SODOCK: Swarm Optimization for Highly Flexible Protein–Ligand Docking

HUNG-MING CHEN,[1] BO-FU LIU,[1] HUI-LING HUANG,[2] SHIOW-FEN HWANG,[1] SHINN-YING HO[3,4]
[1]*Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan*
[2]*Department of Information Management, Jin Wen Institute of Technology, Taipei, Taiwan*
[3]*Department of Biological Science and Technology, National Chiao Tung University, Hsinchu, Taiwan*
[4]*Institute of Bioinformatics, National Chiao Tung University, Hsinchu, Taiwan*

Received 4 February 2006; Revised 22 July 2006; Accepted 28 August 2006
DOI 10.1002/jcc.20542
Published online 21 December 2006 in Wiley InterScience (www.interscience.wiley.com).

Abstract: Protein–ligand docking can be formulated as a parameter optimization problem associated with an accurate scoring function, which aims to identify the translation, orientation, and conformation of a docked ligand with the lowest energy. The parameter optimization problem for highly flexible ligands with many rotatable bonds is more difficult than that for less flexible ligands using genetic algorithm (GA)-based approaches, due to the large numbers of parameters and high correlations among these parameters. This investigation presents a novel optimization algorithm SODOCK based on particle swarm optimization (PSO) for solving flexible protein–ligand docking problems. To improve efficiency and robustness of PSO, an efficient local search strategy is incorporated into SODOCK. The implementation of SODOCK adopts the environment and energy function of AutoDock 3.05. Computer simulation results reveal that SODOCK is superior to the Lamarckian genetic algorithm (LGA) of AutoDock, in terms of convergence performance, robustness, and obtained energy, especially for highly flexible ligands. The results also reveal that PSO is more suitable than the conventional GA in dealing with flexible docking problems with high correlations among parameters. This investigation also compared SODOCK with four state-of-the-art docking methods, namely GOLD 1.2, DOCK 4.0, FlexX 1.8, and LGA of AutoDock 3.05. SODOCK obtained the smallest RMSD in 19 of 37 cases. The average 2.29 Å of the 37 RMSD values of SODOCK was better than those of other docking programs, which were all above 3.0 Å.

 J Comput Chem 28: 612–623, 2007

Key words: flexible docking; scoring function; genetic algorithm; particle swarm optimization; local search

Introduction

Automated protein–ligand docking is an effective tool for structure-based drug design. Protein–ligand docking identifies the translation, orientation, and conformation of a ligand relative to the active site of a target protein. Virtual screening by molecular docking has become a conventional means of improving efficiency in lead finding.[1] Earlier docking methods that treat both ligands and proteins as rigid objects are called *rigid docking*.[2] Docking methods that consider a ligand as an articulated object, and a protein as a rigid object are named *flexible docking*.[3–5] Because most ligands are not rigid, flexible docking is the most conventionally adopted method. Other methods include *soft docking*[6–9] and *partially flexible protein docking*,[4–6,9] which attempt to reflect the side chain flexibility of proteins. Modeling protein flexibility, along with scoring function development, is a rapidly growing aspect of docking.

Protein–ligand docking can be formulated as a parameter optimization problem associated with an accurate scoring function, which aims to identify the docked conformation of a ligand with the lowest energy. An efficient docking method comprises two fundamental components, a good scoring function and an efficient optimization algorithm. The scoring function is an approximation of the free energy of binding interaction between protein and ligands. The native structure of the protein–ligand complex determined from X-ray crystallography is thought to produce the lowest docked energy. An accurate scoring function should reflect this observation. But due to the approximation to the free energy, the native structure may not always have the lowest energy using

Correspondence to: S.-Y. Ho; e-mail: syho@mail.nctu.edu.tw

Contract/grant sponsor: National Science Council of the Republic of China, Taiwan; contract/grant number: NSC-94-2213-E-009-142.

the existing scoring functions. Wang et al. addressed the issue of scoring functions for flexible protein–ligand docking.[10]

The optimization algorithm for solving the optimization problem of flexible docking aims to identify the docked conformation with the lowest energy. The performance of the optimization algorithms can be evaluated in terms of docked energy. Various optimization algorithms, including simulated annealing (SA)[11] and genetic algorithms (GAs),[3,4] have been presented for solving protein–ligand docking problems. The free energy landscapes of scoring functions are usually complex, and exhibit a rugged funnel shape.[12] Hence, an efficient optimization algorithm that can quickly obtain a potentially good approximation to the globally optimal solution of the scoring function is desirable.

The desirable docking method attempts to identify the docked conformation with the smallest root-mean-square deviation (RMSD) between atomic positions in the docked ligand and the corresponding position in crystal-structure ligand. A docked conformation with a smaller RMSD is considered as a more accurate solution to the docking problem. Since a newly created drug-like ligand lacks a crystal structure, RMSD cannot be adopted as an objective function in the optimization process. Therefore, optimization algorithms have to adopt the scoring function based on the docked energy.

The parameter optimization problem for highly flexible ligands with many rotatable bonds is more difficult than that for less flexible ligands using GA-based approaches, due to the large numbers of parameters and high correlations among these parameters. This investigation presents a novel optimization algorithm SODOCK for obtaining the docked conformation of a ligand with the best score for solving highly flexible docking problems. The proposed algorithm is a hybrid of particle swarm optimization (PSO)[13] and a local search. PSO is a population-based search algorithm inspired by social behaviors of organisms, such as the flocking of birds. PSO is simpler and quicker to converge than GAs. Recent studies have revealed that PSO is a robust and efficient optimization algorithm for solving continuous nonlinear optimization problems.[14,15] To improve PSO, an efficient variant of the Solis and Wets local search technique[16] is incorporated into SODOCK such that both techniques cooperate fully with each other. Simulation results reveal that SODOCK has a better convergence and a lower docked energy than PSO alone.

The implementation of SODOCK adopts the environment and energy function of AutoDock 3.05.[4] Computer simulation results reveal that SODOCK is superior to the Lamarckian genetic algorithm (LGA) of AutoDock, in terms of convergence performance, robustness, and obtained energy, especially for highly flexible ligands. This investigation also compared SODOCK with four state-of-the-art docking methods which can be obtained from public packages, including GOLD 1.2,[3] DOCK 4.0,[2] FlexX 1.8,[5] and LGA of AutoDock 3.05.[4] Simulation results reveal that SODOCK can yield more accurate results than the other methods in terms of RMSD.

Problem Definition

The investigated flexible protein–ligand docking problem was formulated as a parameter optimization problem using the same scoring function as AutoDock 3.05.[4] The proposed optimization algorithm attempts to identify the translation, orientation, and conformation of a ligand with the best score, i.e., the lowest energy.

Representation

In AutoDock, the translation, orientation, and conformation of a docked ligand are denoted by the following parameters.[4]

1. Three parameters t_x, t_y, and t_z denote the translation of the ligand relative to the center of a specified 3D grid box. The box should be sufficiently large to enclose the binding site of protein.
2. The orientation of a ligand is represented by a quaternion, which is defined by four parameters, n_x, n_y, n_z, and α. The three parameters n_x, n_y, $n_z \in [0,1]$ denote a vector specifying an axis that the ligand rotates along with, and $\alpha \in [-\pi,\pi]$ denotes the rotation angle about this axis. Although the orientation of a ligand can be represented by three Euler angles, using the quaternion can prevent the gimbal lock problem experienced with Euler angles.[4]
3. Angles $\text{tor}_i \in [-\pi,\pi]$ is associated with rotatable bond i of the ligand, $i = 1, 2, \ldots, T$ where T denotes the number of rotatable bonds.

