Accelerating AutoDock Vina with GPUs

Shidi Tang,[†,‡] Ruiqi Chen,[¶] Mengru Lin,[¶] Qingde Lin,[§] Yanxiang Zhu,[¶] Ji Ding,[†,‡] Jiansheng Wu,[*,†,‡] Haifeng Hu,[‖] and Ming Ling[§]

†*School of Geographic and Biological Information, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
‡*Smart Health Big Data Analysis and Location Services Engineering Research Center of Jiangsu Province, Nanjing 210023, China*
¶*VeriMake Research, Nanjing Renmian Integrated Circuit Technology Co., Ltd., Nanjing 210088, China*
§*National ASIC System Engineering Technology Research Center, Southeast University, Nanjing 210096, China*
‖*School of Telecommunication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*

E-mail: jansen@njupt.edu.cn

Abstract

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 seriously limit the popularity of AutoDock Vina and the flexibility of usage in modern drug discovery. Thus, the design of a new method for accelerating AutoDock Vina with GPUs 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 lane, and then a heterogeneous OpenCL implementation was developed to realize its parallel acceleration leveraging thousands of GPU cores. Large benchmarks show that Vina-GPU reaches a maximum of 403-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. The Vina-GPU code and tool can be freely available at `http://www.noveldelta.com/Vina_GPU` for academic usage.

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 way for modern drug discovery.[1] It allows to predict molecular interactions where a protein and a ligand inducted to 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.[2–4]

Among all molecule docking tools, the AutoDock suite is the most popular, which consists various tools including AutoDock4,[5] AutoDock Vina,[6] AutoDock Vina 1.2.0,[7] AutoDock FR,[8] AutoDock Crank Pep,[9,10] AutoDock-GPU[11,12] etc. AutoDock Vina is usually recommended as the first-line tool in the implementation of molecular docking due to its docking speed and accuracy.[13] Moreover, it wins the best docking power in the last benchmark CASF-2016 for comparative assessment of scoring functions[14] and the best scoring power under the comparison with ten docking programs on a diverse protein–ligand complex sets.[15] AutoDock Vina uses a Monte-Carlo iterated local search method that 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, the position, orientation or torsion is randomly mutated with a disturbance followed by an affinity evaluation 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[16] 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 next iterations. As all we know, the Monte-Carlo based iterated local search method in AutoDock Vina is highly serial because the ongoing iteration depends on the previous outputs.

Previous 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}.[17] 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.[18–20] 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.[18] 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 almost 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)[21] and Open Computing Language (OpenCL).[22] GPU has been applied to accelerate molecular docking in several tools.[11,12,21,23–31] 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.[11] 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.[30] 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 recursively traversed . Thirdly, the CUDA architecture on NVIDIA GPU cards limits its cross-platform portability.

In this work, we propose 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 docking lanes and significantly reduces the search depth in each lane. Second, a heterogeneous OpenCL implementation is 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 a maximum of 403 times speed-up on NVIDIA Geforce RTX 3090 against the original AutoDock Vina on a quad-threaded CPU operation 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 enabled 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 for academic usage at `http://www.noveldelta.com/Vina_GPU`.

2 Methodology

The heterogeneous OpenCL architecture for 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 while 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). Further-

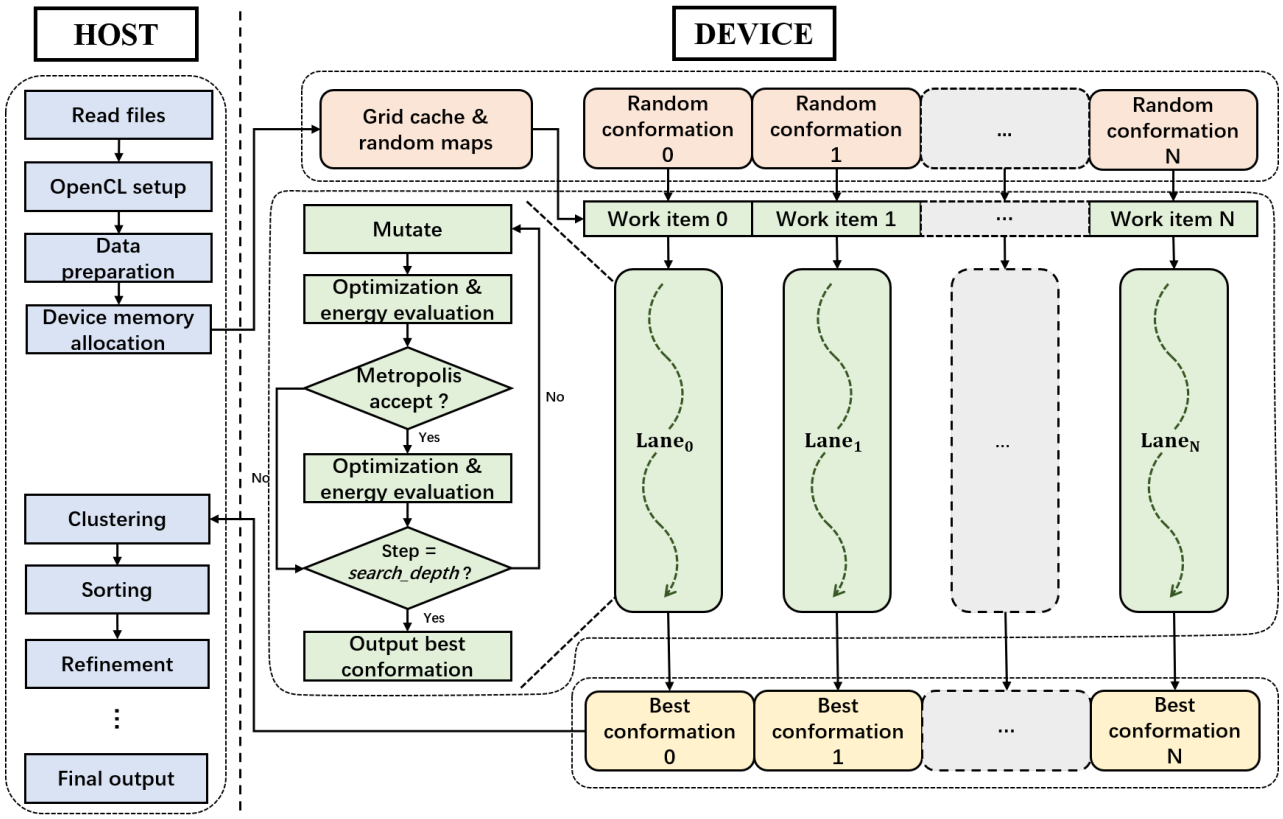


Figure 1: The OpenCL architecture for implementing Vina-GPU, which consists of a host (CPU) and a device (GPU) part of execution. The device part implements thousands of docking lanes, each of which is assigned with an OpenCL work item to perform a Monte-Carlo based local search method with largely reduced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

more, 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, for 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 is created to denote the relationship among these nodes. For example, the node 0 has two children-nodes (the node 1 and the node 4), so that the row 0 has two "T"s (indicating "True") in the 1^{st} and 4^{th} column (Figure 2). Thus, the recursive traverse of the heterogeneous tree can be converted into an iterative traverse of the node list and the children map which fits the OpenCL standard.

