template
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Template

Accelerating AutoDock Vina with GPUs

Shidi Tang [1,2], Ruiqi Chen [3], Mengru Lin [3], Qingde Lin [4], Yanxiang Zhu [2], Ji Ding [1,2], Jiansheng Wu [1,2,*], Haifeng Hu [5], and Ming Ling [4]

[1]School of Geographic and Biological Information, Nanjing University of Posts and Telecommunications, Nanjing, 210023, China,
[2]Smart Health Big Data Analysis and Location Services Engineering Research Center of Jiangsu Province, Nanjing University of Posts and Telecommunications, Nanjing, 210023, China,
[3]VeriMake Research, Nanjing Renmian Integrated Circuit Technology Co., Ltd., Nanjing, 210088, China,
[4]National ASIC System Engineering Technology Research Center, Southeast University, Ltd., Nanjing, 210096, China,
[5]School of Telecommunication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, 210023, China

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: AutoDock Vina is one of the most popular molecular docking tools. In the latest benchmark CASF-2016 for comparative assessment of scoring functions, AutoDock Vina won the best docking power among all the docking tools. Modern drug discovery is facing the most common scenario on large virtual screening of drug hits from huge compound databases. Due to the seriality characteristic of the AutoDock Vina algorithm, there is no successful report on its parallel acceleration with GPUs. Current acceleration of AutoDock Vina typically relies on the stack of computing power as well as the allocation of resource and tasks, such as the VirtualFlow platform. The vast resource expenditure and the high access threshold of users will greatly limit the popularity of AutoDock Vina and the flexibility of usage in modern drug discovery.

Results: We proposed a new method Vina-GPU for accelerating AutoDock Vina with GPUs, which is greatly needed for reducing the investment for large virtual screens, and also for a wide application in large-scale virtual screening on personal computers, station servers or cloud computing etc. Our proposed method Vina-GPU greatly raises the number of initial random conformations and reduces the search depth of each thread, and then a heterogeneous OpenCL implementation was developed to realize its parallel acceleration leveraging thousands of GPU cores. Large benchmark tests show that Vina-GPU reaches an average of 21-fold and a maximum of 50-fold docking acceleration against the original AutoDock Vina while ensuring their comparable docking accuracy, indicating its potential of pushing the popularization of AutoDock Vina in large virtual screens.

Availability: The source code and tools of Vina-GPU are freely available at http://www.noveldelta.com/Vina_GPU.

Contact: jansen@njupt.edu.cn

Supplementary information: Supplementary data are available at another file

1 Introduction

Molecular docking studies how two or more molecular structures (e.g., drug and target) fit together. Molecular docking analysis has become one of the most common ways for modern drug discovery (Meng *et al.*, 2011). It allows to predict molecular interactions where a protein and a ligand inducted fit together in the bound state. Also, molecular docking tools provide an efficient and cheap way in the early stage of drug design for the identification of leading compounds and their binding affinities (Lengauer and Rarey, 1996; Cherkasov *et al.*, 2014; Golbraikh *et al.*, 2003). Among all molecule docking tools, the AutoDock suite is the most popular, which consists of various tools including AutoDock4 (Morris *et al.*, 2009), AutoDock Vina (Trott and Olson, 2010), AutoDock Vina 1.2.0 (Eberhardt *et al.*, 2021), AutoDock FR (Ravindranath *et al.*, 2015), AutoDock Crank Pep (Zhang and Sanner, 2019a,b), AutoDock-GPU (Santos-Martins *et al.*, 2019, 2021) etc. AutoDock Vina is usually recommended as the first-line tool in the implementation of molecular docking due to its docking speed and accuracy (Goodsell *et al.*, 2021). Moreover, it wins the best docking power in the last benchmark CASF-2016 for comparative assessment of scoring functions (Su *et al.*, 2018) and the best scoring power under the comparison with ten docking programs on a diverse protein–ligand complex sets (Wang *et al.*, 2016). AutoDock Vina uses a Monte-Carlo iterated local search method, which comprises iterations of sampling, scoring and optimization. First, an initial random conformation is sampled for a given

ligand, which is represented by its position, orientation and torsion (POT). Then, its position, orientation or torsion is randomly mutated with a disturbance. Finally, the affinity is evaluated for the binding pose of a ligand and a protein. In AutoDock Vina, the binding affinity is calculated by a scoring function which describes the sum of the intermolecular energy (ligand-receptor) and the intramolecular energy (ligand-ligand). Moreover, the conformation is optimized with a Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Fletcher, 2013) that considers the gradients of the scoring function. These gradients can guide the ligand to achieve a better conformation with a lower docking score. In addition, a metropolis acceptance rule, which relies on the difference between the docking score of the initial conformation and that of the optimized conformation, is arranged to decide whether the optimized conformation can be accepted or not. The accepted conformation will be recorded as the initial conformation and further optimized in the next iterations. As all we know, the Monte-Carlo based iterated local search method in AutoDock Vina is highly serialized because the ongoing iteration depends on the previous outputs.

Preceding virtual screens pipeline typically operates only on a scale of $10^6 \sim 10^7$ compound molecules. Such scale of compounds will heavily descend the capability and increase the failure risk of modern drug discovery. Fortunately, the whole chemical space of drug-like molecules has been estimated to reach more than 10^{60} (Bohacek *et al.*, 1996). The scale of compounds in virtual screens is vital since the more candidate compounds to be screened, the lower rate of failure and the more favorable quality of leading compounds can be reached. Hence, the virtual screens on huge compound databases are urgent for identifying excellent drug candidates in modern drug discovery. However, original virtual screening with AutoDock Vina on huge databases is very slow, which cannot meet the need for modern drug discovery. Therefore, an acceleration of AutoDock Vina has become a central problem in current virtual screens of drug hits from huge compound databases. Till now, there are several attempts for the acceleration of AutoDock Vina in large virtual screens (Gorgulla *et al.*, 2020; Li *et al.*, 2012; Jaghoori *et al.*, 2016). For instance, VirtualFlow provides a drug discovery platform that speeds up AutoDock Vina in virtual screening of an ultra-large database with more than 1.4 billion molecules by leveraging over 16,0000 CPUs (Gorgulla *et al.*, 2020). Such huge resource investment and expenditure, as well as the high entry threshold for users seriously weaken the popularization of AutoDock Vina and the flexibility of customer's usage (such as a self-defined target and small molecule dataset). Due to the overall serial design of the AutoDock Vina algorithm, its parallelization mostly depends on the stacking of computing powers as well as the allocation of resources and tasks under such a common scenario that facing large virtual screens in modern drug discovery. A reduction of computational resource investment and user access threshold will advance the broad spread of AutoDock Vina in large virtual screening for modern drug discovery.

