Improving The Efficiency of PSOVina for Protein-Ligand Docking By Two-Stage Local Search

Hio Kuan Tai, Hang Lin, Shirley W. I. Siu
Department of Computer and Information Science
Faculty of Science and Technology
University of Macau
Macau, China
giottotai@yahoo.com.hk, lhang33@126.com, shirleysiu@umac.mo

***Abstract*—Protein-ligand docking programs are valuable tools in the modern drug discovery process for predicting the complex structure of a small molecule ligand and the target protein. Often, the configurational search algorithm in the docking tool consists of global search and local search. The former is to explore widely for promising regions in the search space and the latter is to optimize a candidate solution to a local optimum. However, accurate local search methods such as gradient-based Newton methods are very costly. In this investigation, we present a new approach to enhance the time efficiency of a docking program by introducing a two-stage local search method. Given a candidate solution, a rough local search is performed in the first stage to determine the potentiality of the solution. Only if the solution is promising, the second stage with a full local search will be performed. Our method has been realized in the PSOVina docking program and tested on two data sets. The experimental results show that two-stage local search achieves almost 2x speedup to conventional one-stage method, it also enhances the prediction performance of the docking method in terms of increased success rate and RMSD.**

***Keywords*—protein-ligand docking; local search; PSOVina; PSO; BFGS; optimization**

I. Introduction

Protein-ligand docking has been routinely used for structure-based drug design in the modern drug discovery process [1]. Given a target *protein* relevant to the disease of interest, the main goal of docking is to predict the complex structure of a small-molecule *ligand* bound to the active site of the protein. This structure, if correctly predicted, allows one to study the specific interactions between the binding partners and to estimate the binding affinity. Starting from a large library of ligands, each ligand will be docked to the protein, ranked, and filtered through the process of *virtual screening*; only ligands that bind with a high level of affinity are selected as *lead* compounds and to be further optimized through the lead optimization cycle. The most potent compounds among the candidates will be transferred to the preclinical test phase and finally to the clinical test phase. Therefore, protein-ligand docking serves as the foremost important steps in drug development. High quality docking can significantly cut down the time and cost for the initial lead selection steps and help guiding biological experiments throughout the process.

Computationally, the docking problem is an optimization task which is to search for the minimum energy ligand configuration upon complexation with protein, including its *position*, *orientation* and *conformation*. Here, the ligand conformation refers to the spatial arrangements of the ligand atoms by rotation about single bonds. The number of rotatable bonds in a ligand varies from 0 to 100; the larger the number of rotatable bonds that a ligand has, the greater the flexibility of it. When there are many rotatable bonds to be considered in the docking process, the optimization task becomes extremely challenging as the number of potential ligand configurations are exponential. The docking problem becomes even more intractable when protein flexibility (including change in configurations of selected parts of the protein sidechain or backbone) and other environmental factors (such as location of water molecules and ions) are taken into account.

So far, a wide range of optimization strategies have been proposed to solve the configurational search problem. These include simulated annealing (SA) [2], Monte Carlo (MC) [3], and genetic algorithm (GA) [4], etc. Recently, new heuristic-based approaches using particle swarm optimization (PSO) and other nature-inspired methods such as ant-colony optimization have been introduced as alternative search algorithms. The advantages of these population-based metaheuristics are their simplicity in implementation, fast convergence, robustness, and ease of parallelization in multicore CPU/GPU computing architectures. Of particular interest is PSO and their variants as they are known to be efficient for solving continuous nonlinear optimization problems. A few PSO-based docking methods have been implemented to date, such as SODOCK [5], PSO@AutoDock [6], FIPSDock [7], and PSOVina [8]. All proved to be superior to methods using traditional optimization strategies in terms of convergence performance and obtained energy. In these implementations, PSO serves as the global optimizer to rapidly locate promising regions in the search space while the local search method localizes the approximated solutions to local minima from PSO moves and thus increasing the chance to find the optimal solution. Especially for highly flexible ligands, local search is essential to obtain good docked energies and accurate conformation for highly flexible ligands [5, 6]. The success of local search was attributed to its ability to maintain diversity of PSO particles similar to the mutation operator and could better avoid premature convergence to occur [5]. However, the main disadvantage of incorporating a local search routine into the optimization process is the increased cost for additional steps, for both solution generation and energy evaluation. For example, in the study of [6], method which hybridized PSO with Solis and Wets local search increased the computing time by 3 fold as compared to the one without local search.

Our recently proposed docking method PSOVina, is also a hybrid model that combines PSO global search and BFGS local search for solving the protein-ligand docking problem [8]. Facing the inherently high computing cost of the BFGS local search algorithm, PSOVina provides merely 1.79x speedup over its original code AutoDock Vina although PSO converges faster than the MC search in Vina [9]. To improve the time efficiency while maintaining the docking performance of PSOVina, in this investigation, we presents a novel algorithm called two-stage local search (abbreviated as 2LS) to break up the original local search loop into stages of rough search and full search. This approach successfully reduces the time wasted in optimizing unpromising solutions therefore increases the run time performance of the search. Using the diverse PDBbind dataset and the selected 17 complexes from the GOLD benchmark dataset, we compared PSOVina^{2LS} to PSOVina, AutoDock Vina, and seven additional state-of-the-art docking methods. Experiment results reveal that PSOVina^{2LS} achieves 1.92x speedup over PSOVina, 3.44x speedup over AutoDock Vina and yields the highest success rate among all methods.

