

A GPU Accelerated Fragment-based De Novo Ligand Design by a Bayesian Optimization Algorithm

MOHAMED WAHIB^{1,a)} ASIM MUNAWAR¹ MASAHARU MUNETOMO² KIYOSHI AKAMA²

Received: November 18, 2011, Accepted: December 10, 2011, Released: March 26, 2012

Abstract: De Novo ligand design is an automatic fragment-based design of molecules within a protein binding site of a known structure. A Bayesian Optimization Algorithm (BOA), a meta-heuristic algorithm, is introduced to join predocked fragments with a user-supplied list of fragments. A novel feature proposed is the simultaneous optimization of force field energy and a term enforcing 3D-overlap to known binding mode(s). The performance of the algorithm is tested on Liver X receptors (LXRs) using a library of about 14,000 fragments and the binding mode of a known heterocyclic phenyl acetic acid to bias the design. We further introduce the use of GPU (Graphics Processing Unit) to overcome the excessive time required in evaluating each possible fragment combination. We show how the GPU utilization enables experimenting larger fragment sets and target receptors for more complex instances. The results show how the nVidia's Tesla C2050 GPU was utilized to enable the generation of complex agonists effectively. In fact, eight of the 1,809 molecules designed for LXRs are found in the ZINC database of commercially available compounds.

Keywords: parallel evolutionary algorithms, drug design, Graphics Processing Unit

1. Introduction

The search for drug molecules with computational methods is often performed by high-throughput docking or to a lesser extent by De Novo drug design approaches. While virtual screening relies on pre-existing compounds, De Novo design approaches generate novel molecules out of building blocks consisting of single atoms or fragments. Because of the difficulty to predict synthetic accessibility, De Novo drug design tools often generate molecules that are demanding to synthesize. Approaches that are commonly employed to improve synthetic accessibility include the use of connection rules to join building blocks [9], [17], [31] or the build-up of molecules from fragments obtained by prior decomposition of existing compounds [5]. Connection rules are either derived from organic synthesis reactions [31], [33] or they are based on the knowledge of the occurrence of certain bonds in existing molecules [29].

Due to the huge and non-linear search spaces (Typically, tens of thousands of orientations are generated for each ligand candidate), global optimization algorithms are usually employed to search the chemical space by generating new molecular structures through probing many different fragments in a combinatorial fashion. Traditionally, related projects have embraced Evolutionary Algorithms (a class of global optimization algorithms inspired from the biological phenomenon of evolution) for this problem as will be shown below. In this research, the choice was to use Bayesian Optimization Algorithm (BOA) [28], an EA that

proved to give very good results in complex global optimization problems. Several projects Utilized evolutionary algorithms in general to apply virtual screening. Belda et al. [2] used BOA among other EDAs to tackle the docking problem, however the authors reported relatively poor performance for BOA as BOA needs a large population to construct a well-representing BN. This was the motive to utilize GPUs and thus being capable of using large population sizes to overcome this limitation in BOA. Lameijer et al. [18] conducted a comprehensive study for EAs in drug design and virtual screening. The most notable of these projects include [6] which uses a simple genetic algorithm and rely on a local search method to achieve better compounds. Yet, the primal focus was not on parallelizing the code to achieve the best performance in terms of speed. Wang et al. [29] defined a generic system for virtual screening depending on GAs in the back end. However, the environment required user intervention to direct the search process. Ghosh et al. [10] intensified the effect of GAs by relying mainly on local search techniques and considering the problem as a multiobjective optimization problem.

Here, as a main contribution, a novel approach for De Novo Design of agonists is presented. The algorithm utilizes a fragment-based method that generates molecules by joining predocked fragments with linkers. A parallel version of BOA is used to search for feasible solutions. Only the predocked fragments are encoded by the BOA, while suitable linker fragments are efficiently evaluated with a tabu search [11] using look-up tables. The fitness function used is a novel combination of force field energy and a measure of the 3D-overlap to known binding mode(s). The energy term consists of intra- and intermolecular contributions. The measure of 3D-overlap enforces a spatial distribution of the atoms of the designed molecule similar to the one in the known binding mode of the agonist(s) without explicitly consid-

¹ Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Hokkaido 060-0811, Japan

² Information Initiative Center, Hokkaido University, Sapporo, Hokkaido 060-0811, Japan

^{a)} wahib@ist.hokudai.ac.jp

ering the covalent structure. The algorithm is evaluated on liver X receptors (LXRs) [4], presenting a complex optimization problem due to the large number of fragments used. Different fitness function setups are analyzed for their search efficiency. Notably, the algorithm is able to suggest molecules with new scaffolds or substituents that, at the same time, preserve the main binding interaction motifs of known agonists of LXRs.

For complex structures with high order of used fragments (such as LXRs in this case), the massive computation cost expected makes parallel computing a De Facto issue. Therefore, the system proposed utilizes nVidia GPU in order to harness the high computing power of state-of-art GPUs. A major hurdle with utilizing GPU is the complexity of the GPU architecture and the need to carefully optimize and tune any application running over GPU to achieve a highly efficient performance. A second main contribution in this paper is the design of the De Novo drug design algorithm to run efficiently over the SIMT architecture of GPU. The design includes performance optimization strategies we introduced earlier in Ref. [25].

The rest of paper is as follows. The following section overviews the algorithm and implementation over GPU while Section 3 shows the results and discussion. Finally Section 4 concludes.

2. Algorithm and Implementation over GPU

2.1 BOA

BOA belongs to a class of algorithms known as Estimation of Distribution Algorithms (EDAs) [23]. EDAs are an outgrowth of Genetic Algorithms (GAs). Conventional GAs maintain a population of probable solutions and then they apply genetic operators like selection, mutation, and crossover to find the next population. This process continues until the algorithm finds an acceptable solution. EDAs replace the variation done by the crossover step in conventional GAs with some kind of statistical inference from the existing population to be used to construct the next population. In the case of BOA, variation starts by constructing a Bayesian Network (BN) [12] as a model of promising solutions after selection. New candidate solutions are then generated by sampling the constructed Bayesian network. Finally, new solutions are incorporated into the population, eliminating some old candidate solutions, and the next iteration is executed unless a termination criterion is met. **Figure 1** shows the important steps in a single iteration of BOA. BOA uses BN to encode the structure of a problem. In the chromosome of length n each gene is treated as a variable and is represented by a node in the dependency graph. For each variable X_i it defines a set of variables P_{X_i} it depends on, so the distribution of individuals is encoded as:

$$p(X) = \prod_{i=0}^{n-1} p(X_i | P_{X_i})$$

A directed edge from X_j to X_i in the network implies that X_j belongs to the parent nodes P_{X_i} of X_i . To reduce the space of networks, the number of incoming edges into each node is limited to k . Bayesian Dirichlet (BD) metric [13] is used to measure the quality of the network. A special case of BD metric called K2 metric is used when no prior information about the problem

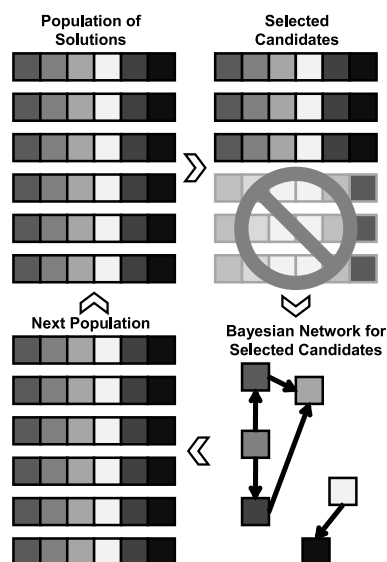


Fig. 1 Bayesian Optimization Algorithm (BOA).

is available. The search for an optimal BN is an NP hard problem, so some kind of search algorithm is necessary for reaching an optimal result in a reasonable amount of time. Different algorithms can be used for building BN. A simple greedy algorithm with only one edge addition in each step is commonly used for the purpose [27], [28]. The algorithm starts with an empty network B and for each edge that can be added (new edge is added only if the edge meets the limit of the incoming edges and does not introduce a cycle in the graph), it computes the BD metrics for the BN. The edge giving the highest improvement is then added to the network B . This process is repeated until no more addition is possible. After this step the variable (nodes) are ordered in a topological order and then the nodes whose parents are already determined are generated using the conditional probabilities. This is repeated until all the variables are generated.