Graphic processing unit (GPU) is a powerful parallel programmable processor with thousands of computing cores that provide a tremendous computational performance. GPU has become an integral part of mainstream computing systems due to the high price-performance ratio and the ease of developing an implementation with well-established standards such as Compute Unified Device Architecture (CUDA) and Open Computing Language (OpenCL). GPU has been applied to accelerate molecular docking in several tools (Santos-Martins *et al.*, 2019, 2021; Mermelstein *et al.*, 2018; Hwu, 2011; Stone *et al.*, 2016; LeGrand *et al.*, 2020; Fan *et al.*, 2021; Ding *et al.*, 2020; Imbernón *et al.*, 2021; Shin *et al.*, 2020; Solis-Vasquez *et al.*, 2020; Kannan and Ganji, 2010). For example, AutoDock-GPU provides an OpenCL implementation of AutoDock4 to exploit both GPU and CPU parallel architectures. By exploring three levels of parallelism (runs, individual, fine-grained tasks) on the Lamarckian Genetic Algorithm (LGA) algorithm, AutoDock-GPU achieves the maximum of 350-fold acceleration on total runtime against the single-threaded CPU implementation (Santos-Martins *et al.*, 2019). Recently, an attempt has been put into the GPU parallel acceleration of AutoDock Vina where the Viking method tried to rewrite the pose search stage of AutoDock Vina on GPU (Shin *et al.*, 2020). Till now, no positive acceleration result has been reported. The reasons for its deficiency probably involve the following three points. Firstly, the Monte-Carlo based optimization process in AutoDock Vina is the most time-consuming (typically more than 90%) and highly dependent whose next iteration relies on the previous outputs. Secondly, each ligand file was presented as a heterogeneous tree structure whose nodes are traversed recursively. Thirdly, the CUDA architecture on NVIDIA GPU cards limits its cross-platform portability.

In this work, we proposed an efficient parallel method, namely Vina-GPU, to accelerate AutoDock Vina with GPUs. First, Vina-GPU applies a large-scale parallelism on the Monte-Carlo based iterated docking threads and significantly reduces the search depth in each thread. Second, a heterogeneous OpenCL implementation was efficiently deployed on Vina-GPU by converting the heterogeneous tree structure into a list structure whose nodes are stored in the traversed order. The two implementations ensure that Vina-GPU can leverage thousands of computational cores on GPU and achieve a large-scale parallelization and acceleration, and realizes the cross-platform portability on both CPUs and GPUs. Large benchmark tests show that Vina-GPU reaches an average of 21.66X and a maximum of 50.80X speed-up on NVIDIA Geforce RTX 3090 against the original AutoDock Vina on a 20-threaded CPU while ensuring their comparable docking accuracy. To further enlarge its potential of pushing the popularity of AutoDock Vina in large virtual screening, more efforts have been taken as follows. First, we fitted a heuristic function for automatically determining the most important hyperparameter ($search\_depth$) on the basis of large testing experiments in order to lower the usage threshold. Second, we developed a user-friendly graphical user interface (GUI) for a convenient operation of Vina-GPU. Third, we enable the implementation of Vina-GPU to be built on Windows, Linux and macOS, ensuring their usability on personal computers, station servers and cloud computations etc. The code and tool of Vina-GPU are freely available at https://github.com/DeltaGroupNJUPT/Vina-GPU or http://www.noveldelta.com/Vina_GPU.

2 Methodology

The heterogeneous OpenCL implementation of Vina-GPU is depicted in Figure 1, which consists of a host part (on CPU) and a device part (on GPU). The host part is mainly in charge of the preparation and post-refinement of the conformations. The device part focuses on the acceleration of the most time-consuming Monte-Carlo iterated local search method by enlarging the scale of parallelism as well as reducing the number of iterations.

2.1 Host part

The host part consists of two sections (see Figure 1). The first section includes four operations, which are the file reading, the OpenCL setup, the data preparation and the device memory allocation, and all operations are implemented for the input to the device part. Specifically, the file reading operation is to read the ligand and protein files in .pdbqt format, and the OpenCL setup operation is to setup the OpenCL environment (platform, device, context, queue, program and kernels). Furthermore, the host part prepares all the required data, including grid cache (for calculating the energy of a conformation), random maps (for generating probability random numbers) and random initial conformations (for Monte-Carlo based method to start from). The data is then re-organized to load in the device memory according to how it is accessed (read-only or read-write). The read-only grid cache, random maps and random conformations are allocated in the constant device memory while the read-write of best conformations returned by the device part is allocated in the global device memory. Such kind of memory management could efficiently boost the speed of reading and writing on GPU. The second section includes multiple operations after the device part. All the best conformations returned from the device part are clustered and sorted in the container by their docking scores. The best 20 conformations will be concretely refined and optimized before generating the final output ligand files.

In the data preparation operation, AutoDock Vina treats each conformation as a heterogeneous tree structure whose nodes are stored with its frame information and a pointer to its children node. Each node is traversed by a depth-first search policy to calculates the conformation energy in a recursive process. However, in Vina-GPU, the OpenCL standard cannot support any recursion in kernels because the allocation of stack space for thousands of threads is too expensive. Besides, various ligands would generate different heterogeneous trees that are not suitable for the OpenCL implementation. Therefore, we transformed the heterogeneous tree structure into a list type (see Figure 2), each of whose node is stored in line with its traversed order. These nodes can be traversed simply by the order of the node list. In addition, a children map was created to denote the relationship among these nodes. For example, the node 0 has two children-nodes (the node 1 and the node 4) and so the row 0 has two "T"s (indicating "True") in the 1^{st} and 4^{th} column (Figure 2). Thus, the recursive