II. Background

A. Protein-Ligand Docking in AutoDock Vina

AutoDock Vina [9] is the new implementation of the AutoDock [10] flexible protein-ligand docking program. In Vina, the protein is treated as rigid while the ligand is treated as a flexible molecule. As such, a ligand configuration is represented by (i) its position in Cartesian coordinates (x, y, z), denoting the translation of the ligand relative to the center of the box enclosing the active site; (ii) its orientation in quaterion (q_0, q_1, q_2, q_3), where [q_0, q_1, q_2] denotes a vector as axis and q_3 the rotation angle about this axis; and (iii) the torsional angles of τ rotatable bonds ($r_0, r_1, .. , r_{\tau-1}$). Therefore, each candidate solution consists of $7 + \tau$ parameters which will be evaluated according to the scoring function of Vina.

In essence, Vina adopts an empirical-based scoring function which is a weighted sum of interaction terms including steric, hydrophobic and hydrogen bond interactions, and a conformation-independent term based on the number of active rotatable bonds between heavy atoms.

For the optimization algorithm, Vina uses the Iterated Local Search global optimizer where a succession of mutations and local optimization based on the Broyden-Fletcher-Goldfarb-Shannon (BFGS) algorithm are taken and the move in each iteration is accepted according to the Metropolis criterion. The maximum number of iteration is a heuristic function of the number of ligand rotatable bonds that is determined at the beginning of the run. Vina allows the user to perform the run several times in parallel starting from different ligand configurations, then it combines and ranks the best solution found in all runs to produce the final result.

B. PSOVina

PSOVina is our recently released docking program based on the AutoDock Vina software [8]. It uses a modified version of the standard PSO as the global optimizer together with the BFGS local search and Vina's scoring methods.

PSO is a population-based search algorithm inspired by social behaviors of birds. The population is called a *swarm* and each individual is called a *particle.* In the standard version of PSO, each particle moves around the search space with a velocity. The position of a particle i is encoded in a D-dimensional vector $X_i = [x_{i1},..,x_{iD}]$ where D is the number of dimensions to be optimized and its velocity in the vector of the same size $V_i = [v_{i1},..,v_{iD}]$. Each particle remembers the best position it has ever visited, which is called *pbest* and denoted as X_i^*; the swarm remembers the best position that the group as a whole has ever visited, which is called *gbest* and denoted as X_g. The particles move over iteration t according to the following update mechanism:

$$V_i(t+1) = \omega V_i(t) + \alpha\, rand()\,(X_i^*(t) - X_i(t)) + \beta\, rand()\,(X_g(t) - X_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

where the positive constants ω, α and β are the inertia, cognitive and social weights to be specified by the user. Proper tuning of these parameters is important to ensure the search efficiency of the swarm. The function rand() is a uniform random number generator yielding the value between 0 and 1 at each entry.

In PSOVina, each PSO particle represents one ligand configuration. At the start of a global search iteration, a ligand transformation type – translation, rotation, or torsional rotation – is randomly selected. Then, a local optimization is carried out on each particle and the fitness value is calculated. The *pbest* and *gbest* are updated to solutions with better fitness values if they are found. Afterwards, the next generation of swarm pertaining to the selected transformation type was generated following the velocity and position updates mechanism. To maintain diversity of the swarm population, particles moved outside of the search space are reinitialized to random positions within the active site based on the *regeneration principle*. Finally, the search terminates until *gbest* is converged or the maximum number of iterations is reached. Our experiments of docking and virtual screening showed that PSOVina achieved a remarkable time reduction of 51-60% as compared to Vina with some improvements regarding docking accuracies.

C. Application of Local Search on Docking Problem

As mentioned above, the goal of local search is to locate the local minimum for the approximated solution resulted from a global search move. It increases the success rate of finding the global minimum in protein-ligand docking problem, especially for highly flexible ligands.

Several local search methods have been widely employed in protein-ligand docking programs. For example, a simplex local search algorithm described by Nelder and Mead (NMS) [11] was used in PLANTS, an ant-colony optimization docking program by Korb et al. [12]. Simply speaking, NMS transforms the vertices of a polytope by reflection, expansion and

Funded by University of Macau (grant number MYRG2015-00212-FST)

contraction operations until the fractional range from the highest to the lowest point in the simplex with respect to the function value is less than a tolerance value or until a maximum number of function evaluations is reached. In PLANTS, NMS local search was conducted on each solution carried by an *ant* and reapplied it to iteration-best ant to further improve the solution.

Another popular local search method for protein-ligand docking problem is Solis and Wets [13]. It was employed in AutoDock 4 [10] and numerous other programs and shown to be greatly efficient [14]. The Solis and Wets algorithm is a directed search method with an adaptive step size. Starting from a candidate, two new solutions are generated by adding or subtracting a random value drawn from a normal distribution whose standard deviation is controlled by a parameter ρ. If either of the new solutions is better, one success is counted and the current solution is replaced with the successful solution; otherwise, one failure is counted. If number of consecutive successes is met, ρ is multiplied by an *expansion factor* to allow larger leaps between the current and the next solution; on the contrary, if number of consecutive failures is recorded, the value of ρ is reduced by multiplying with a *contraction factor* to effectively localizing the search. The search is terminated when ρ falls below a lower-bound threshold or reaches the maximum of steps.