BOA in combinatorial drug design are relevant to the nature of De Novo fragment-based drug design concept. EAs in general are perfect candidates for this kind of problem. EAs are perfect candidates for applications where deterministic or analytic methods fail. For instance, problems where the underlying mathematical model for the search space is not well defined. The optimization algorithm runs in this case in a blackbox fashion. Belda et al. [3] did a thorough study on the use of EAs in De Novo peptide design. After analyzing the problem structure, the authors claimed the superiority of EAs for searching chemical spaces to synthesize peptides. They also confirmed the achievement of high quality output with a range of different EAs. Moreover, the authors reported EDAs specifically to have high potential for such class of problems by evaluating the BN inference systems.

Nevertheless, the authors failed to fully utilize EDAs for such problems due to the large populations required to build a highly representing inference network. Accordingly, a high performance parallel implementation for fragment-based design with EDAs is essential to achieve high quality structures. This paper capitalizes on the robustness of GPUs and strong potential of BOA (as shown by Belda et al. [3]) to construct an end-to-end system for De Novo fragment-based drug design. In addition, as shown be-

low in the results section, the overlapping volume between the target structure and the generated molecules implies the need for a larger the number of individuals which can exchange genetic material. This would partially explain the high solution quality reported by Belda et al. where in this problem the sample size seem to have a high threshold before reaching the phase of model overfitting.

2.2 Fragment-based De Novo Ligand Design

To develop new medicines, the drug industry needs to find promising chemical drug structures called ligands that will match the target receptor protein of an organ in a human body. The conventional approach in drug design called *screening by docking* checks matching between the molecule of the target organ and ligand chemicals taken from their database. The degree of matching is calculated by minimizing energy potential between the molecule and the ligand. De Novo ligand design uses a fragment-based method to generate molecules by joining pre-docked fragments with linkers. **Figure 2** shows how De Novo ligand design can generate new structures by adding fragments. To connect pre-docked fragments with linker fragments we use a combination of two stochastic search procedures, BOA and a tabu search. Heavy atom-hydrogen atom vectors are the connection points, which can be selected by the user. Covalent bonds generated by the algorithm for linking fragments are single bonds. The scoring function is a linear combination of two terms with

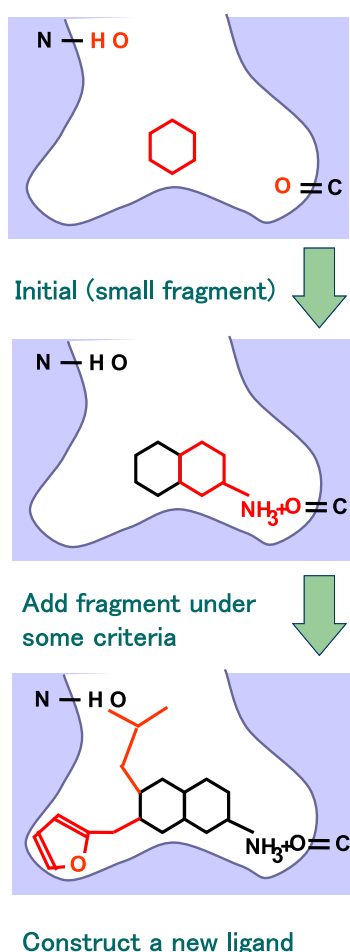


Fig. 2 De Novo ligand design.

multiplicative parameters as input.

The Evolutionary algorithm A version of BOA parallelized for GPU runs the main global optimization problem (i.e., searching the structure search space). Every individual contains a single chromosome consisting of multiple genes. As for encoding, and contrary to classic genetic algorithms, the implementation in BOA uses integers as gene values encoding indexes of docked fragments. Hence, the value of each gene ranges from one to the number of docked poses. **Figure 3** illustrates how the fragments are encoded as chromosomes to be used by the algorithm. Note that the encoded fragments of the children are examined for clashes, and individuals with unfavorable interactions are immediately removed from the population.

Another search algorithm, Tabu search, is used for efficient linking. Linking the encoded fragments for each individual is done by a tabu search. For efficiency reasons (i.e., to avoid conditional branching in the GPU kernel), we built a look-up table containing all distances and angles of all pairs of linker fragment connection vectors. Using cutoff values and the look-up table, all possible connections of fragment pairs of an individual are generated. A connection solution is randomly picked, and the two fragments are joined with the linker defined therein. By employing a simple breadth-first search strategy, fragments of an individual are divided into two sets: a connected set, which are the newly merged fragments and all fragments connected previously, and a not-connected set. A new connection solution is picked with at least one docked fragment being part of the not-connected set, while omitting already occupied connection vectors. The procedure continues until all fragments are connected or a maximal number of connection trials has been exceeded. The score is calculated for the built-up molecule (see below), and the merging procedure is repeated for a user-defined amount of iterations, storing only the docked fragments-linkers assembly with the lowest score for every individual. A tabu list of previously visited solutions is used during these iterations to avoid repeated calculation of scores.

Scoring The scoring function implemented is a linear combination, i.e., a weighted-sum, of two terms: a force field-based binding energy E_{ff} and a measure of similarity (Sim_{3D}) to a user-supplied target structure (e.g., a known agonist).

$$S_{total} = w_{ff}E_{ff} - w_{3D}Sim_{3D}$$

where the multiplicative parameters w_{ff} and w_{3D} are input values. The minus signs for the similarity term is used because optimization is performed by minimization of S_{total} while Sim_{3D} grows with increasing similarity. The two scoring terms are evaluated as follows:

- Force field energy function: We utilize nVidia's Bio Workbench [32], accelerating energy calculation, to calculate binding energy between the ligand and the receptor protein. To calculate the energy, the force field energy function is used. The force field-based energy function consists of van der Waals and electrostatic terms. Both intraligand (intra) and ligand/receptor (inter) interactions are taken into account.

$$E_{ff} = E_{inter}^{vdW} + E_{inter}^{elec} + E_{intra}^{vdW} + E_{intra}^{elec}$$

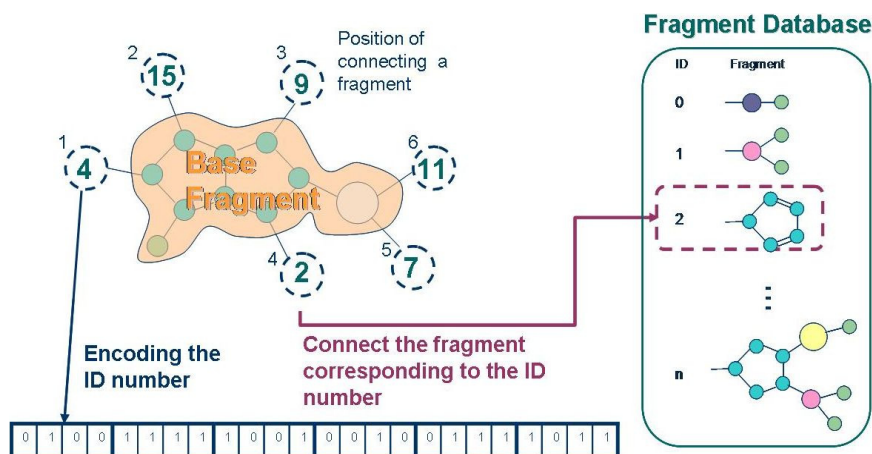


Fig. 3 Encoding of chromosomes representing compounds.