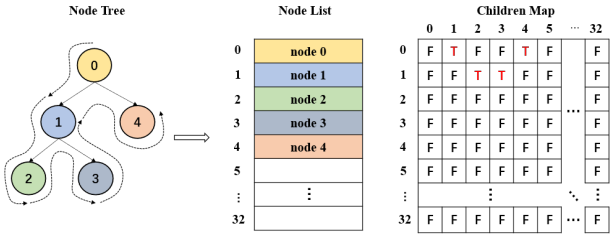


Figure 2: Transformation the node tree structure into the node list format. The heterogeneous tree is re-constructed in its traversed order(depth-first). An additional children map is built to reflect the relationship between nodes.

2.2 Device part

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

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 lane. Within each lane, an OpenCL work item is assigned to a randomly initialized conformation C , which can be represented by its position, orientation and torsion (POT):

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

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 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:

$$e = e_{inter} + e_{intra} \quad (2)$$

In which e_{inter} represents the interaction energy between the ligand and the receptor. It is calculated using trilinear interpolation that approximates the energy of each atom pair by looking up the grid cache. 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 of the complex, the scoring function can be denoted as a function SF of POT variables:

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

After the energy evaluation, a Broyden-Fletcher-Goldfarb-Shanno (BFGS)[16] optimization is applied to minimize the score function SF with its gradients of POT iteratively. Next, a metropolis acceptance criterion is adopted to decide whether to accept the optimized conformation or not with a comparison of the energy before the mutation e_0 and after the optimization e_{opt}, the accept probability P is represented by:

$$P = \begin{cases} 1 & e_0 > e_{opt} \\ \frac{\exp(e_0 - e_{opt})}{1.2} & e_0 \leq e_{opt} \end{cases} \quad (4)$$

It indicates that the accepted conformation is more likely to have a lower energy. Once accepted, the conformation will be further evaluated and optimized with BFGS for a second time. After that, the next iteration will be performed based on the optimized conformation from the previous iteration. After a largely-reduced searching iterations compared to the original Monte-Carlo based iterated local search method in AutoDock Vina, all the best conformations found by work items will be transferred back into the host part. The pseudocode of our proposed algorithm Vina-GPU is shown in Table 1.

3 Results and Discussion

3.1 Experimental Settings

All 140 complexes in the AutoDock-GPU study[12] are assigned as our experimental dataset, which is comprised of 85 complexes from the Astex Diversity Set,[32] 35 complexes from CASF-2013,[33] and 20 complexes from the Protein Data Bank.[34] 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 create 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 $size_x$, $size_y$, $size_z$). The details of our experimental data can be seen in Supplementary Table S4.

AutoDock Vina can be customized by several configurable arguments, including the center and the volume of searching spaces, the number of CPU cores to be utilized (*cpu*) and docking runs (*exhaustiveness*) etc. The most important two arguments (*cpu* and *exhaustiveness*) are set to 4 and 8, and the best docking result of AutoDock Vina are regarded as the golden line in all performance comparison. For Vina-GPU, an acceptable docking is defined by either of the following two criteria. One is the final docking score, which represents the binding affinity between a ligand and a receptor (the lower the score is better). If a difference of less than 1 kcal/mol occurs between the lowest docking score output by Vina-GPU and that by AutoDock Vina on the same complex, the performance of Vina-GPU is regarded as similar to that of AutoDock Vina, hence being an acceptable docking.[6] The other criterion is the root-mean-square deviation (RMSD) which represents the atom distance between each output structure and the ground truth X-ray structure (also the lower the better), and an acceptable docking was defined if the least RMSD among all output structures is smaller than 2 Å.[6]

AutoDock Vina is executed on Intel (R) Xeon

Table 1: Pseudocode of Vina-GPU

1. Generate N Random Conformations: $\{conf_{tmp_0}, conf_{tmp_1}, \ldots, conf_{tmp_N}\}$
2. ***for*** each $lane_i$ $(i = 0, \ldots, N)$ concurrently do:
3. ***for*** $search\_depth = 1, ..., r$:
4. $\quad conf_{cand_i} = \text{Mutation}(conf_{tmp_i})$
5. $\quad (conf_{cand_i}, energy_{cand_i}) = \text{BFGS Optimization \& Energy Evaluation}(conf_{cand_i})$
6. $\quad$ ***if***$(search\_depth == 0 \,||\, \text{Metropolis Accept}(energy_{cand_i}, energy_{tmp_i}))$
7. $\quad\quad conf_{tmp_i} = conf_{cand_i}$
8. $\quad\quad (conf_{tmp_i}, energy_{tmp_i}) = \text{BFGS Optimization \& Energy Evaluation}(conf_{tmp_i})$
9. $\quad$ ***end if***
10. ***end for***
11. ***end for***
12. Clustering & Sorting out the top 20 best conformations from N lanes

(R) Gold 6130 CPU @ 2.1GHz using Windows 10 Operating System with 32 GB memory. The two hyperparameters of AutoDock Vina (*cpu* and *exhaustiveness*) are set to 4 and 8, respectively. Extras including the box center ($center_x$, $center_y$, $center_z$) and the volume ($size_x$, $size_y$, $size_z$) are properly set in the config.txt file (also see Supplementary Table S1). Vina-GPU is developed with OpenCL v.3.0 and executed on three different GPUs (Nvidia Geforce GTX 1060, Nvidia Geforce RTX 2080Ti, Nvidia Geforce RTX 3090). Details are included in Supplementary Table S2. For Vina-GPU, more configurations are needed, including the number of work items (*thread*) and the size of searching iterations in each work item (*search_depth*). The hyperparameter "*thread*" is set to 2500, 5000, 7500 for Nvidia Geforce GTX 1060, Nvidia Geforce RTX 2080Ti and Nvidia Geforce RTX 3090 in order to take full use of their computational resource (from low to high).

3.2 Runtime Comparison

Our OpenCL implementation of Vina-GPU divides the whole computational process into the host and the device part (see Figure 1). Here, we evaluate the runtime acceleration from two aspects: (1) the device runtime Acc_d, and (2) the total (device + host) runtime Acc_{d+h}, which are defined by

$$Acc_d = \frac{t_{mc}}{t_d} \tag{5}$$

$$Acc_{d+h} = \frac{t_{all}}{t_{d+h}} \tag{6}$$

where t_{mc} and t_{all} are the Monte-Carlo iteration runtime and the total runtime of AutoDock Vina; t_d and t_{d+h} are the device runtime and the total runtime of Vina-GPU, respectively. We classified the 140 complexes into 3 subsets by their atom sizes N_{atom} (small: 5-23, medium: 24-36, large: 37-108), and gave out their runtime acceleration on different scales of complexities using violin plots (Figure 3). The average acceleration on each GPU card is highlighted by a white dot.