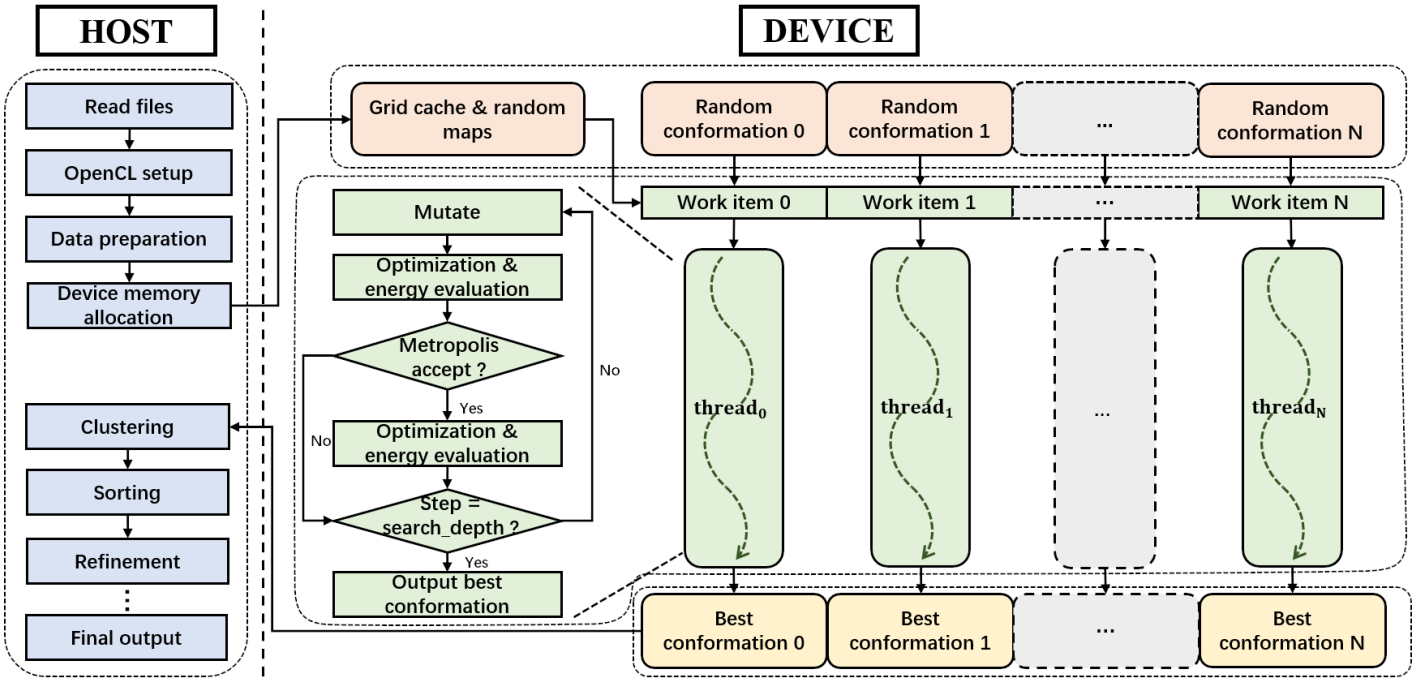


Fig. 1. The OpenCL architecture for implementing Vina-GPU, which consists of a host (CPU) and a device (GPU) part of execution. The device part implements thousands of docking threads, each of which is assigned with an OpenCL work item to perform a Monte-Carlo based local search method with largely reduced search iterations.

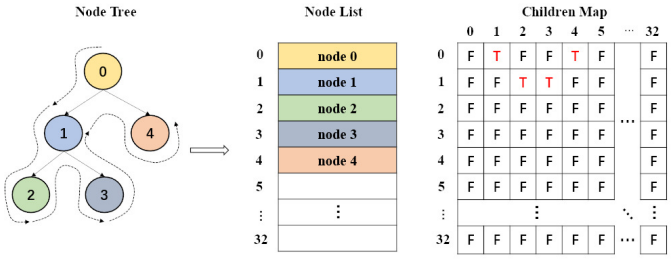


Fig. 2. Transformation the original node tree structure into the node-list format. The heterogeneous node tree was reconstructed in its traversed order(depth-first) into the node list where an additional children map was built to reflect the relationship among the nodes.

traverse of the heterogeneous tree can be converted into an iterative traverse of the node list and children map which fits the OpenCL standard.

2.2 Device part

On the device part, the allocated constant memory (highlighted in orange in Figure 1) is assigned for the initialization and the calculation during the reduced-step Monte-Carlo iterated local search processes (highlighted in green in Figure 1) and the final best conformations are stored in global memory (highlighted in gold in Figure 1).

Vina-GPU enables thousands of reduced-steps iterated local search processes running concurrently within the GPU computational cores. We denote each reduced-step iterated local search process as a docking thread. Within each thread, an OpenCL work item is assigned to a randomly initialized conformation $\mathbf{C}$, which can be represented by its position, orientation and torsion (POT):

$$\mathbf{C} = \{x, y, z, a, b, c, d, \psi_1, \psi_2, ..., \psi_{N_{rot}}\} \tag{1}$$

where x, y, z correspond to the position of the conformation in a pre-determined searching space; a, b, c, d denote its orientation as a rigid body in the quaternion form; $\psi_1, \psi_2, ..., \psi_{N_{rot}}$ represent torsions of N_{rot} rotatable bonds. Then, each conformation $\mathbf{C}$ is to be randomly mutated in one of its POT with the uniform distribution. The conformation will be continuously evaluated with a scoring function that quantifies the free energy of the binding pose. Generally, the free energy e is calculated with the sum of intermolecular energy and intramolecular energy:

$$e = e_{inter} + e_{intra} \tag{2}$$

where e_{inter} represents the interaction energy between the ligand and the receptor, and it is calculated using trilinear interpolation that approximates the energy of each atom pair by looking up the grid cache; and e_{intra} indicates the interaction energy of the pairwise atoms within the ligand. Considering that both e_{inter} and e_{intra} are related to the binding pose, the scoring function $\mathbf{SF}$ can be denoted as a function of POT variables:

$$\mathbf{SF} = f(x, y, z, a, b, c, d, \psi_1, \psi_2, ..., \psi_{N_{rot}}) \tag{3}$$

After the energy evaluation, a Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Fletcher, 2013) optimization is applied to update the ligand conformation by minimizing of the scoring function $\mathbf{SF}$. Essentially, the BFGS method is to substitute the hessian matrix $\mathbf{H} \in \mathbb{R}^{(7+N_{rot})\times(7+N_{rot})}$ with an approximate matrix $\mathbf{B} \in \mathbb{R}^{(7+N_{rot})\times(7+N_{rot})}$ whose inverse matrix $\mathbf{B}^{-1}$ is iteratively updated by the first-order derivatives $\nabla\mathbf{SF}(\mathbf{C}) \in \mathbb{R}^{(7+N_{rot})}$. $\mathbf{B}_{k+1}^{-1}$ in the $(k+1)^{th}$ iteration can be calculated by

$$\mathbf{B}_{k+1}^{-1} = \mathbf{B}_k^{-1} + \frac{\left(\mathbf{s}_k^T\mathbf{y}_k + \mathbf{y}_k\mathbf{B}_k^{-1}\mathbf{y}_k\right)\left(\mathbf{s}_k\mathbf{s}_k^T\right)}{\left(\mathbf{s}_k^T\mathbf{y}_k\right)^2} - \frac{\mathbf{B}_k^{-1}\mathbf{y}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{y}_k^T\mathbf{B}_k^{-1}}{\mathbf{s}_k^T\mathbf{y}_k} \tag{4}$$

where

$$\mathbf{s}_k = -\alpha_k\mathbf{B}_k^{-1}\nabla\mathbf{SF}\left(\mathbf{C}_k\right) \tag{5}$$

$$\alpha_k = argmin\ \mathbf{SF}\ \left(\mathbf{C}_k\ + \alpha\mathbf{p}_k\right) \tag{6}$$

$$\mathbf{y}_k = \nabla\mathbf{SF}\left(\mathbf{C}_k\ + \alpha_k\mathbf{p}_k\right) - \nabla\mathbf{SF}\left(\mathbf{C}_k\right) \tag{7}$$

where $\mathbf{B}_0$ is initiated with identity matrix $\mathbf{E}$ and the detailed calculation of $\nabla\mathbf{SF}(\mathbf{C})$ is described in (Trott and Olson, 2010). Next, a metropolis acceptance criterion is

adopted to decide whether to accept the optimized conformation or not, by comparing the energy e_0 before the mutation and the energy e_{opt} after the optimization. Here, the accept probability P is represented by:

$$P = \begin{cases} 1 & e_0 > e_{opt} \\ \frac{\exp(e_0 - e_{opt})}{1.2} & e_0 \leq e_{opt} \end{cases} \quad (8)$$

It indicates that the accepted conformation is more likely to have a lower energy. Once accepted, the conformation will be further evaluated and optimized by BFGS. Then, the next iteration continues to update the previous optimized conformations until convergence. Finally, all the best conformations found by work items are returned to the host part. The pseudocode of our proposed algorithm Vina-GPU is shown in Supplementary Algorithm S1.

