International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Metaheuristics Based Optimization Technique for Protein-Ligand Docking

Abhishek.K, S. Balaji

ABSTRACT--- Virtual screening using molecular docking requires optimization, which can be solved by using metaheuristics methods. Typically the interaction between two compounds is calculated using computationally intensive Scoring Functions (SF) which is computed in several spots which are called as binding surfaces. In this paper we present a novel approach for molecular docking which is based on parameterized and parallel metaheuristics which is useful in leveraging heterogeneous computing based on heterogeneous architectures. The approach decides on the optimization technique at running time by setting up a new configuration schema that allows parallel offloading of the data intensive sections of the docking. Hence the docking process is carried out in parallel efficiently while performing the metaheuristics execution. The approach carries out docking and computations of molecular interactions required for SF in parallel so that the time efficiency is improved. This opens a new path for further developments in virtual screening methods in heterogeneous platform.

Keywords: Drug discovery, virtual screening, molecular docking, high performance computing, metaheuristics, heterogeneous computing

I. INTRODUCTION

Virtual Screening (VS) is very useful in the process of drug discovery and it uses computationally intensive techniques which can be used to analyze huge libraries of tiny molecules to search for compounds which show affinity towards binding with the target. Typically, these libraries consist of millions of ligands which generate large number of hits. The complexity of VS is decided by two parameters: database size and the accuracy level. Hence, there is a need for different computational techniques that enable docking and calculation of binding affinity of different poses in parallel. This, in turn, enables efficient docking using heterogeneous platform [1-4].

There is a steady transition from normal computing to heterogeneous computing in which CPU is combined with GPU having many cores, which has the capability to speed up the computationally demanding parts of the docking process. Run time parameter is still under consideration which helps in dynamic load balancing and offloading. In particular, concepts like data organization and programmability are still challenges that do exist in heterogeneous platform [6].

The researchers are focusing on various other techniques which can be applied to docking process like image processing, computational modeling, metaheuristics, etc. This results in up scaling of efficiency in computer driven scientific applications. Programmers play an important role in improving the current application scenario and how the parallelism is achieved. Programmers have to rethink or redesign the sections which cause bottlenecks for the whole process. We chose metaheuristics algorithms since they are best suited for the current application though there are various algorithms which suit well for this computing area. Metaheuristics are usually applied to solve Non-Polynomial (NP) hard problems. Many problems from bioinformatics use this approach. For example, DNA analysis applies the method to find out the sequence of chains. A tuning process is necessary to select the metaheuristics to minimize the computational cost [5-9].

We introduce a new technique for VS which generates the metaheuristics schema needed for docking. This is designed for leveraging the heterogeneous architectures. The objective of this technique is to predict the binding confirmations using SF. These functions can be used to compute binding score throughout the protein surface. Following are the highlights of this technique:

1. The methodology can generate a parameterized schema based on metaheuristics based on different set of input parameters.
2. This technique leverages the heterogeneous computing based on CPUs and NVIDIA GPUs having different capabilities. Load balancing strategy has been introduced to distribute workloads among all GPUs in the system
3. About the performance, the following observations can be made:

a. Distribution of workload in a homogeneous manner is not good for systems with GPUs with different computational capabilities

b. Only technical specifications are not enough to achieve peak performance, and there is a need for load balancing at run time, with the workload depending on the application reliability

All these strategies give the opportunity to improve the solution quality in docking.

Manuscript published on 30 September 2019
* Correspondence Author
Abhishek.K*, Research Scholar-Jain University, Dept. of Information Science & Engineering., Jyothy Institute of Technology, Tataguni,Bengaluru-560082, India, (E-mail abhishek.mtech2012@gmail.com)
S. Balaji, Centre for Incubation, Innovation, Research and Consultancy, Jyothy Institute of Technology, Tataguni, Off Kanakapura Road, Bengaluru-560082, India, (E-mail drsbalaji@gmail.com)

II. BACKGROUND

This section discusses about virtual screening methods, metaheuristics and GPU computing which lays foundation for the next sections.

2.1 GPU Computing

The computer architects have been depending on the CPU for all the computations for decades. Of late, heterogeneous architecture has started gaining importance since the data intensive operations have seen a spike recently. In a heterogeneous architecture, both CPU and GPU are used for computations wherein the GPU takes the data intensive part of the problem. Compute Unified Device Architecture (CUDA) programming model is important, since it is widely used for GPU computing. The GPU consists of several processors which are replicated in the silicon area. These multiprocessors are connected to GPU device memory. The CUDA capabilities are increasing in each version and also the number of cores. The power efficiency is improved by a factor of two in new generation.

The CUDA software paradigm is based on multiple levels of abstractions. The thread is the basic execution unit which takes up the basic functionality. Threads are grouped into a number of blocks which are mapped to multiprocessors. There are built-in procedures which can offload the blocks to GPU. But, before this, the sections which can be parallelized are ported to GPU kernels. A kernel is a part of program which can be run in parallel without any data dependency or with less dependency. Later grids can be formed and deployed to GPU [10, 14, 18]. Hence, a kernel is executed by many grids of blocks where threads will run concurrently. Table 1 shows CUDA capabilities summary by generation.

Table 1: CUDA Summary by Generation

Hardware generation and starting year	Fermi 2010	Kepler 2012
Multiprocessors per die (up to)	16	15
Cores per multiprocessor	32	192
Total number of cores (up to)	512	2880
Shared memory size (maximum in KB)	48	48
Device memory size (maximum in GB)	6	12
CUDA Compute capabilities	2.x	3.x
Peak single-precision performance (GFLOPS)	1178	4290
Performance per watt (approx. and normalized)	2	6

2.2 Metaheuristics

It is observed that there are many problems which cannot be optimized by evaluating the possible solutions. Problems like NP-hard require alternative approaches rather than traditional approach. Metaheuristics can be applied for problems which involve heterogeneous platforms. The optimum solution for NP-hard problem can be found out only for small instances [23-27].