For Acc_d, Vina-GPU achieves acceleration with the maximum of 403X, as well as the averages of 54X, 49X and 57X on the 1060, 2080ti and 3090 GPU cards, respectively; The maximum 152X, and the averages of 13X, 13X, 17X acceleration on three GPUs are for Acc_{d+h} (Figure 3). In addition, there are three interesting observations in Figure 3. One is that Vina-GPU on NVIDIA Geforce GTX 1060 shows larger acceleration in some cases (Acc_d in the small subset and Acc_{d+h} in the small and medium subsets) than those on higher-end GPU cards. This is probably because that the GPU execution time only occupies a small part of the total runtime, and for higher-end GPUs, more time is needed to configure their running en-

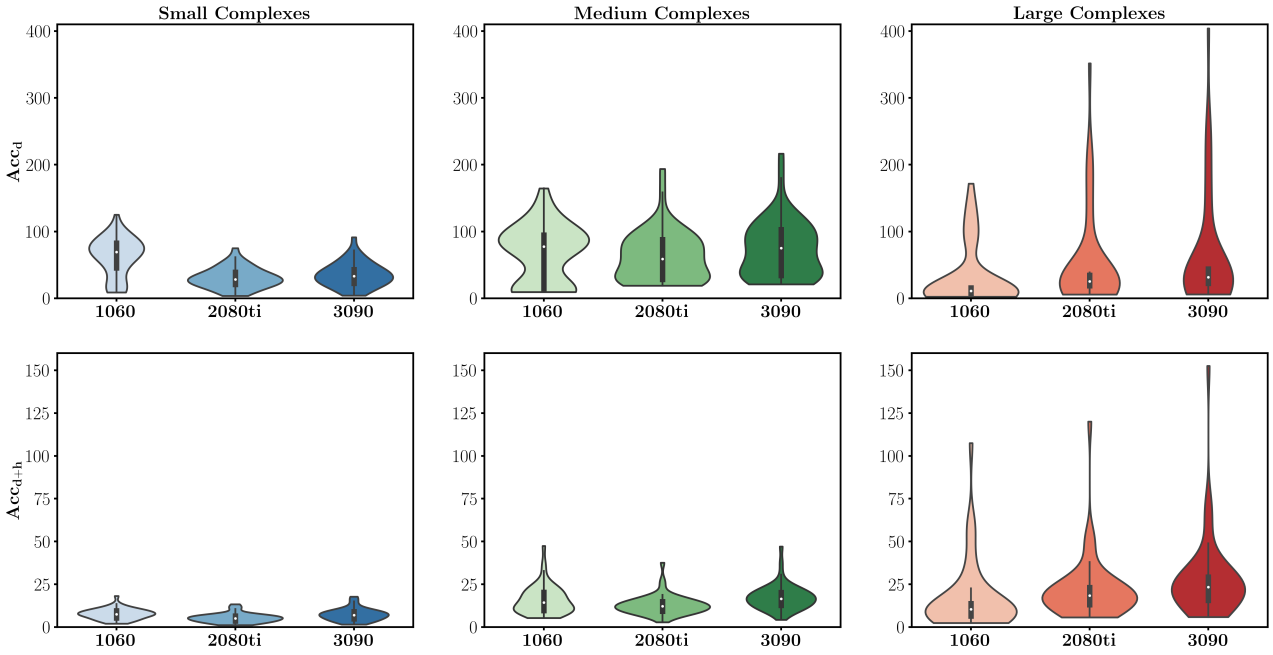


Figure 3: Runtime acceleration in terms of the device part (Acc_d with the maximum of 403X) and the "device + host" part (Acc_{d+h} with the maximum of 152X) on three different GPUs. To explore the acceleration on different scales of complexity, we divide our datasets into 3 subsets by their N_{atom} sizes (small: 5-23, medium: 24-36, large: 37-108). 1060: NVIDIA Geforce GTX 1060; 2080: Nvidia Geforce RTX 2080Ti; 3090: Nvidia Geforce RTX 3090.

vironment before starting the docking process, resulting in their relatively lower acceleration. Another is that the average acceleration on medium complexes are larger than other complexes in most cases. This is probably caused by that the largest decrease of searching iterations is obtained on the medium complexes by our Vina-GPU method. The last one is that the runtime acceleration on the device part (Acc_d) are much higher than the total-runtime acceleration Acc_{d+h}. This is due to the fact that the CPU runtime of one case on three different servers are almost the same, and the Amdahl's law[35] decides Vina-GPU to obtain a lower acceleration in terms of the total runtime.

To exhibit the detailed acceleration on different GPU cards, Figure 4 and 5 show the Acc_d and Acc_{d+h} performance of all 140 complexes with different N_{atom} and N_{rot}. Each bar represents a complex coupling with its corresponding acceleration. The acceleration of Acc_d and Acc_{d+h} varies from 2X to 403X and 1X to 152X, respectively. The maximal acceleration is achieved on the 3drf complex (PDBid) under Nvidia Geforce RTX 3090.

3.3 Docking Accuracy

We compare the docking accuracy in terms of score and RMSD of AutoDock Vina and our Vina-GPU on all 140 complexes (Figure 6). The red dashed line denotes the score or the RMSD criterion that defines an acceptable docking. As for the score performance in Figure 6, the score criterion divides these complexes into two parts. The first part represents that the docking score of Vina-GPU succeeds to meet the score criterion, while the second part means the failed docking by Vina-GPU. It can be seen that all docking scores succeed to meet the score criterion, hence Vina-GPU performed all 140 successful dockings. As for the RMSD performance in Figure 6, the RMSD criterion partitions these complexes into four parts. The first part means that both AutoDock Vina and Vina-GPU meet the RMSD criterion. The second part denotes that Vina-GPU succeeds to meet the RMSD criterion while AutoDock Vina fails, and vice versa in the third part. The last part indicates that neither AutoDock Vina nor Vina-GPU meet the RMSD criterion. There-

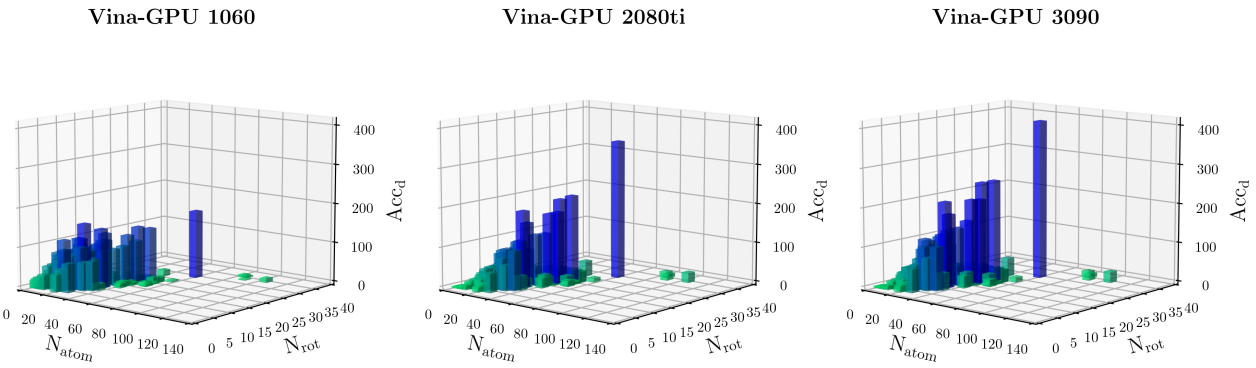


Figure 4: Runtime acceleration in terms of the device part (Acc_d) on 140 complexes with different N_{atom} and N_{rot}. The vertical axis ranges from 0 to 410. Each bar represents a complex coupling with its corresponding acceleration. Vina-GPU 1060, Vina-GPU 2080ti and Vina-GPU 3090 mean that Vina-GPU is executed on Nvidia Geforce GTX 1060, Nvidia Geforce RTX 2080Ti and Nvidia Geforce RTX 3090, respectively.