Intrafragment and intralinker interactions as well as fragment-linker interactions between atoms separated by one or two covalent bonds are not evaluated. The potential of the receptor is calculated and stored on a grid [22] and is used only for the linkers. The energies of the fragment poses are read in from the MOL2-files to save computational time.

- The 3D structure Similarity Sim_{3D} : between the newly assembled molecule (A) and a user-supplied template molecule (B) is evaluated by

$$Sim_{3D}(A, B) = \frac{S_{AB}}{\max(S_{AA}, S_{BB})}$$

$$S_{XY} = \sum_{i \in X} \sum_{j \in Y} w_{iij} e^{-\gamma r_{ij}^2}$$

where r_{ij} is the distance between two atoms ($i \in$ molecule X, $j \in$ molecule Y), w_{iij} is a matrix whose coefficients reflect the similarity between element types (an unit matrix is currently used), and γ is a coefficient which acts on the broadness of the distribution of the positions. The 3D similarity Sim_{3D} does not explicitly consider the covalent structure of molecules but relies on the arrangement of atoms in space. In this way, significantly different binding modes of the same molecule (or of two very similar molecules, e.g., differing by only a halogen atom) are kept in the population and evolved further if they have a good score. In the present application to LXR, the maximal similarity Sim_{3D} of two individuals within a population was set to 0.8 and γ to 0.9. Individuals of the old population similar to new ones, but with less favorable score, are given an arbitrarily high score to reduce the likelihood of their selection in a subsequent mating step. The size of the old population is adjusted after merging the old and the new population by removing individuals with the least favorable score.

Protein preparation Liver X Receptors (LXRs) — members of a super family of nuclear hormone receptors and represented by two subtypes, $LXR\alpha$ and $LXR\beta$ — have been shown to be involved in cholesterol homeostasis. Because of the high correlation in binding affinity for the two isoforms, and the high sequence identity in the ligand binding domains (77%), only one isoform was employed for our study. The crystal structure of $LXR\beta$ (PDB code: 1PQ6, 2.4 Å resolution, $R_{free} = 0.262$) was selected as a representative receptor structure for the docking of the

compound library because of the higher resolution of the crystal structure for the human receptor (2.40 Å for the β isoform compared to 2.90 Å for the highest resolution for a human structure of $LXR\alpha$) [30]. Subsequently, the term LXR shall refer to the $LXR\beta$ isoform. To further relax the protein and obtain a low-energy structure for the docking, the bound agonist and all water molecules were removed, and hydrogens were added to side chains and termini of the protein according to pH 7. CHARMM atom types [7] were assigned, and hydrogens were minimized in the absence of the agonist with the CHARMM force field to a gradient of the energy of $0.01 \text{ kcal mol}^{-1} \text{ \AA}^{-1}$. A distance-dependent dielectric function of $\epsilon(r) = 4r$ and default nonbonding cutoffs of 14 Å were used. The resulting minimum-energy structure was used as template for the docking.

Preparation of fragment library The library of fragments, from which the molecules were constructed, was obtained from Molinspiration Cheminformatics (www.molinspiration.com, March 2011 accession date). The library consisted of 30,000 fragments with one and 30,000 fragments with two connection points occurring in bioactive molecules. CHARMM atom types were assigned, and all fragments were subject to minimization. The connection points defined in the source MOL2-files were used as connection vectors of the fragments, using all possible heavy atom-hydrogen atom vectors. Additionally, geometrically identical vectors or pairs of vectors were searched for by superpositioning all original vectors or pairs of vectors with all possible heavy-atom-hydrogen atom vectors and measuring the 3D-similarity Sim_{3D} of the two structures. The original and superimposed fragments were deemed identical if the similarity was larger than 0.95.

During the evolution process, the process of adding fragments to the base fragment is shown in Fig. 4. When we add a fragment, rotation of the additional fragment is set to determine the degrees to the base fragment, and also, distance between the additional and the base fragments is calculated according to the database of bond distances (i.e., look-up table of linkers). When adding a fragment, rotation of the additional fragment is set to determine the degrees to the base fragment, and also, distance between the additional and the base fragments is calculated according to the database of bond distances. In the rotation process, we calculate

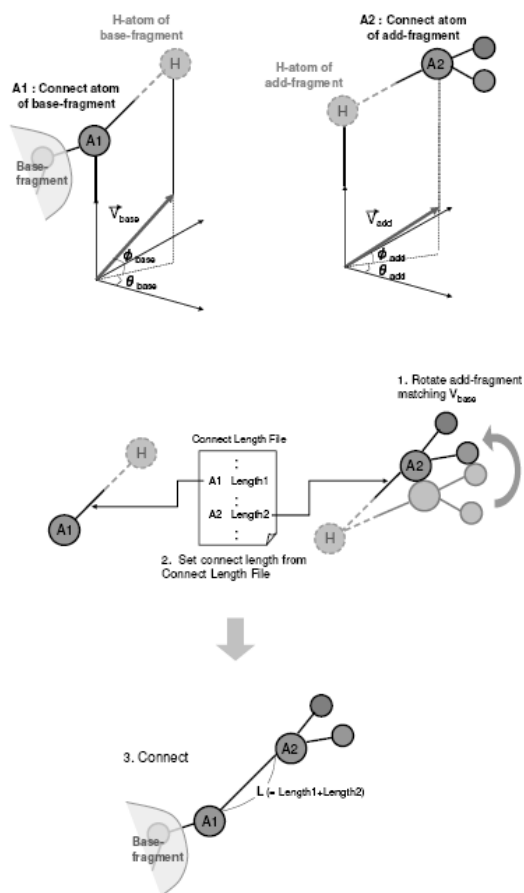


Fig. 4 Process of adding fragments [26].

degrees of rotation θ_{rot} and ϕ_{rot} from those of the base fragment θ_{base} , ϕ_{base} and those of the additional fragment θ_{frag} , ϕ_{frag} as follows: $\theta_{rot} = \theta_{base} - \theta_{frag}$ and $\phi_{rot} = \phi_{base} - \phi_{frag}$.

Employing the degrees calculated as above, we rotate atoms in the additional fragment. First, we perform rotation around y -axis as follows:

$$x' = x \cos(\phi) - z \sin(\phi)$$

$$y' = y$$

$$z' = x \sin(\phi) + z \cos(\phi)$$

Next, rotation around z -axis is performed as follows to obtain final results:

$$x_{rot} = x' \cos(\theta) + y' \sin(\theta)$$

$$y_{rot} = -x' \sin(\theta) + y' \cos(\theta)$$

$$z_{rot} = z'$$

After the rotation is finished, we calculate bond distances based on the database of the distances empirically calculated.