Therefore, a docked conformation needs to optimize $N = 7 + T$ parameters. The N parameters are encoded into individuals, and optimized by GA and SODOCK according to the energy function described in the next section.

Scoring Function

AutoDock 3.05 adopts an empirical energy function as a scoring function to evaluate a docked conformation. The total docked energy of a candidate solution X is expressed as the sum of intermolecular interactions of the protein–ligand complex and the internal energy of the ligand:

$$\min E_{\text{total}}(X) = E_{\text{vdw}} + E_{\text{H-bond}} + E_{\text{elec}} + E_{\text{internal}} + E_{\text{desolvation}}. \quad (1)$$

The first three terms are intermolecular energies, which are van der Waals force, hydrogen bonding, and electrostatic potential. The term E_{internal} denotes the internal energy of the ligand, which also comprises the first three forces. The last term models the desolvation upon binding and the hydrophobic effect. Morris et al. described the energy function in further detail.[4]

Methods

Particle Swarm Optimization

PSO is a general-purpose search algorithm that exploits a population of individuals to probe promising regions in the search space. In this context, the population is called a *swarm*, and the individuals are called *particles*. Each particle moves with an adaptable velocity within the search space, and retains in its memory the best

position that it has encountered. The standard version of PSO is briefly described later.[17–19]

Assume an N-dimensional search space, $S \subset R^N$, and a swarm comprising N_p particles. The position $X = [x_1, \ldots, x_N] = [t_x, t_y, t_z, n_x, n_y, n_z, \alpha, \text{tor}_1, \ldots, \text{tor}_T]$ of a particle in the search space S denotes a candidate solution to the investigated docking problem, where T is the number of rotatable bonds in the ligand. The current position of particle i is an N-dimensional vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{iN}]^t \in S$. The velocity of this particle is also an N-dimensional vector $V_i = [v_{i1}, v_{i2}, \ldots, v_{iN}]^t \in S$, which indicates the displacement for updating the position of each particle in the search space. The best position encountered by particle i is denoted as $P_i = [p_{i1}, p_{i2}, \ldots, p_{iN}]^t \in S$. Assume that g is the index of the particle that attained the best position found by all particles in the neighborhood of particle i. The swarm is manipulated by the following equations:[17–19]

$$v_{id}(t+1) = wv_{id}(t) + c_1 rand()(p_{id}(t) - x_{id}(t)) + c_2 rand()(p_{gd}(t) - x_{id}(t)), \quad (2)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (3)$$

where $i = 1, 2, \ldots, N_p$ is the particle's index; $d = 1, 2, \ldots, N$ denotes the dimension index; and $t = 1, 2, \ldots$, denotes the iteration number. The variable w is a parameter called inertia weight, which balances global and local searches in the PSO. The two positive constants c_1 and c_2 are cognitive and social parameters, respectively. Proper fine-tuning of c_1 and c_2 may improve the performance of the PSO. An extended study of the cognitive and social parameters was proposed by Kennedy,[20] and $c_1 = c_2 = 2$ were recommended as default values. The function *rand*() generates a random number uniformly distributed within the interval [0,1].

Overlapped neighborhoods are classified into two types, named *gbest* and *lbest*.[21] In a *gbest* neighborhood, each particle's neighborhood includes all the particles in the swarm. In an *lbest* neighborhood, each particle is affected by the best positions of its K immediate neighbors in the ring of topological population. For instance, the *lbest* neighborhood of Particle$_i$ with $K = 2$ are Particle$_{i-1}$, Particle$_i$, and Particle$_{i+1}$. PSO with the *gbest* neighborhood tends to converge more rapidly than PSO with the *lbest* neighborhood, but also converges more easily to a local minimum.[22]

Clerc and Kennedy[23] recently indicated that a *constriction factor* χ may help ensure convergence. The new velocity of a particle is derived by the following equation:

$$v_{id}(t+1) = \chi[v_{id}(t) + d_1 rand()(p_{id}(t) - x_{id}(t)) + d_2 rand()(p_{gd}(t) - x_{id}(t))], \quad (4)$$

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \quad (5)$$

where $\phi = d_1 + d_2 > 4$. A setting of $d_1 = d_2 = 2.05$ is typically used, making the above equation equivalent to eq. (2) with $w = 0.72984$ and $c_1 = c_2 = 1.4962$.

To prevent the velocity of a particle from increasing uncontrollably, a maximum velocity V_{max} is defined. The velocity is prevented from exceeding V_{max} on each dimension d for particle i as follows:

$$\text{If } v_{id} > V_{max} \quad \text{then} \quad v_{id} = V_{max}$$

$$\text{else if } v_{id} < -\vee_{max} \quad \text{then} \quad \iota\delta = -\vee_{max}.$$

The swarm and velocities are randomly initialized in the search space, although more sophisticated initialization approaches can improve the overall performance of PSO.[24] The initial velocities are often distributed randomly over $[-V_{max}, V_{max}]$. Eberhart and Shi[25] have recommended restricting the maximal velocity V_{max} to the upper value of the dynamic search range.

Local Search

If the position of a particle is not a local optimum, a local search operation has a good chance to find a better solution in its neighborhood. Local search is a commonly adopted technique to improve the search ability of GA. AutoDock adopts a LGA that is a hybrid of GA and a variant of Solis and Wets local search,[16] and is illustrated in Figure 1. One major advantage of this local search is that it does not need gradient information about the local energy landscape to facilitate the search operation.[4] The performance of the LGA has been shown to be superior to that of both GA and SA.[4]

PSO using an additional local search strategy can also improve the search performance in a limited amount of computation time. Moreover, recent studies[26–28] also indicate that maintaining diversity of particles can improve the performance of PSO. An effective approach is to incorporate a mutation operator into PSO.[27,29] The local search modifies the position parameters of a particle without referring to other particles, and also maintains the diversity among particles. Hence, the behavior and effect of the local search in SODOCK are similar to those of a mutation operator. The simulation result reveals that the average Euclidean distance among particles in the search space using SODOCK is greater than that using PSO alone. Therefore, the local search of SODOCK is beneficial to maintain diversity of particles and prevent premature convergence.

Hybrid Search of SODOCK

SODOCK is a hybrid of the PSO and Solis and Wets local search, and is designed for flexible docking. The parameters of the docked conformations are encoded into particles and optimized by SODOCK. The performance of both *gbest* and *lbest* type neighborhoods in PSO was evaluated with different handling approaches of inertia weight. The *lbest* type neighborhood with linear decreasing of inertia weight[18,19] from $w_{max} = 0.9$ to $w_{min} = 0.4$ was found to perform best when it cooperated with the local search. The constriction factor was not adopted, because it resulted in premature convergence in the test cases. Figure 2 illustrates the SODOCK algorithm. The population is evolved over iterations until a specified maximal number neval$_{max}$ of function evaluations is attained. An iteration of SODOCK comprises four steps: updating velocities, moving particles, applying local search, and updating previous best positions of particles. The local search is only applied to

```
Procedure SWLocalSearch(X, ρ, succ_max, fail_max, iter_max)
b = 0; /* bias vector */
iter = 0;
succ = 0;
fail = 0;
while iter < iter_max and ρ is not too small do
    for each dimension i do
        δ_i = nrand(b_i, ρ_i);
        /* a normal deviate with mean b_i and
           standard deviation ρ_i */
    end for
    X' = X + D; /* D = [δ_1, δ_2, ...] */
    if X' is better than X then
        X = X';
        succ = succ+1;
        fail = 0;
        b = 0.4b + 0.2D;
    else
        X' = X − D;
        if X' is better than X then
            X = X';
            succ = succ+1;
            fail = 0;
            b = b − 0.4D;
        else
            fail = fail+1;
            succ = 0;
            b = 0.5b;
        end if
    end if
    if succ = succ_max then
        ρ = 2ρ;
        succ = 0;
    end if
    if fail = fail_max then
        ρ = 0.5ρ;
        fail = 0;
    end if
    iter = iter+1;
end while
```