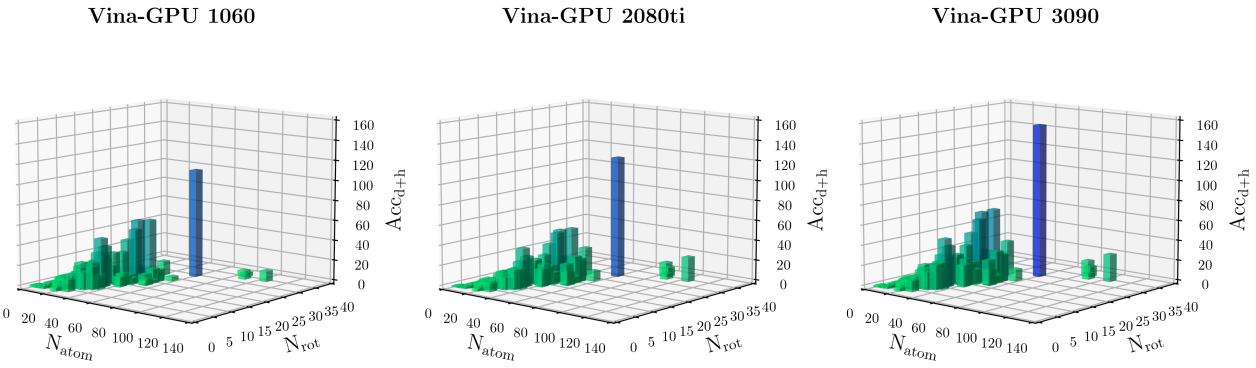


Figure 5: Runtime acceleration in terms of the "device + host" part (Acc_{d+h}) on 140 complexes with different N_{atom} and N_{rot}. The vertical axis ranges from 0 to 160. Each bar represents a complex coupling with its corresponding acceleration. Vina-GPU 1060, Vina-GPU 2080ti and Vina-GPU 3090 denote that Vina-GPU is implemented on Nvidia Geforce GTX 1060, Nvidia Geforce RTX 2080Ti and Nvidia Geforce RTX 3090, respectively.

fore, regardless from the score or the RMSD aspect, our Vina-GPU achieves a comparable docking accuracy in contrast with the original AutoDock Vina.

3.4 Influence of Search Depth

We analyzed the influence of the hyperparameter *search_depth* (from 1 to 100) on the docking accuracy of Vina-GPU, which is evaluated by docking score (kcal/mol) and RMSD (Å). Six typical complexes are selected to represent various levels of complexities (see Table 2), and their results are shown in Figure 7 and Figure 8. The score or RMSD criteria for defining an acceptable docking (see Section 3.1) are plotted with red dashed lines. The average performances on three GPU cards are indicated in blue solid lines. The score or RMSD value of the golden line (defined in Section 3.1) is plotted in a gold straight line.

Table 2: Properties of six complexes

PDBid	N_{atom}	N_{rot}
5tim	5	0
2bm2	33	7
1hvy	34	9
1os0	39	13
1jyq	60	20
3er5	108	31

With the increase of *search_depth*, the performance of Vina-GPU on docking score becomes better and better, where most of complexes can converge to the golden line (Figure 7). Meanwhile, we notice two special cases. One is that for the complex 5tim, its docking score keeps the same as the golden line with the increase of *search_depth*. This is probably because that 5tim is a small complex with its $N_{atom} = 5$ and $N_{rot} = 0$, and both Vina-GPU and AutoDock Vina can find the global minimum with a small size of *search_depth*, and its increase cannot bring any improvement on the score. Another is that, for the complex 3er5, the docking score reaches beyond the golden line when the size of *search_depth* is over 30. It means that the score performance of Vina-GPU is superior to that of AutoDock Vina. This is caused by that the Monte-Carlo method used in AutoDock Vina is non-deterministic, and for such a large complex 3er5 ($N_{atom} = 108$ and $N_{rot} = 31$), local minimums instead of the global one are obtained in the typical implementation.

As indicated in Figure 8, the RMSD of Vina-GPU in most cases keeps below the RMSD criterion with the increase of *search_depth*, except for two complexes 5tim and 3er5. For the small complex 5tim, the exception is probably caused by two reasons. One is the inconsistency between the change of docking score and that of RMSD. The docking results of AutoDock Vina on the complex 1hvy (see Supplementary table S3) verifies our guess. Another one is that in the host part of Vina-GPU (see Figure 1), the best conformations are sorted by their docking scores. Such scheme forces Vina-GPU to pick up those conformations with lower docking scores. However, the chosen conformations occasionally have higher RMSDs, resulting in a worse RMSD performance on the complex 5tim. As for the large complex 3er5, this is because that Vina-GPU utilizes the same scoring function as that of AutoDock Vina, and hence both of them cannot meet the RMSD criterion on the complex 3er5.

3.5 Fitting the Search Depth

The hyperparameter *search_depth* is of the most importance, and its value is sensitive to the docking performance of Vina-GPU. For a convenient usage of Vina-GPU, a heuristic formula is fit to automatically determine the proper size of *search_depth* for a given complex. A large number of test experiments are executed on all 140 complexes to examine their docking performances and runtimes under different sizes of *search_depth* using the same scheme in Section 3.4. The best size of *search_depth* for each complex is arranged by the smallest value in those meeting the score criterion. The best sizes of *search_depth* on the six complexes are shown in Table 3, and those of all 140 complexes can be seen in the supplementary Table S4.