Docking of fragments Of the 60,000 fragments in the library only the 13,788 containing less than four rotatable bonds were used. Their properties are shown in Fig. 5 (histogram). Of these, 6,906 and 6,882 have one and two connection vectors, respectively. They were docked into the receptor binding site with SEED [22], a program for docking mainly rigid fragments with evaluation of protein-fragment energy and electrostatic desolvation. Note that the 6,882 fragments with two connection vectors were used also as linker fragments. A maximal number of

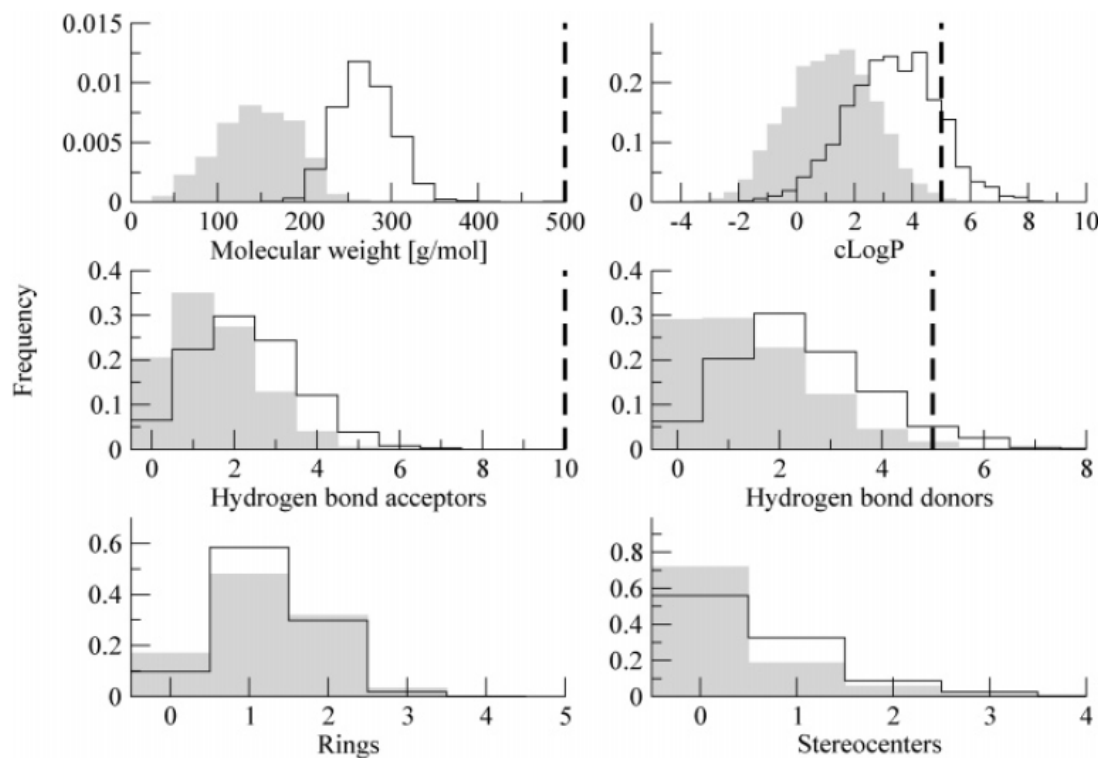


Fig. 5 Normalized distributions of molecular properties calculated by DAIM [16]. Histograms for the 13,788 fragments docked by SEED [22] are in gray and shaded, and those for the 1,809 unique molecules obtained by the 10 runs with the least w_{ff} are black and empty. The vertical thick dashed lines represent the thresholds defined by Lipinski's rule of five [21].

20 cluster representative positions of each fragment type were kept, using also a cutoff value of -5 kcal mol^{-1} for the SEED energy. Fragments were discarded if the difference in energy for any vector between bound and unbound states was larger than 5 kcal mol^{-1} . This procedure yielded a total of 8,506 docked fragments with 72,822 poses.

2.3 Implementation over GPU

GPU is emerging as one of the most powerful parallel processing devices. GPU is especially well-suited to address problems that can be expressed as data-parallel computations with high arithmetic intensity (i.e., ratio of arithmetic operations to memory operations). Applications that process large data sets can use a data-parallel programming model to speed up the computations. In 3D rendering, large sets of pixels and vertices are mapped to parallel threads. In the recent years, many algorithms outside the field of image rendering and processing were also accelerated by data-parallel processing.

Although GPUs can offer unprecedented performance gain, implementation of an algorithm over a GPU to take full advantage of this new technology involves a significant complexity of parallelizing across the multiple cores. Memory management over a GPU makes things even more challenging. CUDA [1] is a parallel computing architecture developed by nVidia. CUDA is the compute engine in nVidia’s CUDA compatible GPUs, and is accessible to software developers through industry standard programming languages like C. CUDA is widely used for programming nVidia’s GPUs for general purpose processing.

Figure 6 illustrates the architecture of nVidia Tesla GPU (used for this research) [20]. The GPU runs its own specified instructions independently from the CPU but it is controlled by the CPU. A thread is the computing element in the GPU. When a GPU instruction is invoked, blocks of threads are defined to assign one thread to each data element. All threads in one block run the same instruction on one streaming Multi Processor (MP), which gives the GPU an SIMD/SIMT architecture. Each MP includes 8 stream processor (SP) cores, an instruction unit, and on-chip memory that come in three types: registers, shared memory, and cache (constant and texture). As shown in Fig. 6, threads in each block have access to the shared memory in the MP, as well as to a global memory in the GPU.

The algorithm used in this proposed system is given at Fig. 7. Earlier work in Ref. [25] elaborates on the optimization and tuning for BOA over GPU. The parallelization this paper follows the conventional Master-Slave model. The entire algorithm runs

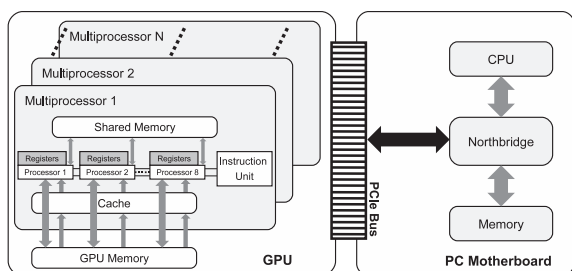


Fig. 6 Hardware architecture of GPU mounted on the PC motherboard.

over the host while evaluation of the scoring, Sim_{3D} and Tabu searching for efficient linking. The implementation of this algorithm over GPU is shown in Fig. 8. The figure shows the CPU and the GPU side portions of the algorithm. All the configurations, memory allocations, initializations are performed over the host processor. After the initialization stage (which includes loading the pre-docked fragments and encoding them to a code table), data is transferred to the device. Then the code running at the host side enters a loop. At that loop, breadth-first search detects all the fragments eligible for binding. Next, the codes of the fragments are transferred to the device. At this point the kernel starts running. The kernel uses the fragments code table and the codes of the fragments in the structure to be evaluated, then decodes the codes to create the structure in thread’s registers. Then the kernel runs the force-field energy of the structure. Finally the kernel computes the fitness (S_{total}) after computing Sim_{3D} between the structure and the template then searching for an efficient linking. Next, the computation transfers back to the host

-
- 1 : Pre-docked fragments are prepared;
 - 2 : Generate initial population;
 - 3 : Start evaluating the individuals by:
 - 4 : For each individual, pre-docked fragments are added to the base fragment;
 - 5 : Calculate the force field energy and the Sim_{3D} :[GPU]
 - 6 : Tabu search for efficient linking:[GPU]
 - 7 : Calculate the fitness function of the individual:[GPU]
 - 8 : Select the fittest individuals based on the fitness value;
 - 9 : Create a Bayesian Network from the selected individuals;
 - 10 : Generate offspring from the individuals sampled from the BN and current population;
 - 11 : Unless terminated, go to 3;
-

Fig. 7 Algorithm of fragment-based drug design using BOA.