Figure 1. Procedure of the Solis and Wets local search.

```
Algorithm SODOCK(N_p, w_max, w_min, neval_max)
neval = 0;
t = 0;
for i = 1 to N_p do
    randomly initialize X_i(t) and V_i(t);
    evaluate docking energy of X_i(t);
    P_i(t) = X_i(t);
end for
while neval < neval_max do
    w = (w_max − w_min)( neval / neval_max) + w_min;
    for i = 1 to N_p do
        find P_gi(t) among the neighborhood of X_i(t);
        calculate new velocity V_i(t+1) using (2) and (3);
        fix V_i(t + 1) if it is out of range;
        X_i (t + 1) = X_i(t) + V_i(t + 1);
        evaluate docking energy of X_i(t + 1) using (1);
    end for
    apply the Solis and Wets local search to the best particle among all X_i(t + 1);
    for i = 1 to N_p do
        if X_i (t + 1) better than P_i(t) then
            P_i(t + 1) = X_i (t + 1);
        else
            P_i (t + 1) = P_i(t);
        end if
    end for
    t = t + 1;
end while
return the best position among all P_i(t);
```

Figure 2. The SODOCK algorithm.

the best (lowest energy) particle. A prespecified maximal number (e.g., $\text{iter}_{\max} = 50$) of steps for each local search is adopted to balance the global and local searches during the search process. The global best position among all P_i is the final docking result of SODOCK.

The hybrid search is efficient for the optimization problem of flexible docking, because the energy function landscape usually contains many local minima. The performance of the global or local search alone is not satisfactory for solving highly flexible docking problems. PSO has a good global search ability to obtain a potentially good approximation to a global minimum. However, the final positions of particles derived from eq. (2) are usually not local minima. Using the local search operation alone, each individual rapidly converges to a local minimum instead of the global or nearly global minimum. But PSO and the local search can efficiently compensate each other in SODOCK. The local search provides a highly efficient means of finding a local minimum near the current position of a particle, which can also maintain diversity of particles and prevent premature convergence. Simulation results reveal that the hybrid search of SODOCK is more efficient than PSO alone for solving highly flexible docking problems.

Implementation

AutoDock[4] is free for academic use, and has fully available source code. AutoDock is a good framework for developing novel optimization algorithms. SODOCK is implemented using C++, and adopts the environment and energy function of AutoDock 3.05. The source code of SODOCK and some supplementary information are available at our website http://iclab.life.nctu.edu.tw/sodock/. All docking simulations were performed on a PC with a Pentium 4 2.8-GHz processor running Linux operating system. Table1 summarizes the SODOCK parameter settings used in the computer simulation.

Table 1. Parameter Setting of SODOCK.

Number of particles, N_p	50
Number of immediate neighbors, K	4
Maximal number of function evaluation, $\text{neval}_{\max}$	250,000
Inertia weight, w	$0.9 \sim 0.4$ (linear decreasing)
Cognitive weight, c_1	2.0
Social weight, c_2	2.0
Maximal velocity, $V_{\max}$	2.0 Å (for t_x, t_y, and t_z) 1.0 (for n_x, n_y, and n_z) 50° (for α and tor_i)
Maximal steps of local search	50

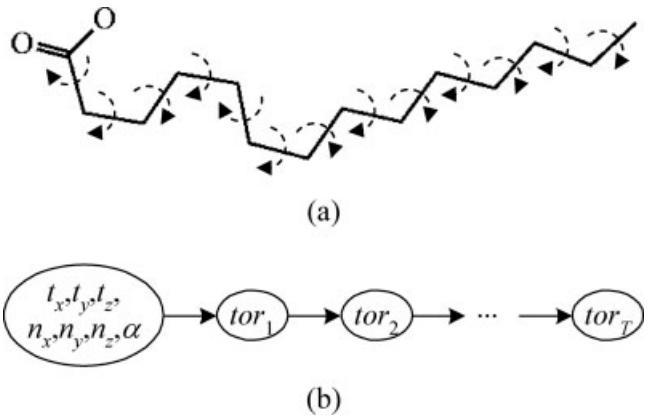

Figure 3. An example for highly flexible ligand with nested torsions. (a) Structure of ligand myr and (b) corresponding interactions among parameters.

The other parameters, such as the size and spacing of grid maps, are the same as default values of AutoDock.

Analysis

The optimization problem of highly flexible docking is more difficult than that of less flexible docking due to the strong correlations among torsion parameters. The major operators in GA and PSO for dealing with torsion parameters were analyzed to demonstrate the superiority of PSO over GA in obtaining solutions with low-docked energies for highly flexible docking problems.

Interactions Among Encoded Parameters

The core of a flexible ligand is a rigid root, which is typically a group of atoms. The other atoms are connected to the rigid root by rotatable bonds (torsions). A flexible docking problem has seven parameters for the rigid root, and T parameters for torsion angles to be optimized. Because of the interdependence among root and torsion parameters, simultaneous adjustments of these parameters can efficiently obtain a solution with low energy. Torsions of a flexible ligand may be nested, as illustrated in the example of Figure 3. Even a small refinement of a torsion near the root results in large adjustments on its following parts. This scenario reflects the strong interactions among the encoded torsion parameters, thus making the optimization problem intractable. Therefore, the ability of search operators in dealing with interactions primarily determines the docking performance of the search methods.

Crossover of GA

Interactions among parameters should be handled appropriately when encoding chromosomes of GA and designing GA operators to solve the optimization problem of flexible docking. High-quality partial solutions are known as building blocks.[30] General, fixed, problem-independent crossover operators often disrupt building blocks, possibly leading to premature convergence.[31] The problem of building block disruption is also called the linkage problem. A serious linkage problem degrades the search performance of a GA when using parameters with strong interactions.[32,33] Several GAs, including Messy GA,[34] Linkage Learning GA,[35] and Estimation of Distribution Algorithm,[36] focus on the linkage problem. However, these algorithms require a high computational cost and have not

Table 2. The Lowest Energy Results of SODOCK and LGAs.

			SODOCK		LGA-default		LGA-large		LGA-arithmetic	
PDB	Ligand	Torsions	Energy	RMSD	Energy	RMSD	Energy	RMSD	Energy	RMSD
1hiv	noa	23	−23.89	1.66	−10.89	11.30	−10.86	12.31	−10.93	9.12
1aaq	psi	20	−18.84	0.97	−9.51	6.29	−13.33	3.29	−6.98	3.00
1epo	mor	17	−16.45	1.27	−15.50	2.04	−14.03	1.78	−9.72	10.05
4phv	vac	15	−20.75	0.53	−15.54	6.39	−16.03	3.77	−14.39	4.28
1hpv	478	14	−15.72	1.30	−13.31	4.59	−13.70	3.86	−12.98	8.68
1htf	g26	13	−15.77	3.87	−15.43	2.27	−14.91	2.60	−14.72	2.59
1tmn	clt	13	−15.55	2.16	−14.76	3.05	−15.24	2.45	−12.85	1.55
1cdg	mal	12	−8.55	5.99	−8.28	4.54	−8.97	6.50	−8.49	6.12
1qbt	146	12	−24.33	1.36	−23.55	1.34	−23.49	1.45	−20.60	1.67
1dwd	mid	10	−15.41	6.64	−14.54	1.82	−14.48	6.46	−14.53	1.62
1ets	nas	10	−17.83	2.15	−16.11	0.87	−16.03	2.23	−13.83	2.05
4hmg	sia	11	−8.57	0.66	−8.15	0.79	−8.06	0.78	−7.19	0.58
1hvr	xk2	10	−21.43	0.64	−21.31	0.59	−21.26	0.51	−21.11	0.93
1stp	btn	5	−10.91	0.47	−10.73	0.52	−10.75	0.40	−10.50	0.39
2mcp	pc	4	−6.51	1.95	−6.38	2.04	−6.47	1.79	−5.80	1.84
2cpp	cam	0	−7.40	3.08	−7.39	3.07	−7.39	3.07	−7.39	3.09
3ptb	ben	0	−6.95	0.34	−6.94	0.34	−6.94	0.34	−6.94	0.34
Average			*−14.99*	*2.06*	*−12.84*	*3.05*	*−13.06*	*3.15*	*−11.70*	*3.41*
Wins			*16*	*6*	*0*	*6*	*1*	*4*	*0*	*5*

