOCKING: AUTODOCKX

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 the capability to make constant progress given more computation resource unless the optimum is reached.

The first such algorithm is the HFC algorithm framework. The main idea is to keep 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 thus achieves sustainable searching by ensuring continuous supply and the incorporation of individuals in the hierarchical levels. Another form of sustainable evolutionary algorithm, ALPS uses a parameter 'age' instead of fitness value, to set up the hierarchy levels.

B. ALPS – Age Layered Population Structure

ALPS is another recently proposed sustainable GA [3], which was shown to have good performance on GA problems. ALPS 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 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 [3].

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 3). 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.

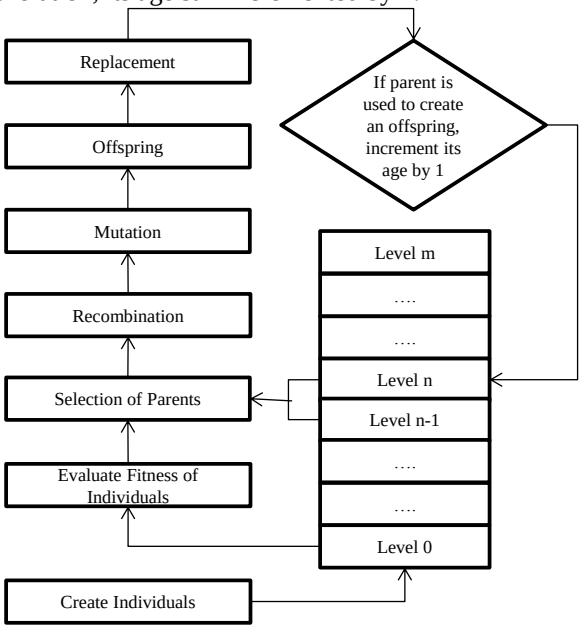

Figure 3: ALPS algorithm in AutoDock

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. This allows younger individuals to be able to find and move into a good basin of attraction before they are pushed into next layer [3].

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 1.

Table 1. Default ALPS parameters

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

IV. EXPERIMENTAL RESULTS

A. *Test Data Prepration*

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 show that sustainable evolutionary algorithms such as ALPS can help to address the premature convergence problem of traditional GAs, we compared the performance of three search algorithms (ALPS, GA, 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 4 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. Since the best binding energy of the 22 tested protein-ligand pairs are different, the calculation of standard deviations of the methods is omitted here. 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.

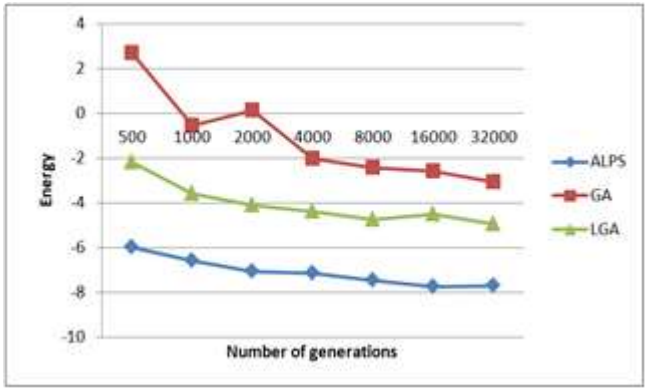

Figure 4: Overall results of different algorithms on protein cox using fixed population size 50 and varying maximum number of generations. Each algorithm was run 10 times for each protein-ligand pair. The averages of the lowest binding energy for all protein-ligand pairs are then calculated.