Then, the least squares method is used to fit

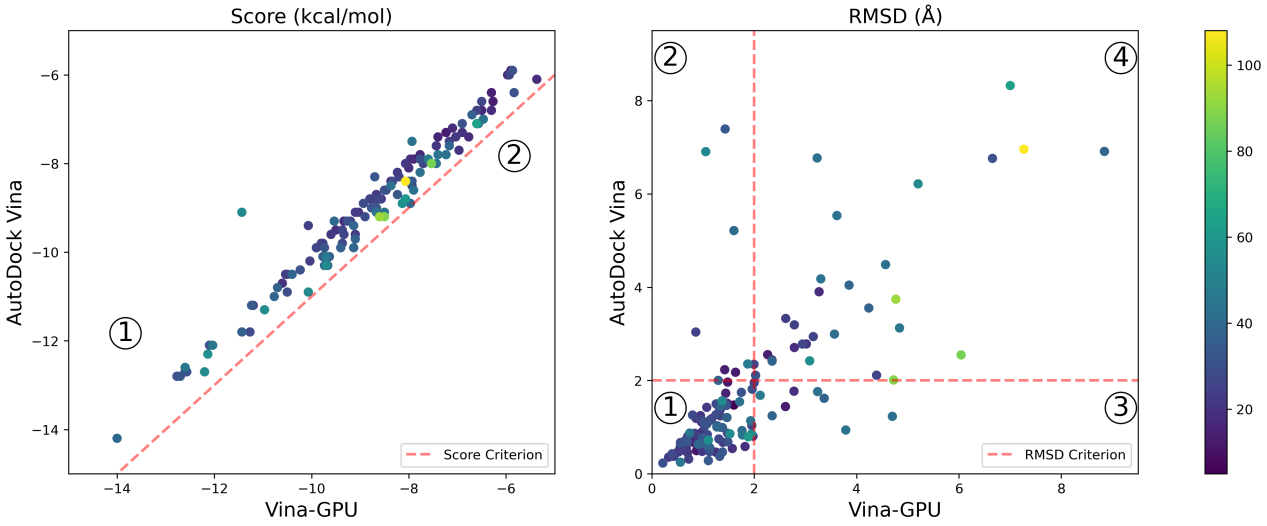


Figure 6: Comparison of docking accuracy between AutoDock Vina and our Vina-GPU on 140 complexes. The color bar encodes the number of atoms of one ligand. The score and RMSD criteria that define an acceptable binding pose are represented by red dashed lines. As for the score performance, the score criterion divides these complexes into two parts, where the first one means that the docking score of Vina-GPU succeeds to meet the score criterion, and the second one denotes the unsuccessful docking score by Vina-GPU. As for the RMSD performance, the RMSD criterion divides these complexes into four parts. The first part means that the docking results of Vina-GPU and AutoDock Vina both succeed to meet the RMSD criterion. The second part denotes that Vina-GPU meets the docking RMSD criterion while AutoDock Vina fails, and vice versa in the third part. The last part represents that neither AutoDock Vina nor Vina-GPU meet the RMSD criterion.

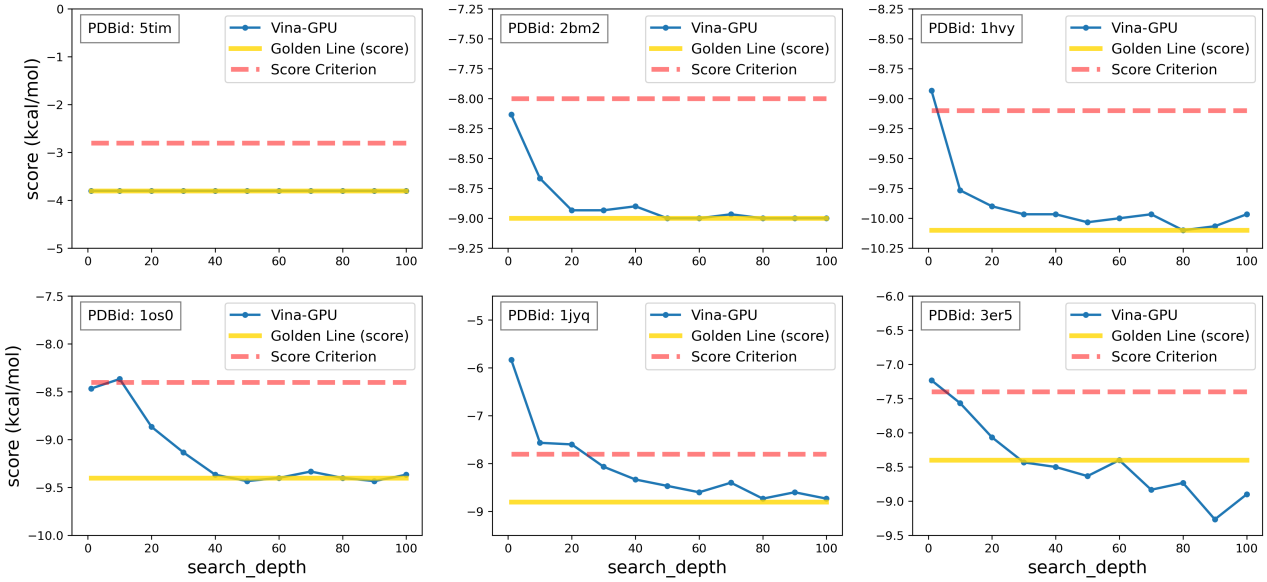


Figure 7: Docking scores of Vina-GPU on six complexes with various levels of complexity. The straight lines (gold) denote the score value of the best conformation returned by AutoDock Vina with the $cpu = 4$ and $exhaustiveness = 8$. The dashed lines (red) denote the upper bounds of the docking score (score criterion) for an acceptable docking. The solid lines (blue) denote the average scores of Vina-GPU on three GPU cards. The horizontal axis indicates the number of searching iterations in each docking lane of Vina-GPU.

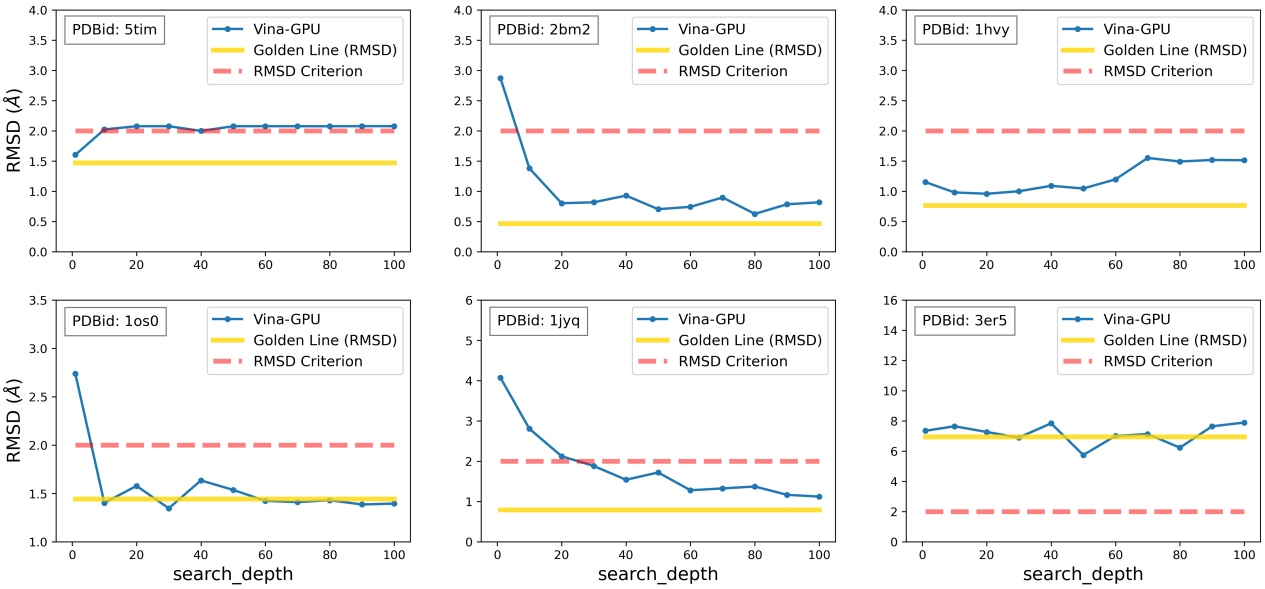


Figure 8: Docking RMSDs of Vina-GPU on six complexes with various levels of complexity. The straight lines (gold) denote the RMSD value of the best conformation returned by AutoDock Vina with the $cpu = 4$ and $exhaustiveness = 8$. The dashed lines (red) denote the upper bounds of the RMSD value (RMSD criterion) for an acceptable docking. The solid lines (blue) denote the average RMSD of Vina-GPU on three GPU cards. The horizontal axis indicates the number of searching iterations in each docking lane of Vina-GPU.