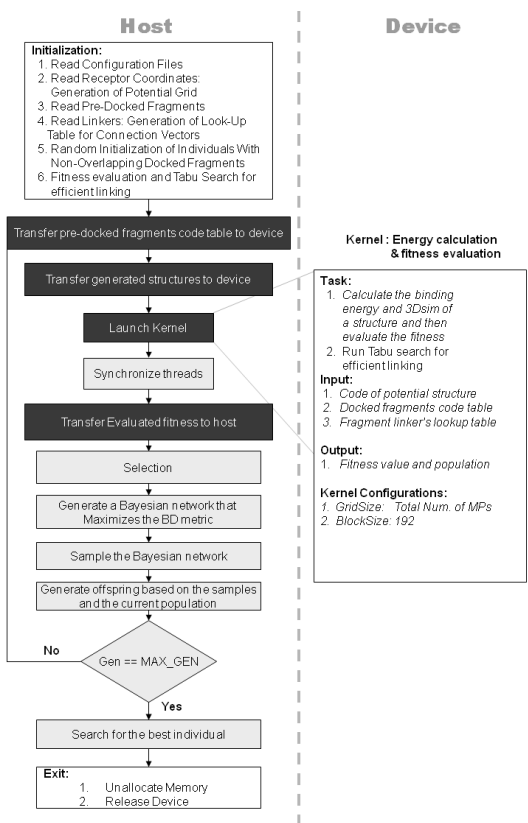


Fig. 8 Flow chart of CPU (host) side and GPU (device) side logic.

after receiving the fitness values. The host then proceeds with the algorithm by selecting candidates from the population to build the Bayesian network. Truncation selection has been used as advised by Ref. [19] to better maintain the building block. Next, the network is constructed using the BD metric as mentioned earlier and the network is sampled to generate new candidate solutions. Lastly, the offspring individuals are inserted into the parent population if no structurally similar parent has a more favorable score to intensify the selection pressure. The cycle is then repeated until a termination criteria is met. The kernel in this algorithm does the following:

- Construct a structure using a code table and a vector of codes representing the potential fragments. A technique inspired from the raster document coders that are typically based on the use of a binary mask layer that efficiently encodes/decodes the text and graphic content [8]. These methods are originally used to yield high compression ratios than natural image compression methods. Yet, we use the technique here to SIMDize the decoding process for optimized performance on GPU.
- The Kernel calculates the force-field energy by running the force-field function of Tesla's Bio Workbench. This part was optimized by using the techniques in Ref. [24]. The techniques that were mostly effective are a) optimizing memory access and reduce memory management overhead. b) Eliminating all the conditional branches except for an unavoidable condition. c) Optimizing the frequent memory transfer. d) Blocks/threads topology was optimized to achieve the highest occupancy possible.
- Next, similarity between the structure and the template is computed. This part was easily ported to be data parallel.
- Tabu search is conducted to find the most efficient linking mode of the fragments.
- Finally fitness is evaluated, and the result is stored in a specific location to make the result vector passable to the host in one step.

3. Results & Discussion

Results given in this section were collected over a system with nVidia Tesla C2050 GPU mounted on a motherboard with Intel® Core™ i7 920@2.67 GHz as the host CPU. C2050 has 3 GB of device memory and the total number of processing cores is 448. The maximum amount of shared memory per block is 64 KB (16 KB was used as cache as advised by CUDA manual) and clock rate is 1.5 GHz. We are using Fedora Core 13 as the operating system and CUDA SDK/Toolkit ver. 4.0 with nVidia driver ver. 270.41.19. Other tools used for optimization and profiling include *CudaVisualProfiler* and *CudaOccupancyCalculator*. C2050 is dedicated to computations only. The system has a separate GeForce 8400 GS GPU acting as a display card.

Setting of Algorithm Runs Calculations were repeated 10 times for each of three settings (i.e., weighted coefficient phasing as seen below) with distinct random seed numbers for 1,000 iterations of the algorithm and 20 iterations of the tabu search per individual. The minimized phenyl acetic acid-based agonist cocrystallized with the protein (PDB code 1PQ6) was used as a

target structure. The coefficients of the scoring function terms were set to $\{w_{ff} = 0.02, w_{3D} = 0.98\}$, $\{w_{ff} = 0.06, w_{3D} = 0.94\}$ and $\{w_{ff} = 0.10, w_{3D} = 0.90\}$. The 13,788 Molinspiration-library fragments with less than four rotatable bonds were used in each of the runs for the three different coefficient pair value. The 6,882 Molinspiration-library fragments with two connection vectors and up to three rotatable bonds were used as linker fragments with a total of 17,372 unique vector pairs. To prevent the generation of unstable molecules, bonds between nitrogen, oxygen, and sulfur atoms were avoided by using a list of forbidden connections (S-S, S-O, S-N, O-O, O-N, N-N) [29]. To maintain diversity throughout the optimization procedure, the maximal 3D-similarity of two individuals was set to 0.8 and coefficient γ to 0.9. All molecules were backed up to disk by the host stored to disk every 20th iterations of the optimization procedure. A single run with 1,000 iterations (restarted 10 times once every 100 iterations) took 74.2 min (SD 3.6%).

Optimization The 3D-similarity term Sim_{3D} in the scoring function seems to dominate the total score given the coefficients of the weighted-sum approach described earlier (see Fig. 9. Yet, the force field-based term E_{ff} is essential for optimizing both binding interactions and intraligand interactions. In fact, the 3D-similarity to the target molecule decreases almost steadily during optimization (Fig. 9, middle), while the force field-based term (Fig. 9, top) decreases fast initially and fluctuates within a certain range afterward. The medium w_{ff} setup (red) displays the fastest decrease in the total score of the fittest individual, while the setting with the least w_{ff} decreases to similar final values. On the other hand, running the algorithm with higher w_{ff} value (i.e., rely more on binding energy) results in worse scores throughout. Similarly, the overlapping volumes between the target structure and the molecules generated increase fast initially and only change marginally afterward (Fig. 10). The increasing volume overlap is a direct effect of the 3D-similarity based scoring function term Sim_{3D} , which penalizes deviations of the generated molecule and its binding mode from the target agonist. Taken together, the results indicate that the size of the population predetermines the amount of search space to be sampled and thus has a direct effect on the quality of the solutions found. Hence, the larger the number of individuals which can exchange genetic material, the faster the decrease in total scores. But these enlarged populations are associated with an increase in CPU time, which might offset the overall improvement. This gives rationale to utilizing GPU for such a problem as assigning a thread per individual gives a virtual unbound limit on the population size. However, the challenge would be optimizing and tuning the GPU kernel to achieve high efficiency for a problem that is not naturally SIMT aligned.

Molecules designed by the algorithm Several of the 100 generated molecules with the most favorable S_{total} when compared with the crystal structure of LXR in the complex with the heterocyclic phenylacetic agonist shows that the generated molecules include key motifs of the target structure, e.g., the two ring systems joined by a linker (Table 1, compound 2). Compounds generated by the algorithm in Table 1 are a consequence of the enforced 3D-structural diversity within populations during optimization. Analysis of the existence of the generated molecules