been applied to real-world applications with strong interactions, such as flexible docking.

In LGA of AutoDock, the N parameters of an individual are encoded as a string of real values. Two-point crossover is adopted where cut points are only located on the boundary between parameters. The number of candidate solutions that can be explored per crossover operation is only $2C_2^{N-1}$. The individual parameters can be modified by only mutation and local search operators. Moreover, building blocks are easily disrupted by the crossover, causing the fitness values of newly generated individuals to thrash severely, as discussed later. Thomsen proposed an arithmetic crossover operation to replace the two-point crossover of LGA for flexible docking.[37] The arithmetic crossover randomly generates two offspring from the hyperbox formed by their parents. That is, no parameter of an offspring is directly inherited from its parents. This property may be undesirable for highly flexible docking problems, because the arithmetic crossover cannot easily keep building blocks growing. These analyses are verified by the simulation results which are described later.

Move Mechanism of PSO

Unlike GA, PSO has no selection operator.[17] Hence, all particles survive during the entire evolution. They can pass through the local minima to the promising region if sufficient iterations are given. In contrast to GA, the parameter order in encoding particles does not affect the performance of PSO, which adopts a move mechanism instead of crossover as the search operator. The new position $X_i(t + 1)$ of particle i generated by the move mechanism is a linear combination of current position $X_i(t)$, current velocity $V_i(t)$, its best previous position $P_i(t)$, and the best previous position $P_g(t)$ in its neighborhood, where t denotes the iteration number. The number of candidate solutions per move operation is much larger than that of the per two-point crossover operation. Another merit of PSO is that each particle maintains its best previous position P_i. Thus, the move mechanism always guides particles to positions better than their current positions. These properties improve the search ability of PSO in dealing with strong interactions among parameters.

Results

The computer simulation comprised two parts. In Part 1, three versions of LGA and four versions of PSO were evaluated using 17 test cases, which consisted of six complexes used by Thomsen[37] and 11 complexes with highly flexible ligands.[38,39] In Part 2, the docking accuracy of SODOCK and four state-of-the-art docking methods were compared using 37 complexes. Molecule files and seed values for random number generator used in the simulation are available on our website http://iclab.life.nctu.edu.tw/sodock/.

Data Preparation

Both ligand and protein input files were saved using AutoDock's PDBQ format. The ligand input files were obtained using the following procedure: (1) extract the coordinates of ligand atoms from the PDB file; (2) add hydrogen to all atoms, assign partial charges, and merge nonpolar hydrogen atoms; and (3) define rigid root and torsions of the ligand. The preparation of proteins is as follows: (1) remove water molecules, ligands, and metal ions not belonging to the binding site; (2) repair residues that have missing atoms; (3) add hydrogen to all atoms, assign partial charges, and merge nonpolar hydrogen atoms; and (4) assign solvent parameters. This molecule file preparation stage was conducted using AutoDock Tools.

Part 1: Evaluation of Search Ability

Since optimization algorithms are guided only by energy function, their search ability can be evaluated in terms of docked energy using the same number of function evaluations. Four versions of PSO were implemented by combining two neighborhoods, *gbest* and *lbest*, and two approaches for handling inertia weight, namely *linear decreasing* and *constriction factor*. For the *lbest* neighborhood, the number of immediate neighbors was set to $K = 4$. In the linear decreasing approach, the inertia weight w was decreased linearly from 0.9 to 0.4 during the optimization

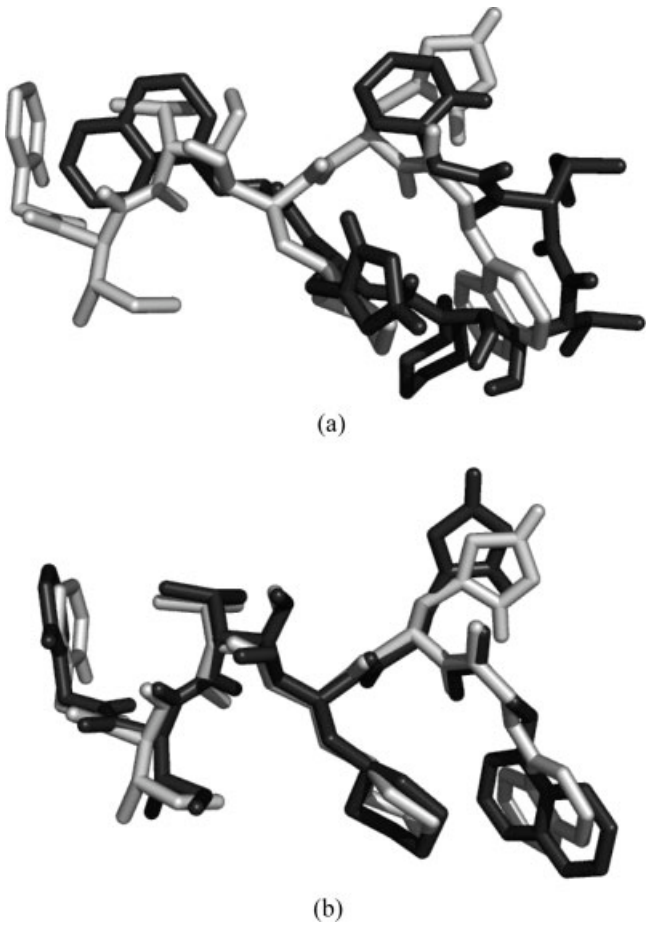

Figure 4. The docked ligand conformations with the lowest energies for 1hiv using 250,000 function evaluations. The native conformations and the predicted conformations are represented by white sticks and black sticks, respectively. (a) LGA-default (RMSD equals to 11.30 Å) and (b) SODOCK (RMSD equals to 1.66 Å).

Table 3. Average Results of LGAs and SODOCK.