The new release of AutoDock program, AutoDock Vina [9] employed a quasi-Newton local search method called BFGS. BFGS is named after the first letter of the names of its inventors, Broyden, Fletcher, Goldfarb and Shanno. It aims to estimate an approximate of the inverse of Hessian matrix and at the same time uses less calculation than the Newton method. The procedure of this method is shown below:

(*Step1*) Set H_k as the approximation of the inverse of the Hessian matrix. The initial point is x_0 and the error limit $\epsilon > 0$. Initially, $H_0 = 1$ and $k = 0$.

(*Step 2*) Calculate the search direction $p_k = -H_k \nabla f(x_k)$ (3)

(*Step 3*) Find the value $\alpha_k > 0$, which makes

$$f(x_k + \alpha_k p_k) = \min_{\alpha \geq 0} f(x_k + \alpha p_k) \quad (4)$$

(Step 4) Update $x_{k+1} = x_k + \alpha_k p_k$ (5)

(Step 5) If $\left\| \nabla f(x_{k+1}) \right\| \leq \epsilon$, terminate the algorithm and output x_{k+1} as the approximate minimum. Otherwise continue.

(Step 6) Update the approximate matrix by

$$s_k = x_{k+1} - x_k \quad (6)$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \quad (7)$$

$$H_{k+1} = H_k + \frac{1}{s_k^T y_k}\left(1 + \frac{y_k^T H_k y_k}{s_k^T y_k}\right) s_k s_k^T - \frac{1}{s_k^T y_k}\left(s_k y_k^T H_k + H_k y_k s_k\right) \quad (8)$$

Set $k = k+1$, go back to (*Step 2*).

In Step 3, α_k is found by performing line search in the direction of p_k.

The primary advantages of BFGS are that there is no need to solve the Hessian matrix which is very costly and the update equation to obtain the approximate matrix in every iteration can be computed easily.

While different docking programs may have different favors of local search for the task, an interesting question to answer is whether there exists the best method for the problem. Tavares and co-workers in the studies of the impact of local search on protein-ligand docking optimization [15, 16] tried to answer this question: They defined a structural distance measure $d_{struct}(A, B)$ of two solutions (A and B) by calculating the absolute difference between the root-mean-square deviations of the two ligand docking poses to the experimental conformation. In this way, the amount of *innovation* introduced by successive m iterations (in each iteration includes global search moves with or without local search moves) in an optimization method can be quantified with the value of $d_{struct}(A, A^m)$. The method having small innovation value implies a small modification in the ligand docking poses after some iterations and is referred as having strong locality; if the value is large, it is referred as weak locality. A good local search algorithm should introduce some degrees of innovation yet non-excessive to the solution generated by global search moves in order to explore the neighborhood of the current solution in focus and efficiently. Based on this framework, Tavares and co-workers showed that neither the simplex local search NMS nor the popular Solis and Wets algorithm is an ideal local search method since its solution quality is similar to global searching using GA. Instead, BFGS was shown to greatly enhance the prediction accuracy and fitness of the final solution in a test of 8 complexes data set [16].

III. The Proposed Two-Stage Local Search Algorithm

Our docking program PSOVina is also a hybrid global and local search algorithm - using PSO for the former and BFGS for the latter – to achieve efficiency and accuracy in searching for the optimal ligand configuration. Based on our observations, the use of local search here is crucial for locating the optimal solution since a slight overlap of atoms (ligand-ligand or ligand-protein atoms) will result in atomic clashes that will lead to a large increase in potential energy. Performing a local search can effectively remove those high energetic interactions and find the best "local" adjustment of ligand atomic positions with respect to the surrounding protein atoms.