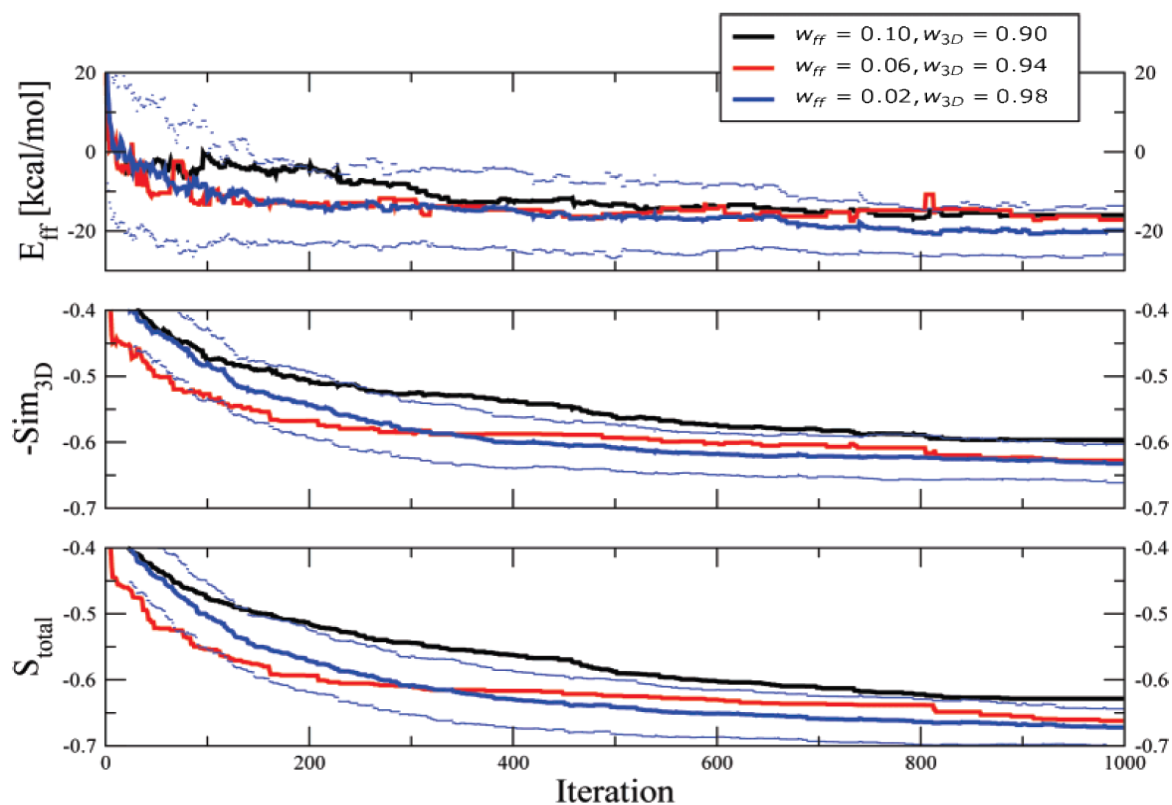
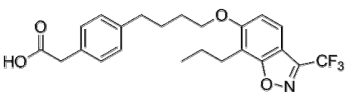
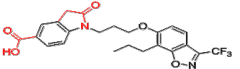
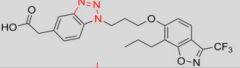
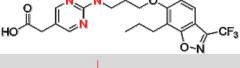

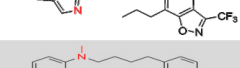
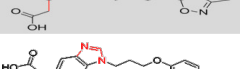
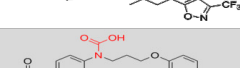



Fig. 9 Evolution of the scoring function terms and the total score of the best individual in each weight setting. The values at each iteration step were averaged over 10 runs. The bold lines are averages, while the thin blue lines are the standard deviation of the run with the least w_{ff} .

Table 1 Molecules with favorable score S_{total} generated in the 10 runs with $w_{ff} = 0.02$.

Structure	Similarity to agonist (PDB code 1PQ6) Sim_{3D}	Scoring S_{Total}	E_{ff} [kcal/mol]	MW [g/mol]
Target (1PQ6) 	(1)	-	-10.7	582.5
Generated molecules with optimal score S_{Total} 	0.705	-0.721	-10.0	535.2
	0.666	-0.694	-10.1	547.8
	0.659	-0.687	-9.7	539.0
	0.642	-0.673	-9.5	567.3
	0.573	-0.541	-10.2	559.4
	0.512	-0.525	-10.5	571.5
	0.510	-0.516	-10.8	541.7
	0.507	-0.499	-10.6	589.3

in the ZINC library [15] reveals that eight out of 1,809 generated molecules are commercially available. Additionally, 586 molecules with a scaffold identical to one of the eight compounds

shown in Table 1 are retrieved from the ZINC library by performing a substructure search with DAIM [16] in Table 2. To search the ZINC database for structures with identical scaffolds, first

the generated structures with favorable scoring were decomposed into fragments by DAIM. The 1,640 molecules which yielded at least three fragments upon decomposition with DAIM were re-docked into the rigid protein binding site. These molecules were first minimized in the absence of the protein to remove any conformational bias introduced by the algorithm. The fragments obtained by decomposition with DAIM were then minimized after updating the partial charges and atom types and docked into the receptor binding site with SEED.

Details of the predicted binding mode for Table 1, compound

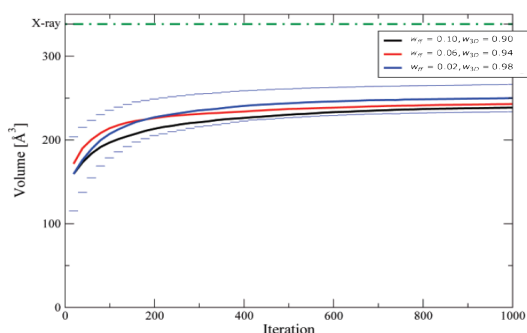


Fig. 10 Evolution of the volume overlap between the crystallographic agonist and the molecules produced by our algorithm. Every 20th iteration step the volume overlap is averaged over all individuals and all 10 runs. The thin blue lines are the standard deviation of the run with the least w_{ff} setting. The volume of the minimized crystallographic agonist (338.37 \AA^3) is shown in green.

Table 2 Average and median of DAIM-fingerprint entries of the ZINC library.

DAIM-Fingerprint	Average \pm Std.Dev.	Median	Normalization factors
# Atoms	44.99 ± 11.00	45	45
# Carbon	19.17 ± 5.09	19	19
# Nitrogen	2.54 ± 1.45	2	2
# Oxygen	2.88 ± 1.66	3	3
# Halogen	0.70 ± 1.06	0	1
# Sulphur	0.53 ± 0.68	0	1
# Phosphorus	$2.79 \times 10^{-3} \pm 0.05$	0	0 ^a
# Aromatic Bonds	10.69 ± 5.23	12	12
# Double Bonds	3.16 ± 1.69	3	3
# Triple Bonds	0.07 ± 0.28	0	0 ^a
# Amide Bonds	0.96 ± 0.94	1	1
# Hydrogen Acceptors	3.76 ± 1.67	4	4
# Hydrogen Donors	1.25 ± 1.03	1	1
# Rings	3.13 ± 1.14	3	3
Total Ring Size	16.72 ± 5.64	17	17
Longest Chain	15.90 ± 3.18	16	16
Wiener Index 4 ^b	3.31 ± 1.19	3.21	3.21

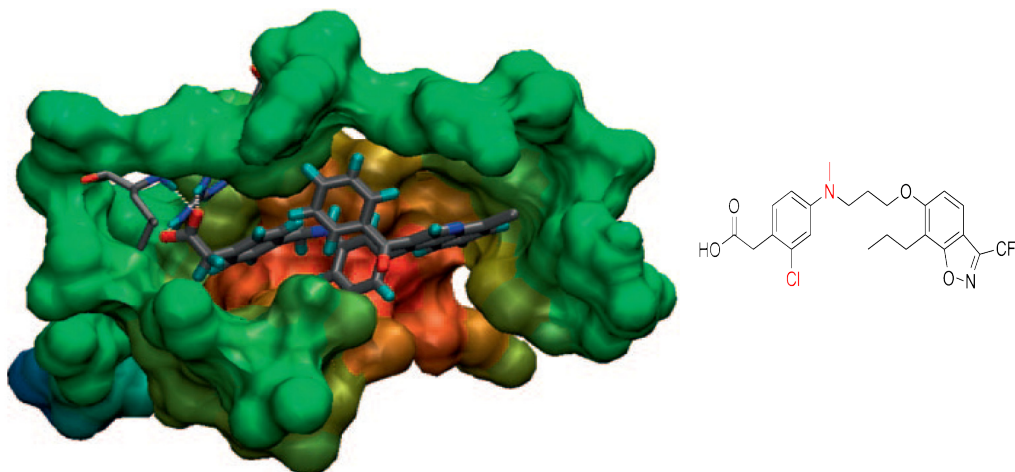


Fig. 11 Details of the predicted binding mode for the phenyl acetic acid based structure (left). The chemical structure is shown on the right.

7 are shown in **Fig. 11**.