	LGA-default					LGA-large				SODOCK						
PDB	Energy	Var.[a]	RMSD	Succ.[b]	Time[c]	Energy	Var.[a]	RMSD	Succ.[b]	Energy	Var.[a]	RMSD	Succ.[b]	Time[c]	$neval_{hit}$[d]	Ratio[e]
1hiv	−3.34	22.61	9.25	0.00	26.35	−3.51	14.80	8.53	0.00	−16.74	22.11	5.42	0.13	36.51	51,500	0.21
1aaq	−3.21	14.38	7.62	0.00	15.94	−6.46	9.01	5.44	0.00	−15.44	4.01	2.92	0.27	23.40	32,000	0.13
1epo	−8.92	4.26	8.23	0.00	15.52	−9.06	3.17	8.16	0.03	−11.62	2.26	7.45	0.07	20.22	68,000	0.27
4phv	−10.53	7.31	8.08	0.03	14.81	−11.34	5.80	7.14	0.00	−17.13	5.47	3.73	0.33	19.92	35,000	0.14
1hpv	−10.14	2.34	6.02	0.00	10.69	−11.36	1.35	4.20	0.23	−14.34	1.59	3.04	0.50	12.83	24,000	0.1
1htf	−13.75	0.86	3.35	0.17	12.68	−13.57	0.48	3.07	0.13	−14.27	1.78	2.96	0.23	16.31	101,000	0.4
1tmn	−10.82	4.32	5.61	0.07	9.56	−12.35	3.01	3.31	0.27	−11.61	2.59	6.62	0.03	11.67	117,500	0.47
1cdg	−6.84	0.17	2.23	0.67	6.59	−7.16	0.27	3.72	0.43	−7.72	0.04	2.12	0.60	7.15	27,500	0.11
1qbt	−9.54	112.21	6.72	0.17	25.86	−14.37	41.59	4.69	0.27	−22.12	26.19	1.60	0.87	36.82	16,500	0.07
1dwd	−13.34	1.29	5.15	0.17	11.12	−13.66	0.24	5.00	0.17	−14.47	1.43	5.72	0.10	14.17	64,500	0.26
1ets	−12.15	1.65	6.21	0.03	11.35	−12.48	1.23	4.63	0.13	−14.20	4.10	4.65	0.20	14.52	51,500	0.21
4hmg	−7.48	0.28	1.88	0.80	13.84	−7.49	0.10	1.42	0.90	−8.28	0.03	0.95	0.97	6.09	22,000	0.09
1hvr	−20.44	6.01	2.16	0.83	12.13	−21.02	0.02	0.88	1.00	−21.36	0.01	0.80	0.97	14.86	18,500	0.07
1stp	−9.61	0.85	1.42	0.73	3.53	−10.09	0.22	0.70	0.97	−10.82	0.01	0.41	1.00	3.84	29,000	0.12
2mcp	−5.91	0.08	2.28	0.37	2.62	−6.18	0.02	1.78	0.67	−6.07	0.07	1.86	0.73	2.51	73,500	0.29
2cpp	−7.31	0.00	1.49	0.87	1.58	−7.34	0.00	2.45	0.40	−7.32	0.00	1.45	0.90	1.42	17,500	0.07
3ptb	−6.94	0.00	0.33	1.00	1.74	−6.94	0.00	0.34	1.00	−6.95	0.00	0.34	1.00	1.89	15,000	0.06
Average	*−9.43*	*10.51*	*4.59*	*0.35*	*11.52*	*−10.26*	*4.78*	*3.85*	*0.39*	*−12.97*	*4.22*	*3.06*	*0.52*	*14.36*	*45,000*	*0.18*
Wins	*0*	*2*	*1*	*2*		*3*	*8*	*4*	*3*	*14*	*11*	*12*	*14*			

[a]Variance of docking energy.
[b]The rate of successful docking (RMSD less than 2.0 Å).
[c]Average execution time in second per run.
[d]Average neval that SODOCK reaches the final energy of LGA-default.
[e]Ratio = $neval_{hit}$/250,000.

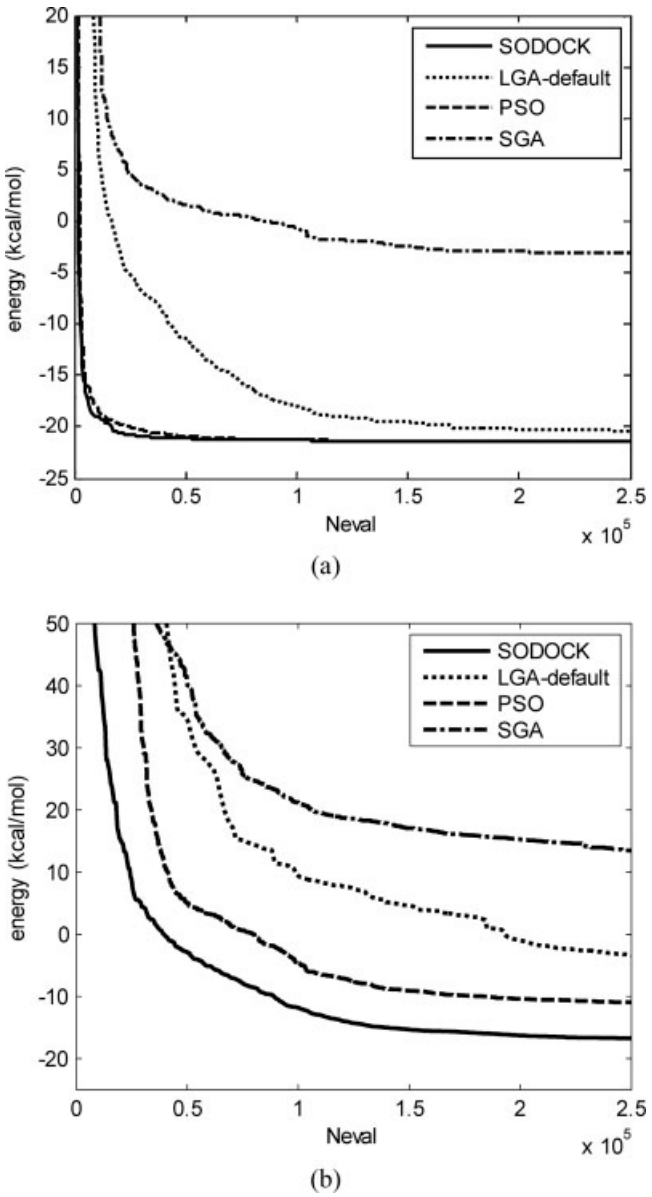

Figure 5. Comparison of average convergence performance: (a) 1hvr and (b) 1hiv.

process, and $c_1 = c_2 = 2.0$. For the constriction factor approach, the inertia weight $w = 0.72984$ and $c_1 = c_2 = 1.4962$. The effectiveness of local search in these versions of PSO was also evaluated. SODOCK refers the combination of the *lbest* neighborhood and the linear decreasing approach, in which local search performs best than in other versions. The supplementary simulation results of various versions of PSO are available on our website http://iclab.life.nctu.edu.tw/sodock/.

The evaluations of the search abilities of SODOCK and LGA obtained in AutoDock are presented here. Table 1 shows the default parameter settings of SODOCK, where the maximal number of function evaluations $\text{neval}_{\max}$ is 250,000. The arithmetic crossover operator proposed by Thomsen,[37] which randomly generates two offspring in the hyperbox formed by two parents, was also implemented as an extension to AutoDock in this study. Three versions of LGA were evaluated: default mode ($N_{\text{pop}} = 50$ and $\text{neval}_{\max} = 250{,}000$), large mode ($N_{\text{pop}} = 200$, $\text{neval}_{\max} = 1{,}000{,}000$), and arithmetic crossover mode (same as the default mode except that arithmetic crossover is adopted). Each method was run 30 times independently for each protein–ligand complex. Table2 shows the best results in terms of docked energy for the four compared methods. The RMSD of heavy atoms between atomic positions in docked ligand and the corresponding positions in the crystal-structure ligand is also given. Simulation results show that SODOCK performs best among all methods in terms of both docked energies and RMSD. SODOCK obtained the lowest energies for 16 out of 17 complexes. For 1hiv and 1aaq with highly flexible ligands, the energies obtained by SODOCK are much lower than those of LGAs. The RMSD values of 1hiv and 1aaq obtained by SODOCK were also much smaller than those of the three LGAs. For the complexes with less flexible ligands, the differences of both docked energy and RMSD between the four methods are not significant. This finding reveals the superiority of SODOCK for highly flexible docking. According to the schema theorem, KrishnaKumar et al.[40] showed that GAs need an enormously large population size and a large number of function evaluations to solve a Large Parameter Optimization Problem. However, the average energy obtained by LGA of the large mode (LGA-large) is just slightly lower than that obtained by LGA of the default mode (LGA-default). This scenario reflects that interactions among encoded parameters, instead of a large search space, make the optimization problem intractable. The LGA with the arithmetic crossover (LGA-arithmetic) had the worst average performance, possibly because LGA-arithmetic is most effective for ligands with a small number of torsions when it cooperates with a modified mutation operator and a specialized parameter setting.[37] Therefore, only the LGA-default and LGA-large were further evaluated. The docked conformations with the lowest energies were also verified. Figure 4 shows that the docked ligand conformation (RMSD equal to 1.66 Å) for 1hiv obtained by SOD-