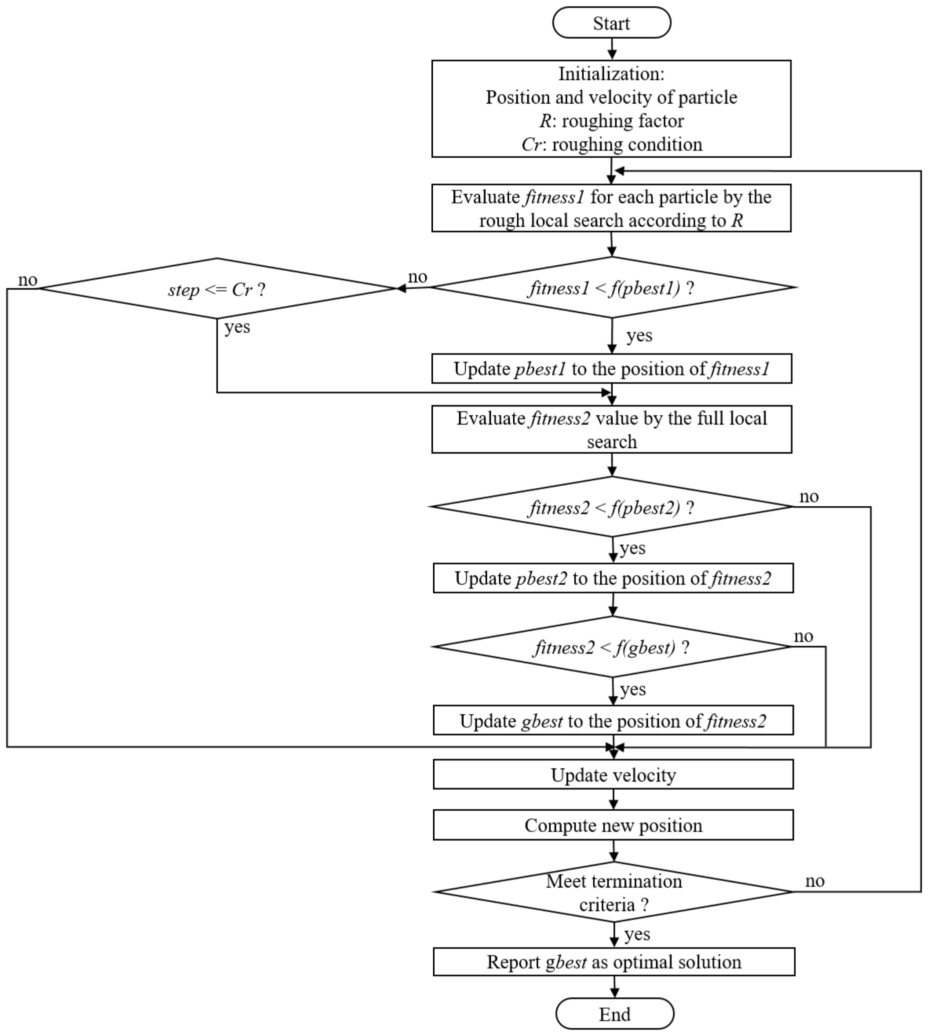
In the BFGS algorithm implemented in AutoDock Vina, a maximum step size of 1000 is set for each local search invocation. The search will stop only if $\left\| \nabla f(x_{k+1}) \right\| \leq 10^{-5}$ (step 5 in BFGS algorithm) or the maximum number of steps is reached. Based on our observations, in most of the cases the searches run until the maximum number of steps and thus a *full* local search is often executed. However, a *full* local search is not always necessary especially when the position of the particle is far from the global optimal solution.

In order to reduce the local search computations in optimizing positions of unpromising particles, here we propose a two-stage local search method. The idea is to first do a shorter or *rough* local search on a given solution in order to assess the potentiality of it to contribute to a successful global search. A high-potential solution is the one which is better than known historical best solution of a particle (*pbest*). If it is a high-potential solution, a *full* local search will be performed to further optimize its positional values. Since the fitness value obtained by a rough local search is greatly different from the one by a full local search, it is necessary to maintain two sets of historical best solutions; one obtained from rough local search and another from full local search.

The flowchart of our proposed two-stage local search method is presented in Fig. 1. At the start, the position and velocity of all particles are initialized to random values. Two-stage local search will be performed to optimize the position of each particle: In the first stage, the fitness (*fitness1*) of a particle is calculated by a rough local search according to the roughing factor (*R*). This roughing factor, typically having value between 0 and 1, is a predefined value to determine how much shorter (or rougher) the local search is by reducing the maximum number of step in the first stage of the local search by a factor of *R*. If the fitness is better than its particle's best position obtained from previous rough local searches (*pbest1*), then *pbest1* will be updated and the second stage is invoked. Here, in the second stage, the fitness (*fitness2*) of a particle is calculated by the full local search. If the fitness is better than the particle's best position obtained from previous full local searches (*pbest2*), then *pbest2* will be updated. The global best solution *gbest* is compared to the fitness and get updated if the fitness is better. Finally, normal PSO procedure is resumed to generate the velocity and position of the particle. The search will stop if *gbest* is converged or the maximum number of step is reached.

Since in the beginning of the PSO exploration, particles are still roaming in random positions far from the optimal solution, the second stage local search should be invoked in the first *Cr* iterations in any case in order to obtain more accurate information of the search space.

Fig. 1. Flowchart of the proposed two-stage local search method.



Start
Initialization:
Position and velocity of particle
R: roughing factor
Cr: roughing condition
Evaluate *fitness1* for each particle by the rough local search according to *R*
fitness1 < *f(pbest1)* ?
no
step <= *Cr* ?
no
yes
yes
Update *pbest1* to the position of *fitness1*
Evaluate *fitness2* value by the full local search
fitness2 < *f(pbest2)* ?
no
yes
Update *pbest2* to the position of *fitness2*
fitness2 < *f(gbest)* ?
no
yes
Update *gbest* to the position of *fitness2*
Update velocity
Compute new position
Meet termination criteria ?
no
yes
Report g*best* as optimal solution
End