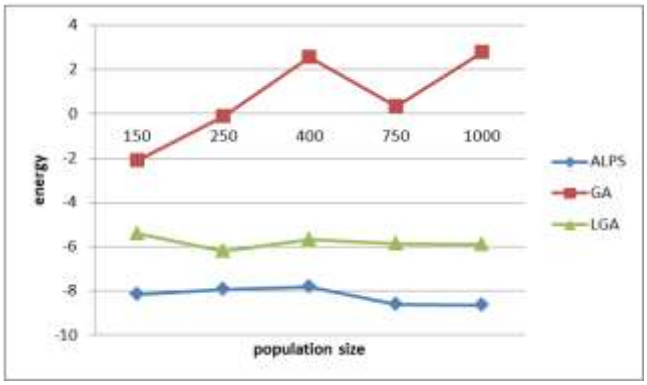

Figure 5: Docking results of three algorithms with fixed generations and varying population sizes. We tested two ligands for protein *cox*, and get the average of the lowest binding energies. We set the number of generations to 10000, and varied the population size from 150 to 1000.

To check whether the population size biased to the ALPS agorithm, 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 5 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. Comparing performance of ALPS, GA, LGA to 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.

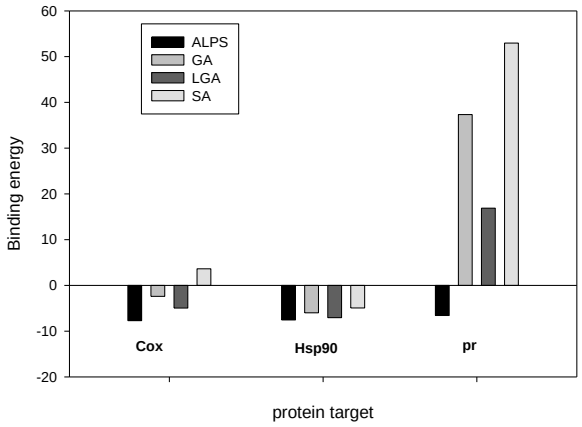

Figure 6: Comparison of binding energies of docked conformations for 4 algorithms: ALPS, GA, LGA, and SA. **ALPS identified the lowest binding energy for all 3 target proteins.**

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 [1] of SA search in Autodock, the range is between $1.19 * 10^5$ 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 6 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.

C. 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 2 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 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 (by saving 9 runs) is thus much better than GA or LGA due to its robust search capability.

Table 2: Docking results of GA, LGA, ALPS on cox-ZINC00012342 pair after 10 runs. GA and LGA generated 10 different clusters from 10 runs. ALPS generated 8 clusters with cluster 3 containing the results of 3 runs with very similar conformations (only the result of the lowest energy is shown for each cluster).

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 7 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.

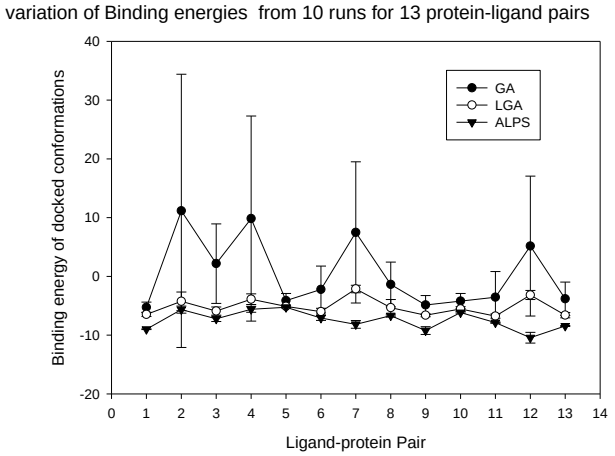

Figure 7: Binding energy variations for 10 runs of three algorithms on 13 protein-ligand pairs. The middle mark shows the mean value. GA has the largest variation among different runs and ALPS has the lowest variation or highest robustness in terms of search quality for multiple runs.

V. CONCLUSIONS

We have developed a new docking program AutoDockX by applying a 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.

VI. ACKNOWLEDGMENTS

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

VII. REFERENCES

[1] Morris, G. M., D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R.K. Belew, and A. J. Olson. 1998. Automated docking using a Lamarckian genetic algorithm and empirical binding free energy function. *Journal of Computational Chemistry* 19 (14): 1639-1662.