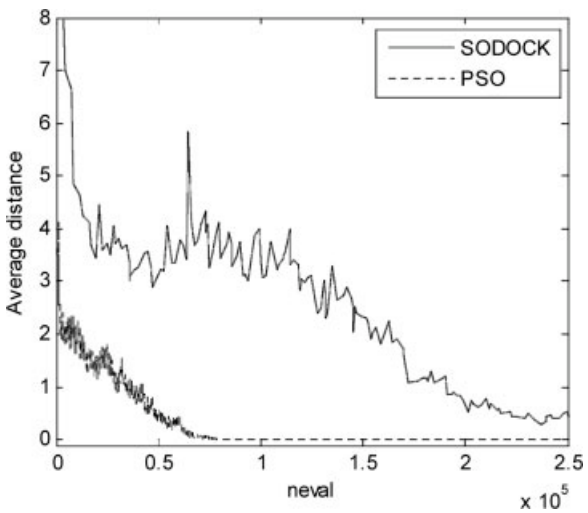

Figure 6. Average Euclidean distance among N_p particles using 1hiv as an example.

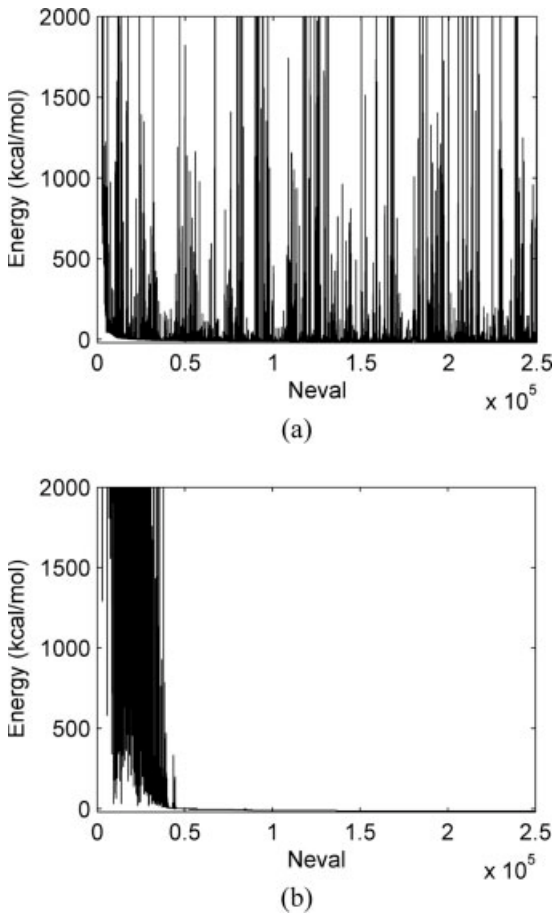

Figure 7. Docked energy of the best individual having the lowest docked energy for 1hiv during the optimization process: (a) SGA, (b) PSO. Final energies obtained by SGA and PSO are −10.52 and −20.13 kcal/mol, respectively.

OCK is much closer to its native conformation. Notably, the result with RMSD equal to 11.30 Å obtained by LGA-default is highly unfavorable.

Table3 summarizes the average results of SODOCK and the LGAs. A docked ligand conformation with RMSD less than 2.0 Å was defined as a successful docking. The docked energies obtained by SODOCK were better than those obtained by LGAs for all complexes except 1tmn. SODOCK is also more robust than LGAs, which have smaller average variances of energies. The simulation results reveal that the success rates of SODOCK are much better than those of LGAs, especially for highly flexible ligands. The average success rate of SODOCK (0.52) is better than those of LGA-default (0.35) and LGA-large (0.39). For LGA-large, increasing the number of function evaluations did not produce a significant improvement. Moreover, the number of function evaluations $\text{neval}_{\text{hit}}$ taken by SODOCK to reach the final energy value of LGA-default was also recorded, showing that SODOCK had better convergence performance using ratio $= \text{neval}_{\text{hit}}/\text{neval}_{\text{max}} = 0.18$ to obtain the same docked energy of LGA-default. The execution time of SODOCK was slightly greater than that of LGA-default using the same number 250,000 of function evaluations. The average execution time of SODOCK per independent run was 14.36 s, while that of LGA-default was 11.52 s. The execution time of LGA-large using 1,000,000 function evaluations was ~4 times that of LGA-default, and is omitted in Table 3.

To further observe the effectiveness of the used local search, this study compared the convergence performance of SODOCK, PSO only, LGA-default, and GA only (SGA). Two complexes were chosen from Table 2, where 1hiv has the ligand noa with 23 torsions and 1hvr has the ligand xk2 with 10 torsions. Figure 5 shows the average convergence performance for the two docking problems. SODOCK had the lowest docked energies for both 1hiv and 1hvr. SODOCK also had the best convergence performance among all methods. Notably, even PSO without local search performed better than LGA and SGA. Additionally, SODOCK and LGA performed better than PSO and SGA, respectively. Thus the local search can improve the search ability of both GA-based and PSO-based methods. For less flexible ligands such as that of 1hvr, PSO alone has sufficient search ability to obtain accurate results (Fig. 5a). However, for highly flexible ligands such as that of 1hiv, the local search of SODOCK is essential to obtain good docked energies and accurate conformations (Fig. 5b).

Figure 6 shows the average Euclidean distance among N_p particles during the optimization process using 1hiv as an example, where the distance is derived as follows:

$$\text{Dist.} = \frac{1}{C_2^{N_P}} \sum_{i=1}^{N_P-1} \sum_{j=i+1}^{N_P} \|X_i - X_j\|. \quad (6)$$

Without the local search, particles of PSO converged rapidly after 10,000 energy evaluations, and thus did not make any further improvement. By contrast, SODOCK using the local search can maintain sufficient diversity among particles during entire evolution, and gives the particles the opportunity to improve the docked energies.