IV. Experiments and Results

A Data Sets

Two data sets were used to assess the docking performance of PSOVina with two-stage local search and without: The first data set is obtained from the PDBbind-CN database, which is a manually curated database of a collection of known 3D structures of protein-ligand complexes with experimentally measured binding affinity data [17]. Specifically, we used the *core-set* data from the latest PDBbind v2014 which has been widely used for testing scoring functions in protein-ligand binding affinity prediction. It contains 195 good quality structures with resolution $\leq$ 0.25 nm obtained from systematically sampling of complexes from protein clusters with strongest, medium, and weakest binding protein-ligand structures. To also benchmark our method to state-of-the-art docking programs, such as FIPSDock, PSO@AutoDock, SODOCK, Glide, GOLD, FlexX, and Surflex, a selection of 17 complexes derived from the GOLD benchmark data set [18], which are also classified as high-quality complexes in the PDBbind database (in the *refined-set* but not the *core-set*), were used. These are (PDB ID) 1ATL, 1BMA, 1CBX, 1FKG, 1FKI, 1HSL, 1LAH, 1LCP, 1LST, 1TMN, 1TNG, 1TNI, 2CGR, 2CHT, 2CTC, 2SIM, 6RNT.

Before the docking process, the search space in the coordinate system of the protein has to be specified. For this, the pocket PDB file of each of the complex included in the data set was used. The center of the search space was defined as the geometric center of the protein's active site and the dimensions were determined from the minimal rectangular box enclosing all active-site atoms.

B. Performance Metrics

To evaluate the accuracy of the ligand docked pose, we compare the predicted position of each ligand atom to its experimental position in the experimental structure using the standard root-mean-square deviation (RMSD):

$$RMSD_{lig} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - x_{expt})^2 + (y_i - y_{expt})^2 + (z_i - z_{expt})^2}{n}} \qquad (9)$$

where n is the number of ligand atoms and x_{expt}, y_{expt}, z_{expt} are the Cartesian coordinates of atoms in the experimental structure.

As a rule of thumb, docking poses having RMSD below or equal to 0.2 nm are considered successful predictions, between 0.2 and 0.3 nm are classified as partially docked.

All docking simulations were conducted on a Dell XPS 8700 desktop with Intel Core i7 quad-core 3.6 GHz processor and 24 GB memory running Linux operating system.

C. Results

In the following experiments, we compare the prediction performances of three protein-ligand docking methods: the proposed two-stage local search method implemented in PSOVina (PSOVina^{2LS}), the original PSOVina, and AutoDock Vina.

To get a general idea of how our proposed two-stage local search method performed, we first ran predictions over the PDBbind core-set and measured the RMSD, correlation coefficient (r) of the predicted binding affinity to the experimental binding affinity, success rate, number of iterations to convergence, and runtime per docking instance. For each complex in the data set, 30 docking repeats were performed and the optimal docked pose was selected to be the one with the lowest predicted binding affinity (here called the *overall-lowest-energy* docked pose).

For PSOVina and PSOVina^{2LS}, the same PSO parameters were used: $N = 8$, $\omega = 0.36$, $\alpha = \beta = 0.99$. For PSOVina^{2LS}, we set $R = 0.5$ and $Cr = 20$.

The prediction accuracy and the prediction behavior of the three methods can be seen in Fig. 2 by plotting the average RMSD of the overall-lowest-energy docked poses over the docking repeat. Overall, the prediction performance follows this PSOVina^{2LS} > PSOVina > AutoDock Vina and the three methods show very different behavior. The performance of AutoDock Vina is very stable regardless of the number of docking repeats conducted. This is a desirable behavior of a prediction method as the need to repeat the docking experiment can be reduced. PSOVina performs better than AutoDock Vina in terms of average RMSD over all docking repeats; however, the prediction performance shows some fluctuations (between the 1st and the 10th docking repeats). It is also noted that more docking repeats do not get the prediction accuracy improved. Interestingly, such instability was not inherited to PSOVina^{2LS}. After 6 docking repeats, PSOVina^{2LS} has already reached the lowest RMSD value of 0.199 nm which is maintained around this value until 30 docking repeats.

Fig. 2. Comparison of prediction accuracies of PSOVina^{2LS}, PSOVina, and Autodock Vina in terms of average RMSD of the overall-lowest-energy docked poses using the PDBbind data set.

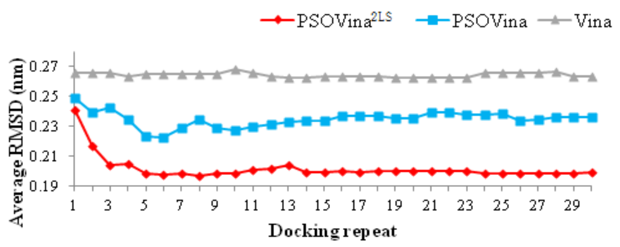


TABLE I. Summary of prediction and runtime perfromances of PSOVina^{2LS}, PSOVina, and AutoDock Vina using the PDBbind data set

	PSOVina^{2LS}		PSOVina		Vina	
	Value	SD	Value	SD	Value	SD
RMSD (nm)	0.199	0.233	0.236	0.270	0.264	0.285
r	0.569	–	0.571	–	0.611	–
Success rate (%)	70.77	–	65.64	–	63.08	–
Number of iteration	938.36	589.10	883.58	576.74	22776.94	7130.47
Runtime (s)	6.53	10.21	8.94	15.74	21.42	46.38

SD: standard deviation

The summary of other prediction metrics (correlation coefficient and success rate) and runtime performances (number of iterations and runtime) of the three methods are presented in Table I. PSOVina^{2LS} has obtained a successful prediction rate of 70.77%, outperforming both PSOVina (-5.13%) and Vina (-7.69%). Although the average number of iterations in the PSO optimization process is increased in PSOVina^{2LS}, PSOVina^{2LS} uses only about 73% of the time in PSOVina to finish each docking instance on average. This is expected since two-stage local search reduces unpromising and unnecessary local search loops; the saved time is used for performing more iterations of global search and promising local search.