Table 3: Best sizes of *search_depth* on the six complexes

PDBid	Best size of *search_depth*
5tim	1
2bm2	1
1hvy	10
1os0	20
1jyq	30
3er5	10

an empirical formula of the best *search_depth* with respect to their N_{atom} and N_{rot} of all 140 complexes. The formula can automatically determine the proper size of *search_depth* of a given ligand as follows,

$$search\_depth = floor(0.24 \times N_{atom} + 0.29 \times N_{rot} - 5.74) \quad (7)$$

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

3.6 Conformation Spaces Analysis

To verify their equivalence in molecular docking, we intend to analyze the full conformation spaces explored by AutoDock Vina or our Vina-GPU. Firstly, we discussed the searching strategy of Vina-GPU and explained the reason why Vina-GPU can achieve a great acceleration on the premise of an acceptable docking accuracy. Then, we visualized and made a comparison of their whole searching spaces of conformations.

Vina-GPU enables thousands of docking lanes to calculate concurrently. In each lane, there is a randomly initiated conformation. These docking lanes will divide the whole search space into thousands of subspaces, in each of which an initial conformation is optimized. We define the solution space that covers all possible conformations as a high-dimensional space $\boldsymbol{S} = \{C_0, C_1, C_2, \dots\}$. By dividing $\boldsymbol{S}$ into n sub-spaces, we have

$$\boldsymbol{S} = \{\boldsymbol{S}_{sub_0}, \boldsymbol{S}_{sub_0}, ..., \boldsymbol{S}_{sub_n}\} \quad (8)$$

Where each initial conformation belongs to a

sub-space

$$C_i \in \boldsymbol{S}_{sub_i} \tag{9}$$

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

To verify their equivalence, Figure 9 shows a comparison of solution space $\boldsymbol{S}$ using AutoDock Vina and Vina-GPU on a representative complex (PDBid: 2bm2, $N_{atom} = 33$, $N_{rot} = 7$). AutoDock Vina is executed with the configuration of "$cpu = 1$, $exhaustiveness = 1$ and searching iterations = 22365". Vina-GPU is executed under various strategies, where different scales of docking lane and *search_depth* size in each lane are used. The whole searching spaces reached by these strategies ($lanes \times$ *search_depth*) are almost the same as that by the original AutoDock Vina. All conformations searched by AutoDock Vina or Vina-GPU during iterations are indicated as orange or blue dots, respectively. Each conformation is represented by its POT and shown in cartesian coordinates, in which a principal component analysis (PCA) method is used to reduce the dimensionality of orientation and torsion into three. The best conformation is shown in red star (indicated by an arrow).

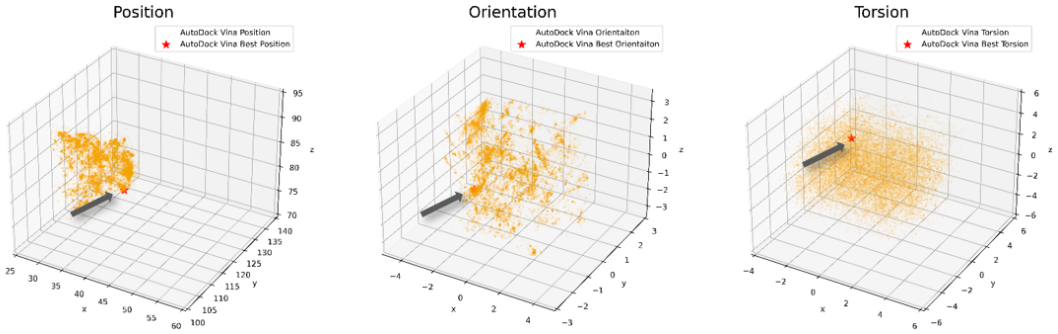
As shown in Figure 9a and Figure 9b, the whole conformation space reached by Vina-GPU or AutoDock Vina are almost the same in their position, orientation or torsion. With the increase of Vina-GPU on its parallelism scale and the reduce of its search depth in each lane, their observations keep almost unchanged (Figure 9c and Figure 9d). Moreover, the best conformations found by Vina-GPU are close to those by AutoDock Vina. It demonstrates that our Vina-GPU has a similar possibility to find excellent conformations with AutoDock Vina through expanding the scale of parallelism and reducing the size of searching iterations in each lane.

3.7 Docking the DrugBank Datasets

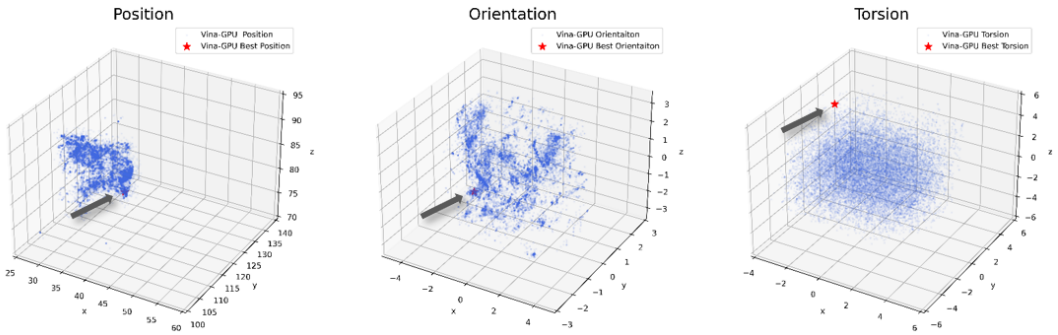
To show the acceleration effect of our Vina-GPU in implementing real virtual screens of large compound databases, an example is detailed on the receptor 3drf (PDBid) with the docking of DrugBank.[36] The total of 9137 molecules on the DrugBank database were downloaded from `https://go.drugbank.com/releases/latest#structures`. Only about 9.52 hours are taken to execute the whole docking process by Vina-GPU while $\sim$ 66.28 hours by AutoDock Vina. In addition, Figure 10 visualized the binding poses of 3 molecules (drugbank4200, drugbank7520, drugbank8545) with the best docking scores (-11.6, -11.5, -11.5) using Pymol.[37]