3 Results and Discussion

3.1 Experimental Settings

All 140 complexes in the AutoDock-GPU study (Santos-Martins *et al.*, 2021) are assigned as our experimental dataset, which is comprised of 85 complexes from the Astex Diversity Set (Hartshorn *et al.*, 2007), 35 complexes from CASF-2013 (Li *et al.*, 2014), and 20 complexes from the Protein Data Bank (Berman *et al.*, 2000). They cover a wide range of ligand complexities and targets properties. Each complex file includes an X-ray structure, an initial random pose of its ligand and the corresponding receptor (in .pdbqt format). Besides, we created a config.txt file for each complex (see the example in Supplementary Table S1), which involves the center (indicated by $center_x, center_y, center_z$) and the recommended volume of the docking box (indicated by $size_x, size_y, size_z$). We classified the 140 complexes into three subsets by their atom sizes N_{atom} (small: 5-23 atoms, medium: 24-36 atoms, large: 37-108 atoms). The details of our experimental data can be seen in Supplementary Table S2.

AutoDock Vina was executed on Intel (R) Core (TM) i9-10900K CPU @ 3.7 GHz using Windows 10 Operating System with 64 GB RAM. AutoDock Vina was customized by several configurable arguments, including the center and the volume of searching spaces, the number of CPU cores (*cpu*) to be utilized and the docking runs ($exhaustiveness$) etc. The argument $exhaustiveness$ was set to 128 (Handoko *et al.*, 2012) , and the argument *cpu* was set to the maximum value of 20 for taking a full use of the CPU computational power.

Vina-GPU was developed with OpenCL v.3.0 and executed on three different GPUs (Nvidia Geforce GTX 1080Ti, Nvidia Geforce RTX 2080Ti, Nvidia Geforce RTX 3090) under single-precision floating-point format (FP32). Details are included in Supplementary Table S3. In our Vina-GPU, we replaced *cpu* and $exhaustiveness$ with the number of threads ($thread$) and the size of searching iterations in each thread ($search\_depth$). These two hyperparameters are of the most importance, and their values are vital to the docking performance of Vina-GPU. For a convenient usage of Vina-GPU, as in AutoDock Vina (Trott and Olson, 2010), a heuristic formula was fitted to automatically determine the proper size of $search\_depth$ for a given complex. Specifically, a large number of tests were executed on all 140 complexes to examine their docking performance under various sizes of $search\_depth$, where the proper $search\_depth$ that guarantees a comparable docking performance was selected. Then, the least squares method was used to fit an empirical formula of the proper $search\_depth$ with respect to the N_{atom} (the number of atoms) and N_{rot} (the number of rotatable bonds) in a ligand. The heuristic formula is given as follows,

$$\begin{aligned} search\_depth = max(1, floor(0.24 * N_{atom} \\ + 0.29 * N_{rot} - 3.41)) \end{aligned} \quad (9)$$

where the function $floor(*)$ gives the largest integer less than or equal to $*$.

3.2 Influence of hyperparameters

We evaluated the influence of the hyperparameters $thread$ (from 100 to 15000) and $search\_depth$ (from 1 to 50) on the docking accuracy (evaluated by docking score and RMSD) as well as the docking runtime of Vina-GPU. The docking score represents the binding affinity between a ligand and a receptor (the lower the score is better) and the RMSD measures the atom distance difference between an output conformation and the ground truth X-ray structure (also the lower the better) (Trott and Olson, 2010). An acceptable docking is defined if the least RMSD among all output conformations of Vina-GPU is smaller than 2 Å(Goodsell *et al.*, 2021). Three complexes (5tim, 2bm2 and 1jyq) were randomly selected, which represent various levels of complexities (small, medium and large). The influence of $thread$ and $search\_depth$ on the docking score, RMSD and docking runtime are shown in Figure 3 and Supplementary Figure S1, respectively. All experiments were executed under NVIDIA Geforce RTX 3090 GPU card.

With the increase of $thread$, the docking score gets better and it becomes convergence when the size of $thread$ reaches around 6000 for 2bm2 and 1jyq, and about 1000 for 5tim (Figure 3a). The same trend is also observed on the RMSD performance (Figure 3b), where 2bm2 and 1jyq converges at around 8000, and 5tim fluctuates slightly nearby 2 Å. In Figure 3c, with the raise of $thread$, the docking runtimes of all three complexes increase slowly. Although it is enough for the small complex 5tim to obtain the best docking accuracy with 1000 $thread$, the size of $thread$ needs to be set around 8000 for the medium complex 2bm2 and large complex 1jyq. Thus, the size of $thread$ was set to be 8000 for all 140 complexes in this paper.

For the small complex 5tim, with the increase of $search\_depth$, its docking score, RMSD and docking runtime keep steady. The size of $search\_depth$ does not influence the docking results, because Vina-GPU can achieve the best performance with a few $search\_depth$ for such a small complex (Supplementary Figure S1). For the medium complex 2bm2, the docking score and RMSD converge quickly with the raise of $search\_depth$, and the docking runtime increases slowly. This is because a medium complex needs more $search\_depth$ to reach the convergence (Supplementary Figure S1). For the large complex 1jyq, the docking score and RMSD converges slowly with $search\_depth$. The docking runtime for 1jyq increases rapidly with $search\_depth$, because the device runtime for such a large complex 1jyq takes the major part of the total (host + device) docking runtime, increasing $search\_depth$ leads to a great expense on the total docking runtime.

3.3 Docking Accuracy

We compare the overall docking accuracy of Vina-GPU with AutoDock Vina in terms of the docking score and RMSD performances on all 140 complexes (Figure 4). The color bar encodes the number of atoms in a ligand. For the docking score, most complexes distribute around the diagonal line and fall into the lavender margin of 0.5 kcal/mol difference and their Pearson correlation coefficient of the scores is 0.965 (Figure 4a), which denotes a significant positive correlation. The results show that our Vina-GPU achieves the comparable docking scores with AutoDock Vina.

A docking conformation is typically acceptable when its RMSD difference with the ground truth structure is smaller than 2 Å(Santos-Martins *et al.*, 2021). In Figure 4b, the red dashed line distinguishes whether a docking conformation is acceptable or not from the RMSD aspect. Figure 4b demonstrates that most complexes fall into the lower left region where both Vina-GPU and AutoDock Vina succeed to obtain the acceptable docking. The results show that our Vina-GPU achieves the comparable docking RMSD with AutoDock Vina. Thus, these findings indicate that Vina-GPU exhibits the comparable docking accuracy with respect to AutoDock Vina on both docking score and RMSD.