We also evaluated the effect of roughing factor (*R*) and roughing condition (*Cr*). As mentioned in Section III, the former is to reduce the number of iterations in the local search loop to achieve a shorter or rougher local optimization, whereas the latter is to control the number of initial PSO iterations which will invoke full local search in any case for the purpose of obtaining accurate information about the search space.

In this experiment, we tested PSOVina^{2LS} using the PDBbind data set with different *R* and *Cr* settings, specifically *R = {0.1, 0.3, 0.5}* and *Cr = {15, 18, 20, 22, 50}*. For each pair of setting, we ran the full data set for 5 times independently and reported the RMSD of the overall-lowest-energy docked poses at the end of the 30th docking repeat. The prediction accuracies in terms of average RMSD and the runtime performances are compared in Fig. 3 and Fig. 4 respectively. It is clear that neither the change in *R* nor *Cr* has much influence on the prediction accuracy although slight variations can be observed. The lowest average RMSD of *0.1924762 ± 0.0065626 nm* is reported by *R = 0.1* and *Cr = 18* and the second lowest of *0.1944068 ± 0.0020471 nm* is reported by *R = 0.3* and *Cr = 15*, whereas the highest average RMSD of *0.2105162± 0.002127943 nm* is reported by *R = 0.5* and *Cr=22*. However, application of different *R* influences greatly the runtime. As shown in Fig. 4, dockings completed with average of *3.9 s*, *5.3 s*, and *6.5 s* when using *R = 0.1*, *0.3*, and *0.5*, respectively. Therefore, smaller *R* speeds up significantly across all tested *Cr* values.

Table II shows the average total of local search function calls and energy evaluation function calls of PSOVina^{2LS} and PSOVina over the complete PDBbind data set in five independent docking runs. On average, as compared to PSOVina, PSOVina^{2LS} reduces the number of local search function calls by -5.85% and the number of energy evaluation by as much as -54%.

Consider *R = 0.1* and *Cr = 18* as the optimal parameters of PSOVina^{2LS}, we performed predictions on a selection of 17 protein-ligand complexes derived from the GOLD benchmark data set. This allows us to compare our proposed method with 9 state-of-the-art docking programs, including FIPSDock, PSO@AutoDock, SODOCK, AutoDock, Glide, GOLD, FlexX, Surflex, and MolDock. For completeness, we included predictions using PSOVina and AutoDock Vina which have not been tested on this data set before.

The average RMSD as the prediction accuracy over 30 docking repeats is shown in Fig. 5. The prediction performance is consistent with the previous experiment using PDBbind data set, i.e. PSOVina^{2LS} > PSOVina > AutoDock Vina. While AutoDock Vina maintains almost stable performance regardless of the number of repeated runs, the two PSOVina methods improve the predictions as more docking repeats were conducted. The final predictions made by PSOVina^{2LS} and PSOVina (at the 30th docking repeat) are with average RMSD of 0.129 and 0.192, respectively. Regarding the runtime, PSOVina^{2LS} achieves 1.92x speedup over PSOVina and 3.44x speedup over AutoDock Vina.

Table IV lists the RMSD of each complex by 11 docking programs, where data of 9 state-of-the-art docking programs was taken from [7]. We can see that PSOVina^{2LS} and PSOVina performed comparably to other methods. A summary of their prediction performances is presented in Table V. Out of 17 complexes, PSOVina^{2LS} predicts 15 cases successfully, yielding the highest success rate of 88.24% among all methods. The method which predicts with the lowest average RMSD is GOLD (11 cases) and FIPSDOCK, and the method which predicts with the lowest average RMSD among all successful cases is Surflex.

Fig. 3. Evaluation of parameters *R* and *Cr* against average RMSD.

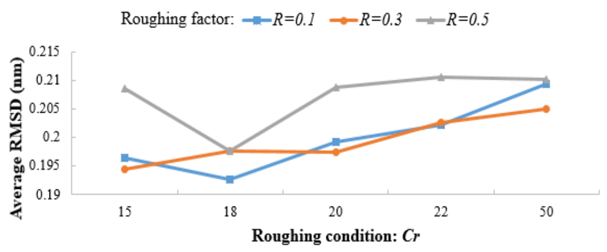


Fig. 4. Evaluation of parameters *R* and *Cr* against average runtime.