3.8 Usage of Vina-GPU

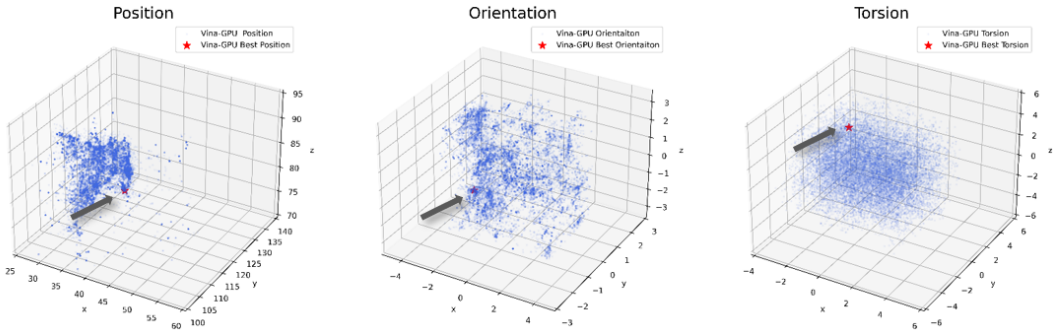
We developed a user-friendly graphic user interface (GUI) instead of the original terminal form. Our GUI can be utilized without installation. It consists of five main components (see supplementary Figure S1). The first one is the *input files* that defines both the ligand and the receptor, these files can be easily chosen by a *select file* button. The second one is the *output files* that determine the final output file of the ligand. The third one is the *docking box* that includes box center and box size. These parameters determine the position and the size of the docking pocket, others including *thread* and *search_depth* can be defined in *optional parameters* of the fourth component. The last part is the *prompt* that outputs the information during the runtime. Finally, Vina-GPU can be executed by the *start* button. In addition, we provide 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. All source codes and tools of Vina-GPU can be freely available at `http://www.noveldelta.com/Vina_GPU` for academic usage.



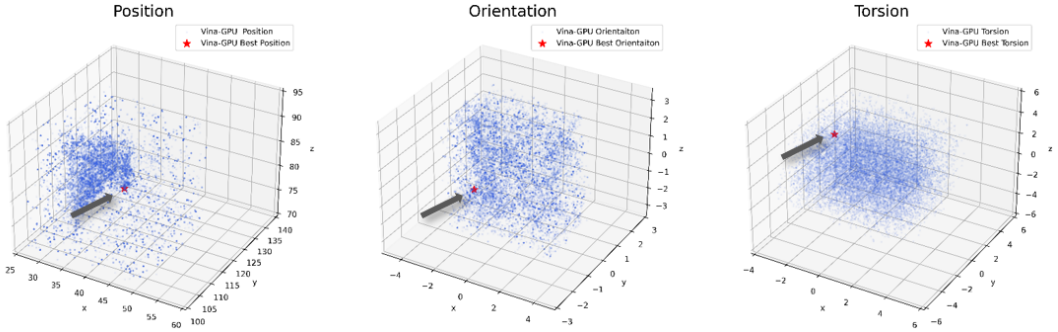
(a) AutoDock Vina: 1 initial conformation with 22365 searching iterations



(b) Vina-GPU: 10 docking lanes with *search_depth* set to 2237

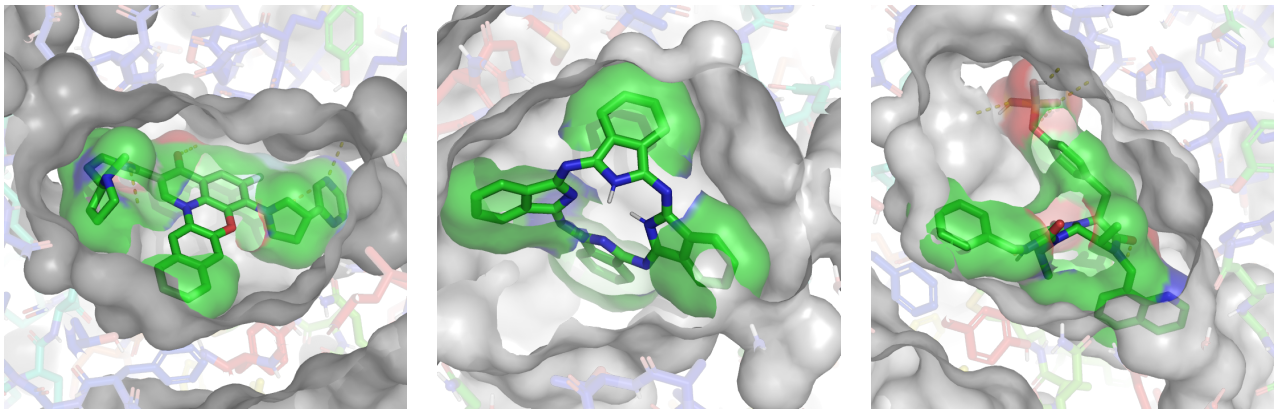


(c) Vina-GPU: 100 docking lanes with *search_depth* set to 224



(d) Vina-GPU: 1000 docking lanes with *search_depth* set to 22

Figure 9: Conformation spaces (of PDBid: 2bm2) explored by AutoDock Vina or Vina-GPU during iterations. AutoDock Vina ($cpu = 1, exhaustiveness = 1$) is executed with one initial conformation and 22365 searching iterations by default. Vina-GPU is executed under different scales of lanes with different *search_depth* in each lane. The overall searching iterations ($lanes \times search\_depth$) are kept almost the same. Each conformation is represented by its position, orientation and torsion (POT) and is plotted with bule or orange dots. The principal component analysis (PCA) method is used to reduce the dimensionality of orientation and torsion into three. The best conformations are highlighted with red stars (pointed by arrows).



(a) drugbank4200 (b) drugbank7530 (c) drugbank8545

Figure 10: Binding poses of conformations with the lowest docking scores on a receptor (PDBid: 3drf). The three small molecules are drugbank4200, drugbank7530 and drugbank8545, and obtain the docking scores of -11.6, -11.5, -11.5, respectively.

4 Conclusion

In modern drug discovery, huge resource investment and high entry threshold seriously weaken the popularity of AutoDock Vina in virtual screening from large compound databases. To advance the wide spread of AutoDock Vina in large virtual screens, we propose a novel method Vina-GPU to speedup AutoDock Vina with GPUs. Vina-GPU obtains a large-scale of parallelism on the Monte-Carlo based docking lanes and greatly reduces the search steps in each lane. Besides, a heterogeneous OpenCL implementation of Vina-GPU is 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 a maximal speed-up of 403 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 have been taken as the follows. A heuristic function is automatically fitted the most important hyperparameter ($search\_depth$) after 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 are freely accessible at `http://www.noveldelta.com/Vina_GPU` for academic usage.

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 analyze 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.

Data and Software Availability

The authors declare no competing financial interest. All the source codes, the documentation and the updates related to Vina-GPU have been uploaded to our website `http://www.noveldelta.com/Vina_GPU` (in the user guideline section) under an Apache-2.0 license. The 140 complexes described in Section 3.1

are available at `https://zenodo.org/record/4031961#.Yags3NBByUk` and the 9137 DrugBank molecules used in Section 3.7 are available at `https://go.drugbank.com/releases/latest#structures`. If there is any problem in reproducing our work, please feel free to contact us.

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

References

(1) Meng, X.-Y.; Zhang, H.-X.; Mezei, M.; Cui, M. Molecular docking: a powerful approach for structure-based drug discovery. *Current computer-aided drug design* **2011**, *7*, 146–157.

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

(3) Cherkasov, A.; Muratov, E. N.; Fourches, D.; Varnek, A.; Baskin, I. I.; Cronin, M.; Dearden, J.; Gramatica, P.; Martin, Y. C.; Todeschini, R., et al. QSAR modeling: where have you been? Where are you going to? *Journal of medicinal chemistry* **2014**, *57*, 4977–5010.