Speedups For comparison, we refrained from comparing the performance with serial implementations as GPU-implementation literature usually do. This is because comparing excessively large problems to serial implementations cause an expected inflation of speedup which does not give a realistic measure of the impact of GPUs. The environment for comparison is as follows: a) a mini cluster of 16 nodes with AMD Opteron 2.6 GHz Dual Core 64 bit processors and 2 GB RAM for each node, the system also has 2 dedicated servers each having a Xeon 2.8 GHz Quadcore with 2 GB RAM. Therefore performance of GPU is compared to a system having 40 cores. The MPI communication library used is MPICH2, 1.2.1. b) OpenMP [14] based parallel implementation over Intel i7 920@2.67 GHz CPU (4 cores/8 threads) with 4 GB memory. All the implementations are optimized for the hardware they are running on. Moreover, for all the implementations we are using the compiler directives to maximize the execution speed. Five different implementations used to obtain the results are shown in **Table 3**.

Table 4 shows the average execution time required and the standard deviation for generating structures with scaffolds identical to the compounds in Table 1 over different kinds of parallel architectures. It is clear from the table that we get a significant reduction in the execution time when we implement the algorithm over the GPU. Note that the results are for double precision. Single precision would almost double the speed for the GPU, yet it was avoided to avoid compromising the output quality.

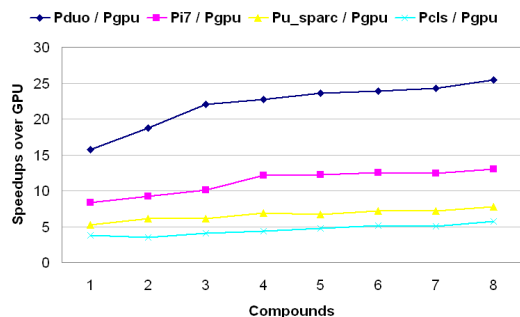
The speedups as compared with other parallel implementa-

Table 3 Different implementations used to obtain the results.

P_{cls}	Parallel implementation over a cluster of 40 cores
P_{duo}	OpenMP based parallel implementation over Intel® Core™2 Duo E8600@3.33 GHz CPU (2 cores/2 threads) with 4 GB of memory.
P_{u_sparc}	OpenMP based parallel implementation over Sun Ultra Sparc T2 1.16 GHz CPU (8 cores/64 threads), 16 GB memory.
P_{i7}	OpenMP based parallel implementation over Intel i7 2.67 GHz CPU (4 cores/8 threads) with 4 GB memory.
P_{gpu}	Implementation of the proposed algorithm over nVidia C2050 GPU.

Table 4 Average execution time required for generating structures with scaffolds identical to the eight compounds in Table 1. All the results shown in this table are an average of 10 independent runs.

Compound	# Structures with identical scaffolds				Average Execution Time (Min) \pm Standard Deviation				
	P ₁₇	P _{sparc}	P _{cls}	P _{gpu}	Parallel _{duo}	Parallel ₁₇	Parallel _{sparc}	Parallel _{cls}	Parallel _{gpu}
1	27	27	27	28	0.21 \pm 0.034	0.112 \pm 0.001	0.050 \pm 0.003	0.132 \pm 0.004	0.072 \pm 0.005
2	30	30	30	30	1.38 \pm 0.269	0.676 \pm 0.063	0.260 \pm 0.028	0.189 \pm 0.033	0.097 \pm 0.008
3	26	25	27	26	2.87 \pm 0.252	1.530 \pm 0.233	0.556 \pm 0.184	0.266 \pm 0.019	0.143 \pm 0.021
4	20	20	20	21	1.48 \pm 0.258	0.768 \pm 0.071	0.300 \pm 0.119	0.145 \pm 0.008	0.081 \pm 0.002
5	23	22	23	21	5.92 \pm 1.002	3.021 \pm 0.213	1.235 \pm 0.195	0.416 \pm 0.068	0.226 \pm 0.045
6	19	19	19	18	3.21 \pm 0.700	1.648 \pm 0.185	0.722 \pm 0.050	0.294 \pm 0.072	0.151 \pm 0.081
7	30	30	30	30	10.86 \pm 1.517	5.575 \pm 0.794	2.442 \pm 0.282	0.701 \pm 0.123	0.370 \pm 0.082
8	29	29	29	31	8.55 \pm 0.428	4.401 \pm 0.819	1.910 \pm 0.057	0.586 \pm 0.075	0.306 \pm 0.078


Fig. 12 Speedups achieved in double precision implementation over nVidia Fermi GPU as compared with different parallel implementations.

tions over are shown in Fig. 12. Even comparing our results with state-of-the-art processors like Intel Core i7 2.67 GHz 4 cores/8 threads, Sun Ultra Sparc 1.167 GHz 8 cores/64 threads and a 40 core mini-cluster yielded a maximum speedup of up to 12x, 8x and 6x respectively. These results clearly show the significance of using GPUs in the area combinatorial drug design.

4. Conclusion

This paper presented a system for fragment-based De novo ligand design. A combination of an evolutionary algorithm (BOA) and tabu search is used for the simultaneous optimization of force field energy and 3D similarity to known agonist(s). Therefore, the design is both binding site-based and ligand-based. Importantly, the relative importance of these two driving forces can be modulated by the user. Due to the proven high computational cost needed to run simulations for complex structures, the entire system was implemented to run over GPU (nVidia's Tesla architecture) in an attempt to accelerate the performance on the GPU instead of high cost of conventional clusters.

In an application to the liver x receptors (LXRs), 1,809 molecules were generated by the algorithm within the ATP-binding site in less than 16 h on a Tesla C2050 using a library of 14,000 fragments with up to three rotatable bonds. Notably, molecules similar to those generated by the algorithm are commercially available providing further evidence of the usefulness of the proposed system for De novo drug design. The algorithm can generate molecules similar to known LXR agonists. Importantly, by enforcing diversity throughout the optimization and by using a 3D-similarity-based scoring function term Sim3D, which does not rely on a covalent structure of the compared molecules, scaffold or linker hopping was observed, retaining the common binding motifs of known LXR agonists. Because a large variety

of ATP-binding site agonists of LXR form hydrogen bonds with the hinge region, in the present application of the algorithm, the 3D-similarity to the binding mode of a known agonist of LXR was enforced (using Sim3D).

Acknowledgments This work was supported by KAKENHI (No.22500196), Japan Society for the Promotion of Science (JSPS).