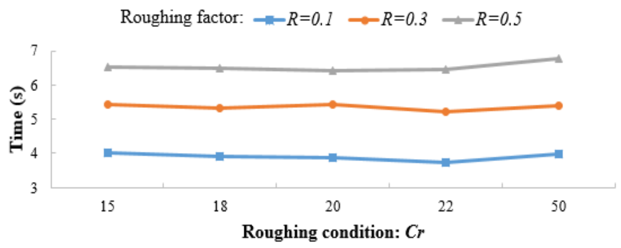


Fig. 5. Comparison of prediction accuracies of PSOVina^{2LS}, PSOVina, and Autodock Vina in terms of average RMSD on the GOLD benchmark data set. The same PSO parameters were used for PSOVina^{2LS} and PSOVina; the optimal parameters *R=0.1* and *Cr=18* are applied in PSOVina^{2LS}.

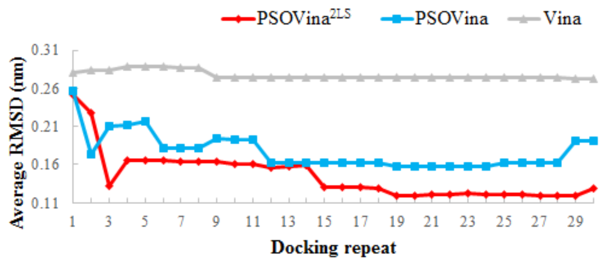


TABLE II. COMPARISON OF THE AVERAGE NUMBER OF LOCAL SEARCH FUNCTION CALLS AND ENERGY EVALUATION FUNCTION CALLS

	Local search function calls		Energy evaluation function calls	
	PSOVina^{2LS}	PSOVina	PSOVina^{2LS}	PSOVina
Repeat 1	**8564.77**	9948.94	**119783.79**	252701.33
Repeat 2	9751.88	**9262.96**	**116849.58**	252870.81
Repeat 3	**8416.94**	9007.15	**118007.1**	252980.98
Repeat 4	**8760.14**	9348.83	**115000.77**	253478.63
Repeat 5	**9025.26**	9717.33	**112331.01**	252993.19
Average	8903.80	9457.04	116394.45	253004.99
SD	235.22	167.56	1279.63	129.47

TABLE III. SUMMARY OF PREDICTION AND RUNTIME PERFROMANCES OF PSOVINA^{2LS}, PSOVINA, AND AUTODOCK VINA ON THE GOLD BENCHMARK DATA SET

	PSOVina^{2LS}		PSOVina		Vina	
	Value	SD	Value	SD	Value	SD
RMSD (nm)	0.129	0.118	0.192	0.163	0.273	0.234
r	0.387	–	0.425	–	0.363	–
Success rate (%)	88.24	–	76.47	–	58.82	–
Number of iteration	1063.27	834.38	865.56	355.95	20475	4311.70
Runtime (s)	2.66	2.53	5.11	5.16	9.16	9.71

V. CONCLUSION

In this paper, we have presented a two-stage local search method called PSOVina^{2LS} which implemented based on the original PSOVina in order to reduce the computing cost of the standard local search. The main idea of this method is to perform a rough local search to determine potentiality of a given solution before an expensive full local search is conducted. Empirical tests show that the rougher the first stage local search is, the less runtime the docking will take.

The performance of PSOVina^{2LS} was compared with PSOVina, AutoDock Vina and nine other state-of-the-art docking programs. Our experimental results show that PSOVina^{2LS} has better prediction accuracy and runtime performance than PSOVina and AutoDock Vina. It also outperforms other docking methods with the highest success rate.

ACKNOWLEDGEMENT

This work is supported by the University Senate Research Committee of University of Macau (grant number MYRG2015-00212-FST).

TABLE IV. COMPARISON OF DOCKING ACCURACIES OF VARIOUS DOCKING PROGRAMS

PDB	Torsions	PSOVina^{2LS}	PSOVina	AutoDock Vina	FIPS	PSO@AutoDock	SODOCK	AutoDock	Glide	GOLD	FlexX	Surflex
1bma	17	0.102	0.508	0.514	0.102	0.106	0.111	0.279	0.931	n/a	1.341	0.100
1tmn	15	0.154	0.150	0.577	0.221	0.189	0.237	0.437	0.280	0.168	0.086	0.130
1atl	13	0.303	0.461	0.455	0.079	0.087	0.091	0.130	0.094	n/a	0.206	0.701
1fkg	13	0.148	0.668	0.675	0.116	0.110	0.106	0.130	0.125	0.181	0.759	0.181
2sim	10	0.114	0.115	0.117	0.109	0.121	0.144	0.157	0.092	0.092	0.199	0.110
6rnt	8	0.075	0.051	0.759	0.214	0.209	0.304	0.214	0.222	0.120	0.479	0.703
1lst	7	0.076	0.101	0.104	0.052	0.048	0.053	0.052	0.014	0.087	0.071	0.033
2cgr	7	0.032	0.032	0.033	0.081	0.082	0.081	0.087	0.038	0.099	0.353	0.163
1lah	6	0.027	0.032	0.078	0.040	0.037	0.035	0.010	0.013	n/a	0.028	0.030
1lcp	6	0.119	0.121	0.286	0.079	0.067	0.081	0.074	0.198	n/a	0.165	0.201
1cbx	5	0.156	0.156	0.165	0.102	0.213	0.712	0.133	0.036	0.054	0.135	0.070
1tni	5	0.353	0.340	0.299	0.202	0.353	0.492	0.261	0.218	n/a	0.271	0.297
1hsl	4	0.156	0.155	0.154	0.035	0.037	0.032	0.110	0.131	0.097	0.059	0.051
2ctc	4	0.089	0.070	0.147	0.092	0.091	0.133	0.133	0.161	0.032	0.197	0.038
2cht	3	0.089	0.089	0.089	0.062	0.062	0.062	0.260	0.042	0.059	0.458	0.042
1fki	2	0.025	0.025	0.025	0.080	0.080	0.080	0.080	0.192	0.071	0.059	0.070
1tng	2	0.188	0.188	0.189	0.070	0.053	0.232	0.062	0.019	n/a	0.193	0.022