3.4 Runtime Comparison

The runtime acceleration (Acc) of Vina-GPU against AutoDock Vina is defined by

$$Acc = \frac{t_{vina}}{t_{vina-gpu}} \quad (10)$$

where t_{vina} and $t_{vina-gpu}$ is the runtime of AutoDock Vina and Vina-GPU, respectively. Figure 5 shows the runtime acceleration (Acc) on various scales of complexity (small: 5-23 atoms, medium: 24-36 atoms, large: 37-108 atoms) and different GPU cards (Nvidia Geforce GTX 1080Ti, Nvidia Geforce RTX 2080Ti, Nvidia Geforce RTX 3090). The average acceleration is highlighted by a white dot in the center.

As indicated in Figure 5, Vina-GPU achieves the maximal acceleration of 50.80X, as well as the average of 8.84X, 12.70X and 21.66X on the 1080ti, 2080ti and 3090 GPU cards, respectively. The results show that the average acceleration increases along with the complexity of the complex (from small to large) and also raises with higher end GPU cards (from NVIDIA Geforce GTX 1080ti to NVIDIA Geforce GTX 3090). Figure 6 shows the Acc performance of all 140 complexes along with different

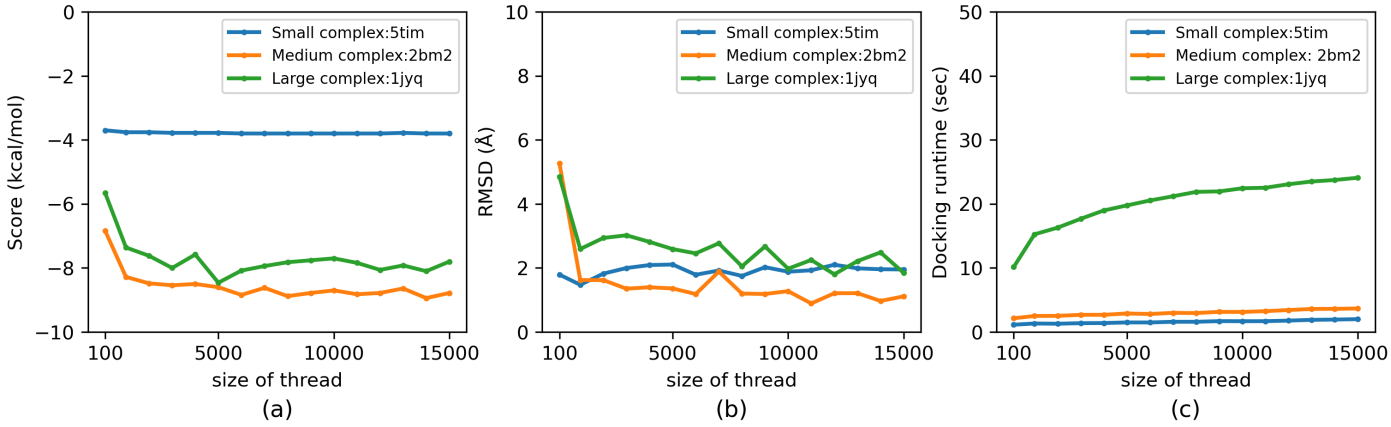


Fig. 3. Influence of the size of $thread$ on docking accuracy (score and RMSD) and docking runtime of Vina-GPU. Three typical PDB complexes are randomly selected from all 140 complexes which represent small, medium and large ones, respectively (5tim: small, 5 atoms; 2bm2: medium, 33 atoms; 1jyq: large, 60 atoms). All experiments were executed on NVIDIA RTX 3090 GPU card.

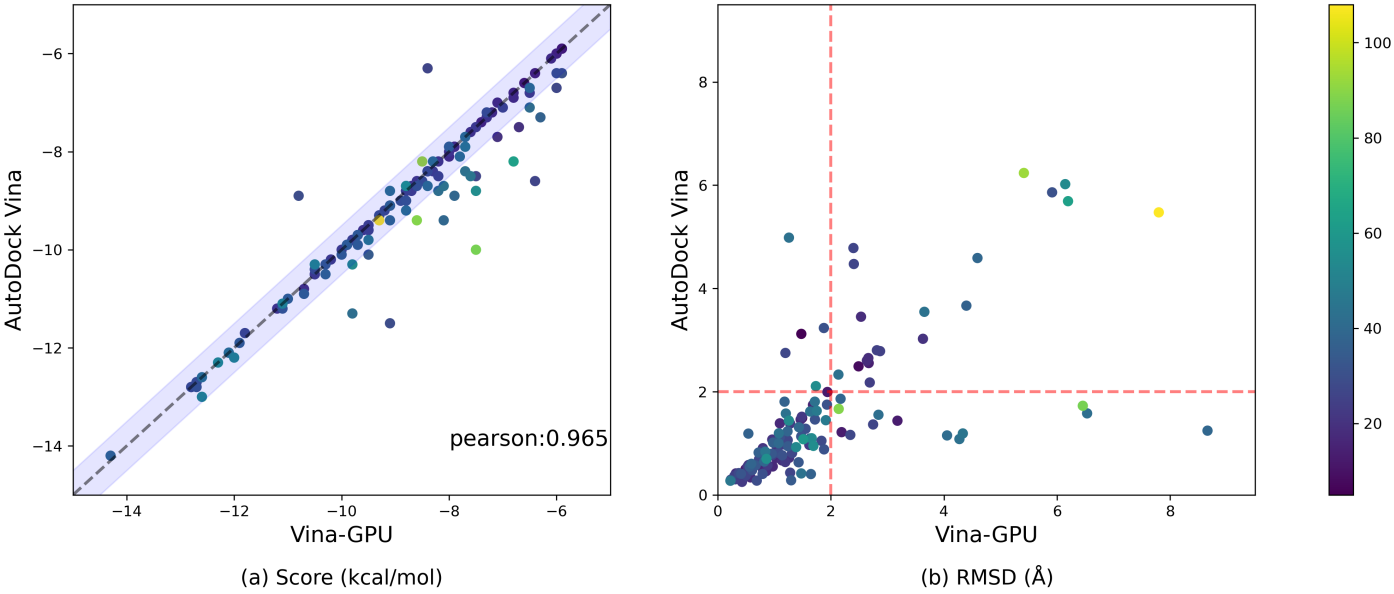


Fig. 4. Comparable docking accuracy between AutoDock Vina and our Vina-GPU on all 140 complexes. The color bar encodes the number of atoms in a ligand. A margin of 0.5 kcal/mol difference on the docking score between Vina-GPU and AutoDock is highlighted with lavender in Figure 4a. The Pearson correlation coefficient of their docking scores is 0.965 (indicated by "pearson"). The RMSD value that indicates an acceptable binding pose (< 2 Å) are separated by a red dashed line in Figure 4b.