Reference

- [1] nVidia CUDA Programming Guide – CUDA 4.0 SDK Documentation (2011).
- [2] Belda, I., Llorà, X. and Giralt, E.: Evolutionary algorithms and de novo peptide design, *Soft Comput.*, Vol.10, No.4, pp.295–304 (2006).
- [3] Belda, I., Llorà, X. and Giralt, E.: Evolutionary algorithms and de novo peptide design, *Soft Comput.*, Vol.10, pp.295–304 (2006).
- [4] Bonn, M., Sun, T., Ljunggren, S., Ahola, J., Wilhelmsson, H., Gustafsson, A., Jan-Ake and Mats, C.: The three-dimensional structure of the liver X receptor beta reveals a flexible ligand-binding pocket that can accommodate fundamentally different ligands, *Journal of Biological Chemistry*, Vol.278, No.40, pp.38821–38828 (2003).
- [5] Douguet, D., Munier-Lehmann, H., Labesse, G. and Pochet, S.: LEA3D: A Computer-Aided Ligand Design for Structure-Based Drug Design, *J. Med. Chem.*, Vol.48, pp.2457–2468 (2005).
- [6] Douguet, D., Munier-Lehmann, H., Labesse, G. and Pochet, S.: LEA3D: A Computer-Aided Ligand Design for Structure-Based Drug Design, *Medical Chemistry Journal*, Vol.48, pp.2457–2468 (2005).
- [7] Momany, F. and Rone, R.: Validation of the general purpose QUANTA 3.2/CHARMm force field, *Journal of Computational Chemistry*, Vol.13, pp.888–900 (1992).
- [8] Feng, G., Cheng, H. and Bouman, C.: Rate Distortion Optimized Document Coding using Resolution Enhanced Rendering, *Proc. 2001 International Conference on Image Processing*, Vol.3, pp.430–433 (2001).
- [9] Firth-Clark, S., Todorov, N.P., Alberts, I.L., Williams, A., James, T. and Dean, P.M.: Exhaustive de novo Design of Low-Molecular-Weight Fragments Against the ATP-Binding Site of DNA-Gyrase, *Journal of Chemical Information and Modeling*, Vol.46, No.3, pp.1168–1173 (2006).
- [10] Ghosh, A. and Sen, A.: Hybrid-genetic algorithms for flexible ligand docking, *J. Intell. Fuzzy Syst.*, Vol.18, No.6, pp.561–574 (2007).
- [11] Glover, F. and Laguna, M.: *Tabu Search*, Kluwer Academic Publishers, Norwell, MA, USA (1997).
- [12] Heckerman, D.: A Tutorial on Learning With Bayesian Networks, Technical report, Learning in Graphical Models (1996).
- [13] Heckerman, D., Geiger, D. and Chickering, D.: Learning Bayesian Networks: The Combination of Knowledge and Statistical Data, *Machine Learning*, Vol.12, pp.197–243 (1994).
- [14] OpenMP, available from (<http://www.openmp.org/>).
- [15] Irwin, J.J. and Shoichet, B.K.: ZINC – A Free Database of Commercially Available Compounds for Virtual Screening, *Journal of Chemical Information and Modeling*, Vol.45, No.1, pp.177–182 (2005).
- [16] Kolb, P. and Caffisch, A.: Automatic and Efficient Decomposition of Two-Dimensional Structures of Small Molecules for Fragment-Based High-Throughput Docking, *J. Med. Chem.*, Vol.49, pp.7384–7392 (2006).
- [17] Lameijer, E.-W., Bäck, T., Kok, J.N. and Ijzerman, A.P.: Evolutionary Algorithms in Drug Design, Vol.4, pp.177–243 (2005).
- [18] Lameijer, E.-W., Bäck, T., Kok, J.N. and Ijzerman, A.P.: Evolutionary Algorithms in Drug Design, *Natural Computing: An International*

- Journal*, Vol.4, No.3, pp.177–243 (2005).
- [19] Lima, C.F., Pelikan, M., Goldberg, D.E., Lobo, F.G., Sastry, K. and Hauschild, M.: Influence of selection and replacement strategies on linkage learning in BOA, *IEEE Congress on Evolutionary Computation*, Vol.1, pp.1083–1090 (2007).
 - [20] Lindholm, E., Nickolls, J., Oberman, S. and Montrym, J.: NVIDIA Tesla: A Unified Graphics and Computing Architecture, *IEEE Micro*, Vol.28, No.2, pp.39–55 (2008).
 - [21] Lipinski, C.A., Lombardo, F., Dominy, B.W. and Feeney, P.J.: Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings, *Advanced Drug Delivery Reviews*, Vol.46, pp.3–26 (2001).
 - [22] Majeux, N., Scarsi, M., Apostolakis, J., Ehrhardt, C. and Cafilisch, A.: Exhaustive docking of molecular fragments with electrostatic solvation, *Proteins: Struct. Funct. Genet.*, Vol.37, No.1, pp.88–105 (1999).
 - [23] Muhlenbein, H., Mahnig, T. and Rodriguez, A.O.: Schemata, Distributions and Graphical Models in Evolutionary Optimization, *Journal of Heuristics*, Vol.5, pp.215–247 (1999).
 - [24] Munawar, A., Wahib, M., Munetomo, M. and Akama, K.: Hybrid of genetic algorithm and local search to solve MAX-SAT problem using nVidia CUDA framework, *Genetic Programming and Evolvable Machines*, Vol.10, No.4, pp.391–415 (2009).
 - [25] Munawar, A., Wahib, M., Munetomo, M. and Akama, K.: Theoretical and Empirical Analysis of a GPU based Parallel Bayesian Optimization Algorithm, *PDAA'09: Proc. International Workshop on Parallel and Distributed Algorithms and Applications*, pp.127–133, IEEE Press (2009).
 - [26] Munetomo, M., Akama, K. and Maeda, H.: De novo ligand evolution using Bayesian Optimization Algorithms, *EC'09: Proc. 10th WSEAS International Conference on Evolutionary Computing*, pp.126–131, World Scientific and Engineering Academy and Society (WSEAS) (2009).
 - [27] Ocenasek, J. and Schwarz, J.: The Parallel Bayesian Optimization Algorithm, *Proc. European Symposium on Computational Intelligence*, pp.61–67, Springer Verlag (2000).
 - [28] Pelikan, M., Goldberg, D.E. and Cantu-Paz, E.: *BOA: The Bayesian Optimization Algorithm*, Morgan Kaufmann, pp.525–532 (1999).
 - [29] Wang, R., Gao, Y. and Lai, L.: A Multi-Purpose Program for Structure-Based Drug Design, *Journal of Molecular Modeling*, Vol.6, pp.498–516 (2000).
 - [30] Svensson, S., Östberg, T., Jacobsson, M., Norström, C., Stefansson, K., Hallén, D., Johansson, I.C., Zachrisson, K., Ogg, D. and Jendeborg, L.: Crystal structure of the heterodimeric complex of LXR alpha and RXR beta ligand-binding domains in a fully agonistic conformation, *EMBO Journal*, Vol.22, pp.4625–4633 (2003).
 - [31] Schneider, G., Lee, M.-L., Stahl, M. and Schneider, P.: De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks, *Journal of Computer-Aided Molecular Design*, Vol.14, No.5, pp.487–494 (2000).
 - [32] Stone, J.E., Phillips, J.C., Freddolino, P.L., Hardy, D.J., Trabuco, L.G. and Schulten, K.: Accelerating molecular modeling applications with graphics processors, *Journal of Computational Chemistry*, Vol.28, pp.2618–2640 (2007).
 - [33] Fechner, U. and Schneider, G.: Flux (2): Comparison of Molecular Mutation and Crossover Operators for Ligand-Based de Novo Design, *Journal of Chemical Information and Modeling*, Vol.47, pp.656–667 (2007).



Mohamed Wahib was born in 1980. He received his B.E. degree from Suez Canal University Egypt in 2001. He worked as a teaching assistant in the Computer Science Department, Suez Canal University until 2006. He later received MEXT scholarship for graduate studies from the government of Japan, and now, he is

working on his Ph.D. in Hokkaido University's Information Initiative Center.



and Technology, Hokkaido University.

Asim Munawar was born in 1981. He received his B.E. from National University of Science & Technology, Pakistan in 2003. After that he was engaged in a research organization named Center for Advanced Research in Engineering (CARE) for 2 years. He received his Ph.D. from Graduate School of Information Science



1996 respectively from the same university. He also worked as a visiting researcher at Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois at Urbana-Champaign, Illinois, USA.

Masaharu Munetomo is an associate professor at Information Systems Design Laboratory, Division of Large Scale Computing Systems, Information Initiative Center, Hokkaido University, Japan. He finished his bachelor's degree from Hokkaido University in 1991. After that he finished M.Sc. and Ph.D. in 1993 and



(Faculty of Letters). In 1989 he became an associate professor (Faculty of Engineering). Since 1999 he has been working as a professor in Hokkaido University.

Kiyoshi Akama is a professor at Information Systems Design Laboratory, Division of Large Scale Computing Systems, Information Initiative Center, Hokkaido University, Japan. He started as an assistant in Tokyo Institute of Technology in 1979. Later on he took instructor position at Hokkaido University, Instructor

(Communicated by *Masakazu Sekijima*)