TABLE V. COMPARISON OF AVERAGE RMSDS AND SUCCESS RATES OF VARIOUS DOCKING PROGRAMS

Method	Average RMSD (nm)	Success rate	Average RMSD < 0.2 nm
PSOVina^{2LS}	0.129	**88.24%**	0.085
FIPSDock	**0.102**	82.35%	0.059
PSO@AutoDock	0.114	82.35%	0.063
Glide	0.165	76.47%	0.068
Surflex	0.173	76.47%	**0.055**
PSOVina	0.192	76.47%	0.076
AutoDock	0.153	70.59%	0.068
SODOCK	0.176	70.59%	0.053
GOLD (11 cases)	**0.096**	64.71%	0.062
FlexX	0.298	58.82%	0.070
AutoDock Vina	0.273	58.82%	0.065

REFERENCES

[1] D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath, “Docking and scoring in virtual screening for drug discovery: methods and applications,” Nat. Rev. Drug Discov., vol. 3, pp. 935-949, 2004.

[2] D. S. Goodsell and A. J. Olson, “Automated docking of substrates to proteins by simulated annealing,” Proteins: Struct., Funct., Genet., vol. 8, pp. 195-202, 1990.

[3] C. McMartin and R. S. Bohacek, “QXP: Powerful, rapid computer algorithms for structure-based drug design,” J. Comput. Aid. Mol. Des., vol. 11, pp. 333-344, 1997.

[4] G. Jones, P. Willett, R C. Glen, A. R. Leach, and R. Taylor, “Development and validation of a genetic algorithm for flexible docking,” J. Comput. Aid. Mol. Des., vol. 267, pp. 727-748, 1997.

[5] H. M. Chen, B. F. Liu, H. L. Huang, S. F. Hwang, S. Y. Ho, “SODOCK: swarm optimization for highly flexible protein-ligand docking,” J. Comput. Chem., vol. 28, pp. 612-623, 2007.

[6] V. Namasivayam and R. Günther, “pso@autodock: a fast flexible molecular docking program based on swarm intelligence,” Chem. Biol. Drug. Des., vol. 70, pp. 475-784, 2007.

[7] Y. Liu, L. Zhao, W. Li, D. Zhao, M. Song, Y. Yang, “FIPSDock: a new molecular docking technique driven by fully informed swarm optimization algorithm,” J. Comput. Chem., vol. 34, pp. 67-75, 2013.

[8] M. C. K. Ng, S. Fong, and S. W. I. Siu, “PSOVina: the hybrid particle swarm optimization algorithm for protein-ligand docking,” J. Bioinform. Comput. Biol., vol. 13, 1541007, 2015.

[9] 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, pp. 455-461, 2010.

[10] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Bellew, and A. J. Olson, “Automated docking using a Lamarckian genetic algorithm and an empirical binding free enregy function,” J. Comput. Chem.,vol. 19, pp. 1639–1662, 1998.

[11] J. A. Nelder and R. Mead, “A simplex method for function minimization,” Comput. J., vol. 7, pp. 308–313, 1965.

[12] O. Korb, T. Stützle, and T. E. Exner, “An ant colony optimization approach to flexible protein-ligand docking,” Swarm Intell., vol. 1, pp. 115–134, 2007.

[13] F. Solis and R. Wets, “Minimization by random search techniques,” Math. Ops. Res., vol. 6, pp. 19–30, 1981.

[14] D. T-H. Chang, Y-J. Oyang, and J-H. Lin, “MEDock: a web server for efficient prediction of ligand binding sites based on a novel optimization algorithm,” Nucleic Acids Res., vol. 33, W233-W238, 2005.

[15] J. Tavares, S. Mesmoudi, and E.-G. Talbi, “The impact of local search on protein-ligand docking optimization,” The 8th Intl. Conf. on Hybrid Intell. Syst.(HIS 2008), pp.53–58, 2008.

[16] J. Tavares, S. Mesmoudi, and E.-G. Talbi, “On the efficiency of local search methods for the molecular docking problem,” EvoBIO LNCS, vol. 5483, pp.104–115, 2009.

[17] R. Wang, X. Fang, Y. Lu, S. Wang, “The PDBbind Database: Collection of Binding Affinities for Protein-Ligand Complexes with Known Three-Dimensional Structures,” J. Med. Chem., vol. 47, pp. 2977-2980, 2004.

[18] J. W. M. Nissink, C. Murray, M. Hartshorn, M. L. Verdonk, J. C. Cole, and R. Taylor, “A new test set for validating predictions of protein-ligand interaction,” Proteins, vol. 49, pp. 457–71, 2002.