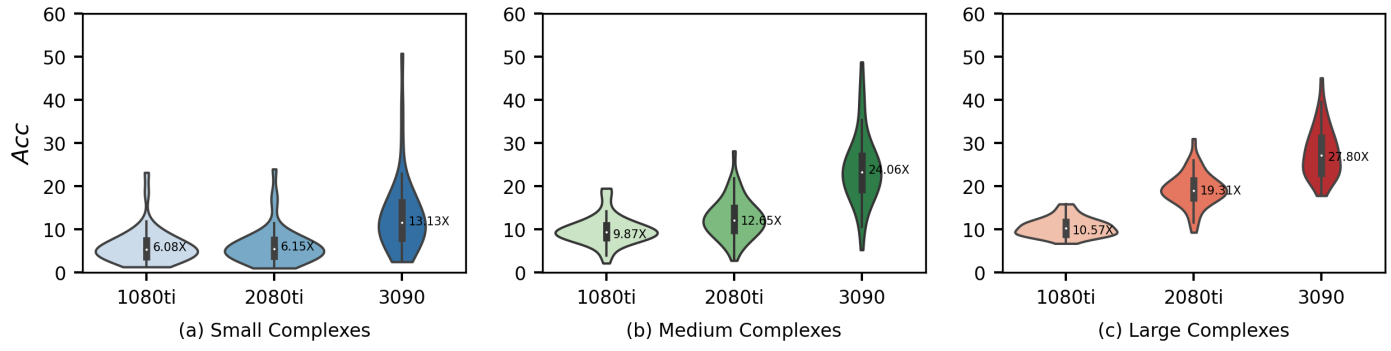
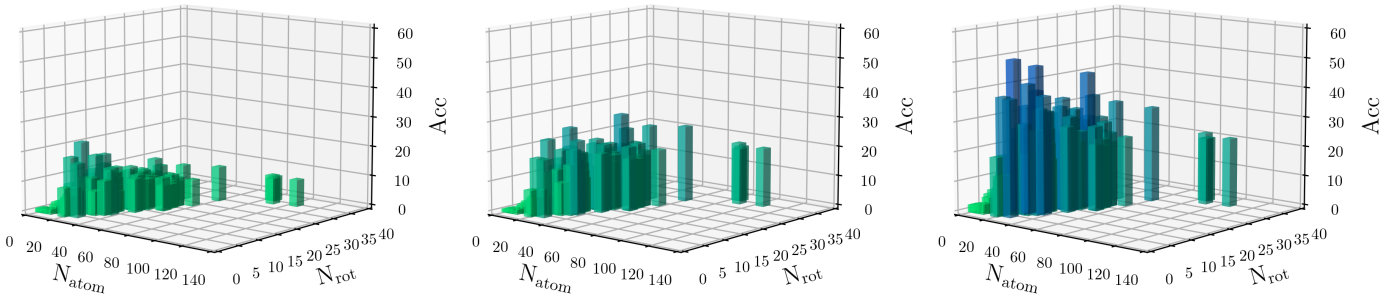


Fig. 5. Acceleration of docking time (Acc) of our Vina-GPU against AutoDock Vina on three different GPUs and various scales of complexity (small: 5-23 atoms, medium: 24-36 atoms, large: 37-108 atoms). 1080ti: NVIDIA Geforce GTX 1080ti; 2080ti: Nvidia Geforce RTX 2080Ti; 3090: Nvidia Geforce RTX 3090.



(a) Vina-GPU 1080ti (b) Vina-GPU 2080ti (c) Vina-GPU 3090

Fig. 6. Details for the acceleration of docking time (Acc) of our Vina-GPU against AutoDock Vina on all 140 complexes. The complexity is depicted with their number of atoms (N_{atom}) and rotatable bonds (N_{rot}). The vertical axis ranges from 0 to 60, and each bar represents a complex coupling with its corresponding acceleration (Acc). Vina-GPU 1080ti, Vina-GPU 2080ti and Vina-GPU 3090 mean that Vina-GPU was executed on Nvidia Geforce GTX 1080ti, Nvidia Geforce RTX 2080Ti and Nvidia Geforce RTX 3090, respectively.

N_{atom} and N_{rot}. Each bar represents a complex coupling with its corresponding acceleration. As shown in Figure 6, the acceleration varies from 1.03X to 50.80X. The maximal acceleration 50.80X is achieved on the 1xm6 (PDBid) complex under Nvidia Geforce RTX 3090 GPU card.

Among the whole Vina-GPU program, the Monte-Carlo based optimization process is the most time-consuming part (typically more than 90%), which is performed in the "device" part utilizing GPU computational cores. For a better exhibition of the acceleration in the most time-consuming part, we defined the device runtime acceleration Acc_d as

$$Acc_d = \frac{t_{mc}}{t_d} \tag{11}$$

where t_{mc} is the Monte-Carlo based optimization part runtime of AutoDock Vina and t_d is the device part runtime of Vina-GPU. As shown in Supplementary Figure S2 and Supplementary Figure S3, Vina-GPU achieves the maximum of 191.68X and the average of 18.94X, 43.58X and 48.48X acceleration on Nvidia Geforce GTX 1080ti, Nvidia Geforce RTX 2080Ti and Nvidia Geforce RTX 3090 GPU cards, respectively.

3.5 Conformation Spaces Analysis

To verify their equivalence in molecular docking, we intend to analyze the full conformation spaces explored by AutoDock Vina and our Vina-GPU. Firstly, we discussed the searching strategy of Vina-GPU and explain why Vina-GPU can achieve a great acceleration on the premise of comparable docking accuracy. Then, we visualized and compared their whole searching of conformation spaces.

Vina-GPU enables thousands of docking threads to run concurrently. These docking threads divide the whole search space into thousands of subspaces, and in each subspace an initial conformation is being optimized. We define the search space that covers all possible conformations as a high-dimensional space $\mathbf{S} = \{\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2, \ldots\}$. By dividing $\mathbf{S}$ into n sub-spaces, we have

$$\mathbf{S} = \left\{\mathbf{S}_{sub_0}, \mathbf{S}_{sub_1}, \ldots, \mathbf{S}_{sub_n}\right\} \tag{12}$$

and each initial conformation belongs to a sub-space

$$\mathbf{C}_i \in \mathbf{S}_{sub_i} \quad (i = 0, 1, 2, \ldots) \tag{13}$$

For each initial conformation $\mathbf{C}_i$, the corresponding searching space $\mathbf{S}_{sub_i}$ is much smaller than the whole searching space $\mathbf{S}$. Therefore, we can greatly reduce the searching iterations of each initial conformation in each $\mathbf{S}_{sub_i}$. By clustering and sorting all the best conformations, Vina-GPU ensures a comparable docking accuracy with original AutoDock Vina.

Then, we detailed a case (PDBid: 2bm2, $N_atom = 33$, $N_{rot} = 7$) and visualized their full searching of conformation spaces in Supplementary Figure S4. AutoDock Vina was executed with the configuration of "$cpu = 1, exhaustiveness = 1$ and $search\_depth = 22365$ (default value)". Vina-GPU was executed under various strategies, where different sizes of $thread$ and $search\_depth$ were used. The whole searching spaces ($lanes \times search\_depth$) keep almost the same as that of original AutoDock Vina. All conformations searched by AutoDock Vina or our Vina-GPU are indicated as orange or blue dots, respectively. Each conformation is represented by its POT in cartesian coordinates, where a principal component analysis (PCA) method was used to reduce the dimensionality of orientation and torsion into three. The best conformation is shown in red star (indicated by an arrow).