(4) Golbraikh, A.; Shen, M.; Xiao, Z.; Xiao, Y.-D.; Lee, K.-H.; Tropsha, A. Rational selection of training and test sets for the development of validated QSAR models. *Journal of computer-aided molecular design* **2003**, *17*, 241–253.

(5) Morris, G. M.; Huey, R.; Lindstrom, W.; Sanner, M. F.; Belew, R. K.; Goodsell, D. S.; Olson, A. J. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *Journal of computational chemistry* **2009**, *30*, 2785–2791.

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

(7) Eberhardt, J.; Santos-Martins, D.; Tillack, A.; Forli, S. AutoDock Vina 1.2. 0: new docking methods, expanded force field, and Python bindings. **2021**,

(8) Ravindranath, P. A.; Forli, S.; Goodsell, D. S.; Olson, A. J.; Sanner, M. F. AutoDockFR: advances in protein-ligand docking with explicitly specified binding site flexibility. *PLoS computational biology* **2015**, *11*, e1004586.

(9) Zhang, Y.; Sanner, M. F. AutoDock CrankPep: combining folding and docking to predict protein–peptide complexes. *Bioinformatics* **2019**, *35*, 5121–5127.

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

(11) Santos-Martins, D.; Eberhardt, J.; Bianco, G.; Solis-Vasquez, L.; Ambrosio, F. A.; Koch, A.; Forli, S. D3R Grand Challenge 4: prospective pose prediction of BACE1 ligands with AutoDock-GPU. *Journal of computer-aided molecular design* **2019**, *33*, 1071–1081.

(12) Santos-Martins, D.; Solis-Vasquez, L.; Tillack, A. F.; Sanner, M. F.; Koch, A.; Forli, S. Accelerating AutoDock4 with GPUs and gradient-based local search. *Journal of Chemical Theory and Computation* **2021**, *17*, 1060–1073.

(13) Goodsell, D. S.; Sanner, M. F.; Olson, A. J.; Forli, S. The AutoDock suite at 30. *Protein Science* **2021**, *30*, 31–43.

(14) Su, M.; Yang, Q.; Du, Y.; Feng, G.; Liu, Z.; Li, Y.; Wang, R. Comparative assessment of scoring functions: the CASF-2016 update. *Journal of chemical information and modeling* **2018**, *59*, 895–913.

(15) Wang, Z.; Sun, H.; Yao, X.; Li, D.; Xu, L.; Li, Y.; Tian, S.; Hou, T. 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* **2016**, *18*, 12964–12975.

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

(17) Bohacek, R. S.; McMartin, C.; Guida, W. C. The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews* **1996**, *16*, 3–50.

(18) Gorgulla, C.; Boeszoermenyi, A.; Wang, Z.-F.; Fischer, P. D.; Coote, P. W.; Das, K. M. P.; Malets, Y. S.; Radchenko, D. S.; Moroz, Y. S.; Scott, D. A., et al. An open-source drug discovery platform enables ultra-large virtual screens. *Nature* **2020**, *580*, 663–668.

(19) Li, H.; Leung, K.-S.; Wong, M.-H. idock: A multithreaded virtual screening tool for flexible ligand docking. 2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). 2012; pp 77–84.

(20) Jaghoori, M. M.; Bleijlevens, B.; Olabarriaga, S. D. 1001 Ways to run AutoDock Vina for virtual screening. *Journal of computer-aided molecular design* **2016**, *30*, 237–249.

(21) Kannan, S.; Ganji, R. Porting autodock to CUDA. IEEE Congress on Evolutionary Computation. 2010; pp 1–8.

(22) Munshi, A.; Gaster, B.; Mattson, T. G.; Ginsburg, D. *OpenCL programming guide*; Pearson Education, 2011.

(23) Mermelstein, D. J.; Lin, C.; Nelson, G.; Kretsch, R.; McCammon, J. A.; Walker, R. C. Fast and flexible gpu accelerated binding free energy calculations within the amber molecular dynamics package. 2018.

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

(25) Stone, J. E.; Hynninen, A.-P.; Phillips, J. C.; Schulten, K. Early experiences porting the NAMD and VMD molecular simulation and analysis software to GPU-accelerated OpenPOWER platforms. International conference on high performance computing. 2016; pp 188–206.

(26) LeGrand, S.; Scheinberg, A.; Tillack, A. F.; Thavappiragasam, M.; Vermaas, J. V.; Agarwal, R.; Larkin, J.; Poole, D.; Santos-Martins, D.; Solis-Vasquez, L., et al. GPU-accelerated drug discovery with docking on the summit supercomputer: porting, optimization, and application to COVID-19 research. Proceedings of the 11th ACM international conference on bioinformatics, computational biology and health informatics. 2020; pp 1–10.

(27) Fan, M.; Wang, J.; Jiang, H.; Feng, Y.; Mahdavi, M.; Madduri, K.; Kandemir, M. T.; Dokholyan, N. V. Gpu-accelerated flexible molecular docking. *The Journal of Physical Chemistry B* **2021**, *125*, 1049–1060.

(28) Ding, X.; Wu, Y.; Wang, Y.; Vilseck, J. Z.; Brooks III, C. L. Accelerated CDOCKER with GPUs, parallel simulated annealing, and fast Fourier transforms. *Journal of chemical theory and computation* **2020**, *16*, 3910–3919.

(29) Imbernón, B.; Serrano, A.; Bueno-Crespo, A.; Abellán, J. L.; Pérez-Sánchez, H.; Cecilia, J. M. METADOCK

2: a high-throughput parallel metaheuristic scheme for molecular docking. *Bioinformatics* **2021**, *37*, 1515–1520.

(30) Shin, J. H.; Kim, J.; Chae, J.; Yun, S. J. GPU-accelerated autodock vina: Viking. 2020.

(31) Solis-Vasquez, L.; Santos-Martins, D.; Tillack, A. F.; Koch, A.; Eberhardt, J.; Forli, S. Parallelizing Irregular Computations for Molecular Docking. 2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3). 2020; pp 12–21.

(32) Hartshorn, M. J.; Verdonk, M. L.; Chessari, G.; Brewerton, S. C.; Mooij, W. T.; Mortenson, P. N.; Murray, C. W. Diverse, high-quality test set for the validation of protein- ligand docking performance. *Journal of medicinal chemistry* **2007**, *50*, 726–741.

(33) Li, Y.; Han, L.; Liu, Z.; Wang, R. Comparative assessment of scoring functions on an updated benchmark: 2. Evaluation methods and general results. *Journal of chemical information and modeling* **2014**, *54*, 1717–1736.

(34) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The protein data bank. *Nucleic acids research* **2000**, *28*, 235–242.

(35) Hill, M. D.; Marty, M. R. Amdahl's law in the multicore era. *Computer* **2008**, *41*, 33–38.

(36) Wishart, D. S.; Feunang, Y. D.; Guo, A. C.; Lo, E. J.; Marcu, A.; Grant, J. R.; Sajed, T.; Johnson, D.; Li, C.; Sayeeda, Z., et al. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic acids research* **2018**, *46*, D1074–D1082.

(37) DeLano, W. L., et al. Pymol: An open-source molecular graphics tool. *CCP4 Newsletter on protein crystallography* **2002**, *40*, 82–92.

TOC Graphic