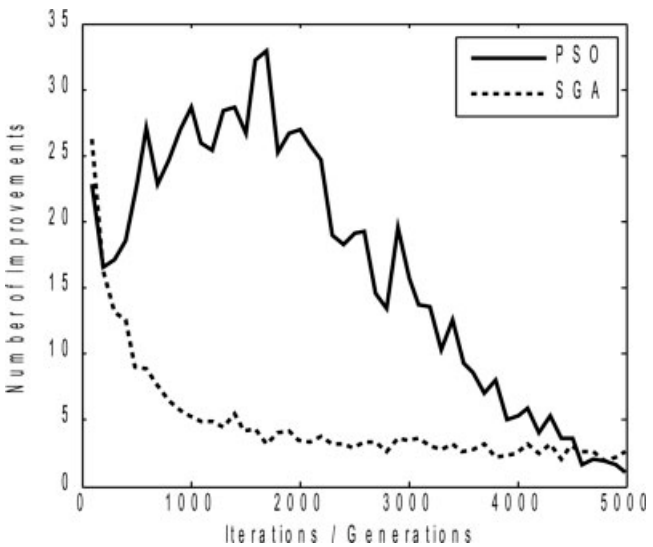

Figure 8. Average number of improvements in terms of docked energy of the best individual in every 100 iterations/generations for 30 independent runs on 1hiv.

Table 4. RMSD Values of the Results with the Best Scores of Each Method.

PDB	Ligand	Torsions	SODOCK	AutoDock[a]	DOCK[a]	FlexX[a]	GOLD[a]
1apt	iva	19	2.18	1.89	8.06	5.95	8.82
6cpa	zaf	17	1.11	8.30	8.30	9.83	4.96
1apu	iva	16	1.42	9.10	7.58	8.43	10.70
1icn	ola	15	7.79	3.99	3.88	2.95	2.05
6tmn	cbz	14	2.99	8.72	7.78	4.51	8.54
2ifb	plm	14	1.91	3.09	1.43	8.94	2.61
1cnx	eg2	12	7.15	10.90	7.35	6.83	6.32
1icm	myr	12	5.26	1.80	3.99	2.94	2.30
1phg	hem	12	0.81	3.52	5.57	4.87	4.20
5tln	ina	10	9.18	5.34	1.39	6.33	1.60
1ets	nas	10	2.15	5.06	3.93	2.11	2.39
1nsc	sia	10	0.89	1.40	4.86	6.00	1.02
1phf	hem	10	0.84	2.09	2.39	4.68	4.42
1etr	mqi	9	1.14	4.61	6.66	7.26	5.99
1nnb	dan	9	0.71	0.92	4.51	0.92	0.84
1nsd	dan	9	0.47	1.20	4.51	1.56	0.96
1ett	tos	8	2.57	8.12	1.33	6.24	1.30
1pph	tos	8	0.92	5.14	3.91	3.27	4.23
3tmn	val	7	4.10	4.51	7.09	5.30	3.96
3cpa	gly	7	1.37	2.26	6.48	1.51	1.87
1rhl	2gp	6	0.86	0.96	0.71	1.15	1.08
1rls	3gp	6	0.68	0.98	1.75	4.33	1.16
1tpp	apa	6	1.65	1.80	3.25	1.95	2.33
5abp	gla	6	0.23	0.48	3.89	4.68	0.59
1cbx	bzs	5	7.12	1.33	3.13	1.32	1.87
1tni	pbn	5	3.92	2.61	5.26	2.73	4.93
1cil	ets	4	2.80	5.81	2.78	3.52	6.04
1abf	fca	4	0.31	0.48	3.25	0.76	0.50
1abe	ara	4	0.25	0.16	1.87	0.55	0.18
1tnk	pra	4	1.50	1.69	1.87	1.70	3.08
1okl	mns	3	1.52	8.54	5.65	4.22	3.55
1gsp	sgp	3	0.54	2.67	1.16	3.71	0.70
1tnj	pea	3	2.12	1.21	1.56	1.73	1.90
1tng	amc	2	2.32	0.62	0.86	1.08	1.89
1tnl	tpa	2	0.46	0.41	2.08	3.74	1.61
3ptb	ben	0	0.34	0.80	0.59	1.11	1.09
2cpp	cam	0	3.08	3.40	2.48	0.44	3.49
Average			*2.29*	*3.40*	*3.87*	*3.76*	*3.11*
Wins			*19*	*7*	*4*	*3*	*4*

[a]Values are obtained from Bursulaya et al.[41]

The search operators of PSO and SGA were further analyzed. Figure 7 shows the docked energy of the best individual, i.e., the one with the lowest docked energy, during the optimization process, taking 1hiv as an example. The energy of SGA was recorded before applying an elitist strategy that directly inherits the best individual of previous generation. The energy of the best individual thrashed seriously in SGA, indicating that the crossover and mutation operators usually disrupt the best individual and produce poor offspring. By contrast, the energy of the best individual in PSO stabilized after a period of evolution. The ability of search operators to generate high-quality offspring was also evaluated by counting the number of energy improvements on the best individual every 100 generations (iterations). Figure 8 shows the average number of improvements from 30 runs for 1hiv, showing that PSO can more easily improve the energy of the best individual than SGA. Above simulation results reveal that PSO is stable in solving docking problems.

Part 2: Docking Accuracy

The 37 protein–ligand complexes obtained from Bursulaya et al.[41] were adopted to evaluate the docking accuracy of SODOCK. Table4 shows the heavy-atom RMSDs of the best scored (lowest-energy) results, where the values of four state-of-the-art docking methods, namely GOLD 1.2,[3] DOCK 4.0,[2] FlexX 1.8,[5] and AutoDock 3.05,[4] were obtained from Bursulaya et al.[41] Each method conducted 30 independent runs with its default parameter setting. Since each method adopted different scoring functions, parameter setting, and computation costs, their docking abilities could not be directly compared by the reported RMSD values in

Table 4. However, all reported RMSD values of the well-known methods could be used as a baseline for evaluating the performance of SODOCK. Table 4 shows SODOCK performed well, obtaining the smallest RMSD values in 19 of 37 cases. The average RMSD 2.29 Å of SODOCK was better than those of other docking programs, which were all above 3.0 Å. AutoDock obtained the smallest RMSD values in 7 cases; GOLD in 4 cases; DOCK in 4 cases; and FlexX in 3 cases. Apart from SODOCK, no method is significantly better than any other in terms of RMSD.

Conclusions and Future Work

This investigation has presented a novel optimization algorithm SODOCK for flexible protein–ligand docking. SODOCK is more simple and efficient than GA-based methods. Simulation results reveal that SODOCK is superior to the LGA of AutoDock for solving flexible docking problems, in terms of convergence performance, docked energy, robustness, and success rate. The docking accuracy of SODOCK was also evaluated using 37 complexes, whose ligands had numbers of torsions ranging from 0 to 19. The results show that SODOCK can obtain more accurate docked conformations than FlexX 1.8, DOCK 4.0, GOLD 1.2, and LGA of AutoDock 3.05.

Our results indicate that the energy function of AutoDock cannot reflect the affinity between ligand and protein for some test cases, due to the approximation in the scoring function. Figure 9 illustrates the scatter plot of test case 1ets using 30 docked results for each of SODOCK and LGA-default, where each point denotes the result of an independent run. From Figure 9, although the lowest energy of 30 runs obtained by SODOCK is lower than that of LGA-default, the corresponding RMSD of the docked conformation with the lowest energy of SODOCK is worse than that of LGA-default. A more appropriate scoring function than that of AutoDock would further improve the docking accuracy of SODOCK in terms of RMSD. Integration of other scoring functions such as X-Score[42] with SODOCK is in progress. Moreover, the coefficients of AutoDock's energy function are also being refined using efficient evolutionary algorithms such as intelligent evolutionary algorithm.[43]

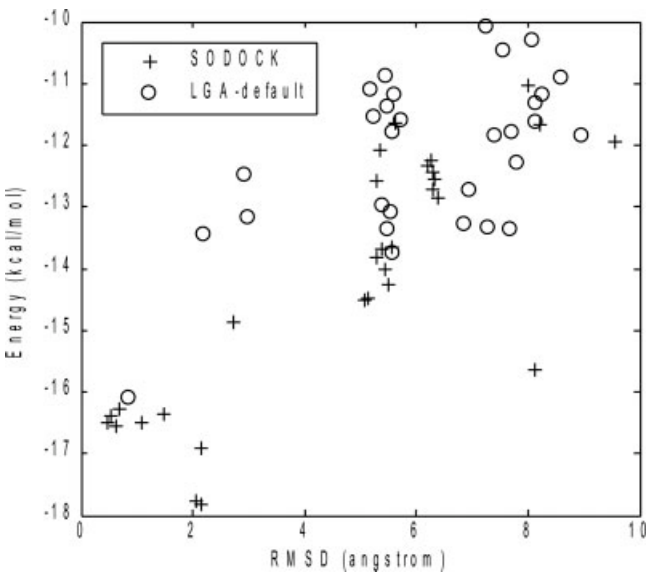

Figure 9. The scatter plot of results obtained by SODOCK and LGA-default for 1ets.