As shown in Supplementary Figure S4a and Supplementary Figure S4b, the whole conformation space reached by Vina-GPU or AutoDock Vina is almost the same in their position, orientation or torsion. With the increase of Vina-GPU on its parallel threads and the reduce of $search\_depth$ in each thread, these observations keep unchanged (Supplementary Figure S4c and Supplementary Figure S4d). Moreover, the best conformations found by AutoDock Vina or our Vina-GPU are very close to each other. These results demonstrate that our Vina-GPU can achieve comparable docking accuracy with original AutoDock Vina.

3.6 Comparison with the Implementation of Vina-GPU on CPUs

Due to the inherently serial characteristic of the AutoDock Vina algorithm, our Vina-GPU proposed an improved algorithm and then accelerated it with GPUs. For evaluating the contributions of the algorithm improvement and the GPU hardware acceleration separately, we gave out the performance comparison with the implementation of Vina-GPU on CPUs (Supplementary Figure S5 and Supplementary Figure S6). The implementation of Vina-GPU on CPUs was executed on Intel (R) Core (TM) i9-10900K CPU @ 3.7 GHz. Both these implementations were executed on all 140 complexes with the same settings of $thread$ (8000) and $search\_depth$. The results of our Vina-GPU on GPUs are identical to those in Figure 5.

For the docking score, most complexes lie around the diagonal line and fall into the lavender margin of a 0.5 kcal/mol difference, only with a few exceptions due to the randomness of Vina-GPU algorithm (Supplementary Figure S5a). The Pearson correlation coefficient of their docking scores is 0.963 (Supplementary Figure S5a). The results show that the implementation of Vina-GPU on CPU achieves the comparable docking scores. For the docking RMSD, most complexes gather in the bottom left region, which indicates that they are acceptable dockings for the implementations of Vina-GPU on CPUs and our Vina-GPUs (Supplementary Figure S5b). The comparable docking scores and RMSD mean that the implementation of Vina-GPU on CPU obtains the almost same docking accuracy. For the docking runtime, the acceleration of the implementation of Vina-GPU on GPUs is higher than that on CPUs, and the latter one only achieves the maximal acceleration of 26.19X and the average of 3.49X, 8,81X and 18.76X on the small, medium and large complexes against the original AutoDock Vina, respectively (Supplementary Figure S6). These results indicate that our improved algorithm with the implementation on both the CPUs and GPUs gains the comparable docking accuracy, and it is more suitable for the implementation on GPU hardware to achieve higher accelerations.

3.7 A Case for Large Virtual Screening

To show the acceleration effect of our Vina-GPU in implementing real virtual screens of large compound databases, a case was detailed on the receptor 1xm6 (PDBid) with the docking of DrugBank (Wishart *et al.*, 2018). The receptor 1xm6 is the catalytic domain of human phosphodiesterase 4B in complex with (R)-mesopram, and DrugBank is one of the most popular drug databases that contains comprehensive information on drugs and drug targets. A total of 9125 molecules were downloaded from the DrugBank database at https://go.drugbank.com/releases/latest#structures. Both Vina-GPU and AutoDock Vina were executed on the same computer with Intel (R) Core (TM) i9-10900K CPU @ 3.7 GHz and NVIDIA Geforce RTX 3090 GPU card. The $exhaustiveness$ and cpu of AutoDock Vina were set to 128 and 20, respectively. The $thread$ and $search\_depth$ of Vina-GPU were set to 8000 and the heuristic value, respectively. Only ~9.44 hours were taken to execute the whole docking process by Vina-GPU while ~133.90 hours by AutoDock Vina, indicating that the acceleration of 14.18X are achieved by our Vina-GPU. The docking scores of all 9125 molecules on Vina-GPU and AutoDock Vina are shown in Supplementary Data S1. We evaluated the similarity of top i compounds with the lowest docking scores on AutoDock Vina or our Vina-GPU by Jaccard index (Jaccard, 1912) as defined by

$$J_i = \frac{\left|T^i_{vina} \cap T^i_{vina-GPU}\right|}{\left|T^i_{vina} \cup T^i_{vina-GPU}\right|} \tag{14}$$

where $i = 15, 50, 100, 200, 300$, and T^i_{vina} and $T^i_{vina-GPU}$ represents subset of top i compounds of AutoDock Vina and Vina-GPU, respectively. Supplementary Table S4 shows that all the Jacard indexes are larger than 0.8, indicating a high similarity of the docking results of Vina-GPU and AutoDock Vina.

Supplementary Figure S7 shows the comparison of docking scores between Vina-GPU and AutoDock Vina, where most compounds lie around the diagonal line and within the margin (in lavender) of 0.5 kcal/mol difference on the docking score. The Pearson correlation coefficient of their docking scores is 0.981. The results show that our Vina-GPU achieves the highly similar docking scores with AutoDock Vina. In addition, Supplementary Figure S8 visualizes the binding poses of 3 molecules (DrugBank accession number: DB08418, DB07700, DB07270) with the best docking scores (-13.7, -13, -12.8) using Pymol for a better exhibition of real docking results by our Vina-GPU.

3.8 Usage of Vina-GPU

We developed a user-friendly graphic user interface (GUI) instead of the original terminal form. Our GUI can be utilized without installation and is described in Supplementary Figure S9. In addition, we provided a detailed guideline on how to build and run Vina-GPU on mainstream operating systems (Windows, Linux and MacOS), and it can also ensure the usability of Vina-GPU on personal computers, station servers and cloud computations etc (see Supplementary Text S1). All source codes and tools of Vina-GPU can be freely available at http://www.noveldelta.com/Vina_GPU or https://github.com/DeltaGroupNJUPT/Vina-GPU.

4 Conclusion

In modern drug discovery, huge resource investment and high entry threshold seriously weaken the popularity of AutoDock Vina in large virtual screening from compound databases. To advance the wide spread of AutoDock Vina in large virtual screens, we proposed a novel method Vina-GPU to speedup AutoDock Vina with GPUs. Vina-GPU obtains a large-scale of parallelism on the Monte-Carlo based iterations and greatly reduces the search depth in each iteration. Besides, a heterogeneous OpenCL implementation of Vina-GPU was efficiently assigned by transforming the heterogeneous tree structure into a list structure whose nodes are visited in the traversed line. Vina-GPU can fully utilize abundant computational GPU cores to reach a large-scale of parallelization and acceleration. Also, Vina-GPU can realize the cross-platform operation on both CPUs and GPUs. Large benchmarks demonstrate that Vina-GPU achieves an average of 21 folds and a maximal speed-up of 50 folds on NVIDIA Geforce RTX 3090 over the original AutoDock Vina when keeping their comparable docking accuracy. To further enlarge its popularity of AutoDock Vina in large virtual screens, more efforts had been taken as the follows. A heuristic function was automatically fitted the most important hyperparameter ($search\_depth$) based on large testing experiments. Moreover, a graphical user interface (GUI) was designed for a convenient usage of Vina-GPU. In addition, an extension of Vina-GPU was provided on Windows, Linux and macOS, and also ensure its usage on personal computers, station servers and cloud computations etc. The source codes of Vina-GPU can be freely accessible at https://github.com/DeltaGroupNJUPT/Vina-GPU or http://www.noveldelta.com/Vina_GPU. In future studies, the following aspects would be taken into consideration for pushing the popularization of AutoDock Vina in large virtual screens. We will further analyze and mend the AutoDock Vina algorithm so that it can obtain a higher acceleration with GPUs. In addition, we will study other mainstream tools in the AutoDock Vina suites and accelerate them with GPUs. Besides, we will rewrite the AutoDock Vina algorithm to realize its acceleration on FPGA with higher price-performance ratio and more flexibility.