[2] Hu, J., E. Goodman, K. Seo, Z. Fan, and R. Rosenberg. 2005. The hierarchical fair competition (HFC) framework for sustainable evolutionary algorithms. *Evolutionary Computation* 13(2): 241-277.

[3] Hornby, G. S. Alps: The age-layered population structure for reducing the problem of premature convergence. *Gecco 2006: Proceedings of the 8th annual conference on genetic and evolutionary computation*. In M. K. et al., editor, *Proc. of the Genetic and Evolutionary Computation Conference, GECCO-2006*, pp. 815-822, Seattle, WA, 2006. ACM Press.

[4] Hornby, G. S. A Steady-State Version of Age Layered Population Structure EA. *Genetic Programming Theory and Practice VII*, ch. 6: 87-102, Ann Arbor 2009. *Springer*.

[5] Hornby, G. S. Steady-state Alps for real-valued problems. *Proceedings of the 11th annual conference on genetic and evolutionary computation*. 2009, pp.795-802, Montreal, Quebec, Canada, 2009. ACM Press.

[6] J. Hu, E. D. Goodman, K. Seo, and M. Pei. Adaptive hierarchical fair competition AHFC model for parallel evolutionary algorithms. In Proc. of the Genetic and Evolutionary Computation Conference, pages 772-779. Morgan Kaufmann, 2002.

[7] J. Hu and E. D. Goodman. The hierarchical fair competition HFC model for parallel evolutionary algorithms. In Proc. of the 2002 Congress on Evolutionary computation, pages 49-54. IEEE Press, 2002.

[8] Taylor, R.D., P. J. Jewsbury, and J. W. Essex. 2002. A review of protein-small molecule docking methods. *J Comput-Aided Mol Des* 16 (3): 151–166.

[9] Trott, O. and A. J. Olson. 2009. Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. Journal of Computational Chemistry 9999 (9999): NA+.

[10] Teodoro, M. L., G. N. Phillips, and L. E. Kavraki. 2001. Molecular docking: A problem with thousands of degrees of freedom. *IEEE International Conference on Robotics and Automation*. IEEE Press, Seoul, Korea. Vol. 2001, pp.960-966.

[11] Zhang, S., K. Kumar, X. Jiang, A. Wallqwist, and J. Reifman. 2008. Dovis: an implementation for high-throughput virtual screening using autodock. *BMC Bioinformatics* 9 (1).

[12] Jiang, X., K. Kumar, X. Hu, A. Wallqvist, and J. Reifman. 2008. Dovis 2.0: an efficient and easy to use parallel virtual screening tool based on autodock 4.0. *Chemistry Central Journal* 2 (1): 18.

[13] Goodsell, D. S., G. M. Morris, and A. J. Olson. 1996. Automated docking of flexible ligands: Applications of autodock. *Journal of Molecular Recognition* 9 (1): 1-5.

[14] Morris, G. M., D. S. Goodsell, R. Huey, and A. J. Olson. 1996. Distributed automated docking of flexible ligands to proteins: Parallel applications of autodock 2.4. *Journal of Computer- Aided Molecular Design* 10 (4): 293-304.

[15] Goodsell, D. S. and A. J. Olson. 1990. Automated docking of substrates to proteins by simulated annealing. *Proteins: Structure, Function and Genetics* 8: 195-202.

[16] Kavraki, Lydia. "Protein-Ligand Docking, Including Flexible Receptor- Flexible Ligand Docking. "Connexions. March22, 2007. http://cnx.org/content/m11456/1.10/

[17] Rosin, C., R. S. Halliday, H. Packard, W. E. Hart, and R. K. Belew. A comparison of global and local search methods in drug docking. *In proceeding of seventh international conference on genetic algorithms*. 1997, pp. 221-228. San Francisco, CA. Morgan Kaufmann Publishers, Inc.

[18] Moitessier N, Englebienne P, Lee D, Lawandi J, Corbeil CR.Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go.Br J Pharmacol. 2008 Mar;153 Suppl 1:S7-26