References

1. Jain, A. N. Curr Opin Drug Discov Dev 2004, 7, 396.
2. Ewing, T. J. A.; Makino, S.; Skillman, A. G.; Kuntz, I. D. J Comput Aided Mol Des 2001, 15, 411.
3. Jones, G.; Willett, P.; Glen, R. C.; Leach, A. R.; Taylor, R. J Mol Biol 1997, 267, 727.
4. Morris, G. M.; Goodsell, D. S.; Halliday, R. S.; Huey, R.; Hart, W. E.; Belew, R. K.; Olson, A. J. J Comput Chem 1998, 19, 1639.
5. Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. J Mol Biol 1996, 261, 470.
6. Gehlhaar, D. K.; Verkhivker, G. M.; Rejto, P. A.; Sherman, C. J.; Fogel, D. B.; Fogel, L. J.; Freer, S. T. Chem Biol 1995, 2, 317.
7. Lorber, D. M.; Shoichet, B. K. Protein Sci 1998, 7, 938.
8. Mandell, J. G.; Roberts, V. A.; Pique, M. E.; Kotlovyi, V.; Mitchell, J. C.; Nelson, E.; Tsigelny, I.; Ten Eyck, L. F. Protein Eng 2001, 14, 105.
9. Zavodszky, M. I.; Kuhn, L. A. Protein Sci 2005, 14, 1104.
10. Wang, R. X.; Lu, Y. P.; Wang, S. M. J Med Chem 2003, 46, 2287.
11. Goodsell, D. S.; Olson, A. J. Proteins 1990, 8, 195.
12. Miller, D. W.; Dill, K. A. Protein Sci 1997, 6, 2166.
13. Kennedy, J.; Eberhart, R. C. IEEE Int Conf Neural Netw, Perth, WA, 1995; pp. 1942–1948.
14. van den Bergh, F.; Engelbrecht, A. P. IEEE Trans Evol Comput 2004, 8, 225.
15. Liu, B.-F.; Chen, H.-M.; Ho, S.-Y. In Proceedings of Genetic and Evolutionary Computation Conference, Washington DC, 2005; pp. 267–268.
16. Solis, F.; Wets, R. Math Oper Res 1981, 6, 19.
17. Eberhart, R. C.; Shi, Y. In Proceedings of Seventh International Conference on Evolutionary Programming, San Diego, CA, 1998; pp. 611–616.
18. Shi, Y.; Eberhart, R. C. IEEE Int Conf Evol Comput, Anchorage, AK, 1998; pp. 69–73.
19. Shi, Y.; Eberhart, R. C. In Evolutionary Programming; Porto, V. W.; Saravanan, N.; Waagen, D.; Eiben, A. E., Eds.; Springer-Verlag: Berlin, Germany, 1998; pp. 591–600.
20. Kennedy, J. In Evolutionary Programming; Porto, V. W.; Saravanan, N.; Waagen, D.; Eiben, A. E., Eds.; Springer-Verlag: Berlin, Germany, 1998; pp. 581–590.
21. Eberhart, R. C.; Simpson, P.; Dobbins, R.Computational Intelligence PC Tools; Academic Press: Boston, 1996.
22. Kennedy, J. In Proceedings of the 1999 Congress on Evolutionary Computation, Piscataway, NJ, 1999; pp. 1931–1938.
23. Clerc, M.; Kennedy, J. IEEE Trans Evol Comput 2002, 6, 58.
24. Parsopoulos, K. E.; Vrahatis, M. N. In Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation; Grmela, A.; Mastorakis, N., Eds.; WSEAS Press: Interlaken, Switzerland, 2002; pp. 216–221.
25. Eberhart, R. C.; Shi, Y. IEEE Int Conf Evol Comput, La Jolla, CA, 2000; pp. 84–88.
26. Mohais, A. S.; Mendes, R.; Ward, C.; Posthoff, C. In AI 2005: Advances in Artificial Intelligence; Zhang, S.; Jarvis, R., Eds.; Springer-Verlag: Berlin, Germany; 2005; pp. 776–785.
27. Li, X. Y.; Tian, P.; Kong, M. In AI 2005: Advances in Artificial Intelligence; Zhang, S.; Jarvis, R., Eds.; Springer-Verlag: Berlin, Germany; 2005; pp. 1305–1310.

28. Liang, J. J.; Qin, A. K.; Suganthan, P. N.; Baskar, S. IEEE Int Conf Syst Man Cybern 2004, 3659.
29. Coello, C. A. C.; Pulido, G. T.; Lechuga, M. S. IEEE Trans Evol Comput 2004, 8, 256.
30. Goldberg, D. E. Genetic Algorithms in Search, Optimization and Machine Learning; Addition-Wesley: Reading, MA, 1989.
31. Pelikan, M.; Goldberg, D. E.; Cantu-Paz, E. Evol Comput 2000, 8, 311.
32. Davidor, Y. In Foundations of Genetic Algorithms; Rawlins, G. J. E., Ed.; Morgan Kaufmann: San Francisco, 1991; pp. 23–35.
33. Rochet, S. Inf Sci 1997, 102, 133.
34. Goldberg, D. E.; Deb, K.; Korb, B. Complex Syst 1990, 4, 415.
35. Harik, G. R.; Goldberg, D. E. Comput Methods Appl Mech Eng 2000, 186, 295.
36. Mühlenbein, H.; Paaß, G. In Parallel Problem Solving from Nature—PPSN IV; Eiben, A.; Bäck, T.; Shoenauer, M.; Schwefel, H.-P., Eds.; Springer-Verlag: Berlin, 1996; pp. 178–187.
37. Thomsen, R. Biosystems 2003, 72, 57.
38. Kellenberger, E.; Rodrigo, J.; Muller, P.; Rognan, D. Proteins 2004, 57, 225.
39. Wu, G. S.; Robertson, D. H.; Brooks, C. L.; Vieth, M. J Comput Chem 2003, 24, 1549.
40. KrishnaKumar, K.; Narayanaswamy, S.; Garg, S. In Genetic Algorithms in Engineering and Computer Science; Winter, G.; Périaux, J.; Galán, M.; Cuesta, P., Eds.; Wiley: Chichester, UK, 1995.
41. Bursulaya, B. D.; Totrov, M.; Abagyan, R.; Brooks, C. L. J Comput Aided Mol Des 2003, 17, 755.
42. Wang, R. X.; Lai, L. H.; Wang, S. M. J Comput Aided Mol Des 2002, 16, 11.
43. Ho, S.-Y.; Shu, L.-S.; Chen, J.-H. IEEE Trans Evol Comput 2004, 8, 522.