Acknowledgements

We acknowledge the support from Mr. Xianqiang Shi for providing the Huawei cloud service and Ms. Yemin Diao for offering us a work place.

Funding

This work was supported in part by the National Natural Science Foundation of China (61872198, 81771478 and 61971216); the Basic Research Program of Science and Technology Department of Jiangsu Province (BK20201378).
Conflict of Interest: none declared

References

Berman, H. M. *et al.* (2000). The protein data bank. *Nucleic acids research*, **28**(1), 235–242.

Bohacek, R. S. *et al.* (1996). The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews*, **16**(1), 3–50.

Cherkasov, A. *et al.* (2014). Qsar modeling: where have you been? where are you going to? *Journal of medicinal chemistry*, **57**(12), 4977–5010.

Ding, X. *et al.* (2020). Accelerated cdocker with gpus, parallel simulated annealing, and fast fourier transforms. *Journal of chemical theory and computation*, **16**(6), 3910–3919.

Eberhardt, J. *et al.* (2021). Autodock vina 1.2. 0: new docking methods, expanded force field, and python bindings.

Fan, M. *et al.* (2021). Gpu-accelerated flexible molecular docking. *The Journal of Physical Chemistry B*, **125**(4), 1049–1060.

Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons.

Golbraikh, A. *et al.* (2003). Rational selection of training and test sets for the development of validated qsar models. *Journal of computer-aided molecular design*, **17**(2), 241–253.

Goodsell, D. S. *et al.* (2021). The autodock suite at 30. *Protein Science*, **30**(1), 31–43.

Gorgulla, C. *et al.* (2020). An open-source drug discovery platform enables ultra-large virtual screens. *Nature*, **580**(7805), 663–668.

Handoko, S. D. *et al.* (2012). Quickvina: accelerating autodock vina using gradient-based heuristics for global optimization. *IEEE/ACM transactions on computational biology and bioinformatics*, **9**(5), 1266–1272.

Hartshorn, M. J. *et al.* (2007). Diverse, high-quality test set for the validation of protein- ligand docking performance. *Journal of medicinal chemistry*, **50**(4), 726–741.

Hwu, W.-M. W. (2011). *GPU computing gems emerald edition*. Morgan Kaufmann Publishers Inc.

Imbernón, B. *et al.* (2021). Metadock 2: a high-throughput parallel metaheuristic scheme for molecular docking. *Bioinformatics*, **37**(11), 1515–1520.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. 1. *New phytologist*, **11**(2), 37–50.

Jaghoori, M. M. *et al.* (2016). 1001 ways to run autodock vina for virtual screening. *Journal of computer-aided molecular design*, **30**(3), 237–249.

Kannan, S. and Ganji, R. (2010). Porting autodock to cuda. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

LeGrand, S. *et al.* (2020). Gpu-accelerated drug discovery with docking on the summit supercomputer: porting, optimization, and application to covid-19 research. In *Proceedings of the 11th ACM international conference on bioinformatics, computational biology and health informatics*, pages 1–10.

Lengauer, T. and Rarey, M. (1996). Computational methods for biomolecular docking. *Current opinion in structural biology*, **6**(3), 402–406.

Li, H. *et al.* (2012). idock: A multithreaded virtual screening tool for flexible ligand docking. In *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 77–84. IEEE.

Li, Y. *et al.* (2014). Comparative assessment of scoring functions on an updated benchmark: 2. evaluation methods and general results. *Journal of chemical information and modeling*, **54**(6), 1717–1736.

Meng, X.-Y. *et al.* (2011). Molecular docking: a powerful approach for structure-based drug discovery. *Current computer-aided drug design*, **7**(2), 146–157.

Mermelstein, D. J. *et al.* (2018). Fast and flexible gpu accelerated binding free energy calculations within the amber molecular dynamics package.

Morris, G. M. *et al.* (2009). Autodock4 and autodocktools4: Automated docking with selective receptor flexibility. *Journal of computational chemistry*, **30**(16), 2785–2791.

Ravindranath, P. A. *et al.* (2015). Autodockfr: advances in protein-ligand docking with explicitly specified binding site flexibility. *PLoS computational biology*, **11**(12), e1004586.

Santos-Martins, D. *et al.* (2019). D3r grand challenge 4: prospective pose prediction of bace1 ligands with autodock-gpu. *Journal of computer-aided molecular design*, **33**(12), 1071–1081.

Santos-Martins, D. *et al.* (2021). Accelerating autodock4 with gpus and gradient-based local search. *Journal of Chemical Theory and Computation*, **17**(2), 1060–1073.

Shin, J. H. *et al.* (2020). Gpu-accelerated autodock vina: Viking. `https://www.morressier.com/o/event/5e733c5acde2b641284a7e27/article/5e73656bcde2b641284aa4e5`.

Solis-Vasquez, L. *et al.* (2020). Parallelizing irregular computations for molecular docking. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*, pages 12–21. IEEE.

Stone, J. E. *et al.* (2016). Early experiences porting the namd and vmd molecular simulation and analysis software to gpu-accelerated openpower platforms. In *International conference on high performance computing*, pages 188–206. Springer.

Su, M. *et al.* (2018). Comparative assessment of scoring functions: the casf-2016 update. *Journal of chemical information and modeling*, **59**(2), 895–913.

Trott, O. and Olson, A. J. (2010). Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, **31**(2), 455–461.

Wang, Z. *et al.* (2016). Comprehensive evaluation of ten docking programs on a diverse set of protein–ligand complexes: the prediction accuracy of sampling power and scoring power. *Physical Chemistry Chemical Physics*, **18**(18), 12964–12975.

Wishart, D. S. *et al.* (2018). Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, **46**(D1), D1074–D1082.

Zhang, Y. and Sanner, M. F. (2019a). Autodock crankpep: combining folding and docking to predict protein–peptide complexes. *Bioinformatics*, **35**(24), 5121–5127.

Zhang, Y. and Sanner, M. F. (2019b). Docking flexible cyclic peptides with autodock crankpep. *Journal of chemical theory and computation*, **15**(10), 5161–5168.