Metaheuristics consists of an abstraction layer which provides good solution for optimization problem. The technique reduces the search area so that only missing areas can be concentrated. Hence, the technique does not guarantee optimal solutions for all the poses.

Many metaheuristics algorithms have different characteristics (Blum and Roli, 2003), which provides several optimal solutions to the same problem. Among them, we highlight the following.

1. Distributed metaheuristics, searches for solutions within the entire solution space. These techniques work with populations or sets of elements which get combined to generate better solutions gradually. Some examples are ant colony and particle swarm optimization, genetic algorithms, scatter search, etc.
2. Neighborhood metaheuristics, searches for best elements in its neighborhood using the given solution space. Examples include guided local search, simulated annealing, etc.

2.3 Virtual Screening

VS search libraries consist of small molecules that have the potential to bind to a drug target. This creates a complex which has disease curing abilities. Molecular docking technique docks small molecules to the macromolecular targets. The goal here is to find out the optimal binding sites by ranking the chemical compounds according to the estimated bonding affinity.

The impact of VS has not yet been up to the mark. Both VS and SF are not still used efficiently to identify high affinity ligands reliably. VS methods should be very fast and reliable enough to identify numerous potential candidates. Hence, these techniques require thousands of CPU hours to complete the process. In VS techniques, we focus on protein-ligand docking. This technique uses MPI along with multithreading. Molecular docking for heterogeneous platform uses OpenCL for portability. The surface is usually derived from position of a ligand from complex. The problem with the existing techniques is that of fixed position of binding site. The binding site is fixed initially upon which all the ligands start interactions. Hence the other portion of the protein is completely discarded. In our new technique, the entire protein structure is divided into many regions [29]. These regions are given as input for docking among all the available GPU cores. Finally, the results are computed based on the best available optimal solution that exists among all the regions. Resulting new spot are considered and SF is applied all over the protein surface. Figure 1 shows the crystallographic structure of N-ethyl bound to HSP90 protein (Green Color).

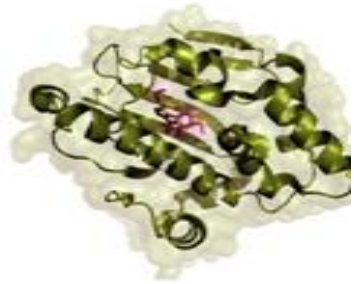

Figure 1: Docked Complex (N-ethyl-HSP90 Complex)

III. METAHEURISTICS FOR VIRTUAL SCREENING

Parallel executions are not that effective when they are executed in heterogeneous platforms. Adaptable super computers have shown this versatility since all the groups have same processing capabilities [28]. This area demonstrates our proposition, metaheuristic based virtual screening applications that influence enormously parallel and heterogeneous PCs. We acquaint user with the structure of our VS approach before demonstrating the system for heterogeneous circulation of the outstanding load on the system. The algorithm shown works on improving the hits based on the parameterized metaheuristic schema.

Initialize(S, ParamIni)
while no End condition(S) do
Select(S,Ssel, ParamSel)
Combine(Ssel, Scom,ParamCom)
Improve(Scom, ParamImp)
Include(Scom,S, ParamInc)
end while

Algorithm 1: Parameterized Metaheuristic Schema

3.1 Metaheuristics for VS Methods

The proposed VS system separates the entire protein surface into subjective and free locales (or spots). Spots are determined around carbons of the protein spine, with the goal that we can guarantee a full checking of the protein surface. Every one of these spots is autonomous of one another and, in this manner, offer extraordinary open doors for information based parallelization. As a matter of fact, the calculation spots duplicate the similar ligand at every one of those spots.

These duplicates (otherwise called people or adaptations) are different from one another as they have an alternate position and direction concerning each spot. Docking reproductions look for an enhanced compliance for protein and ligand and the relative direction between them, with the end goal that the free vitality (given by the scoring capacity) of the general framework is limited. In this way, our technique utilizes an enhancement procedure where the scoring capacity, that models the non-reinforced connections among protein and ligand, is limited all through the execution. In light of that, we initially present the advancement technique utilized in our technique before quickly depicting the GPU execution of the hidden scoring function. The scoring capacity calculation speaks to over 95% of our work in general computation time and consequently it is transferred to the GPU to build by and large application execution.

3.1.1 Search Method based on a Parameterized Metaheuristic Schema

Every one of the capacities in the calculation performs with different set populaces For our situation, a competitor arrangement is a con-development. Along these lines, a few people are chosen (Ssel) to be consolidated, so, creating another arrangement of components, Scom. Applicant arrangements can likewise be improved by applying a neighborhood search; for example, moving, deciphering as well as turning concerning each spot.

Advancing in creating bound together metaheuristics plans is the presentation of a few parameters, for example metaheuristic parameters, in every one of these capacities to give a more extensive scope of metaheuristics. Cutillas-Lozano et al. (2012) demonstrated that the utilization of a parameterized outline of meta-heuristics finds acceptable metaheuristics and to tune them for a specific issue. A few meta-heuristics could be assessed for the issue (each with its comparing tuning procedure), and cross breed metaheuristic plans can likewise be considered. As a result, the choice and tuning for an attractive metaheuristic or hybridization for an issue is a complex process, requires huge execution time.

This technique depends on that brought together metaheuristic composition and is utilized for simulations. As referenced in the 'metaheuristics' and as appeared in Algorithm 1, the composition resembles a template that characterizes a lot of capacities to be executed for a specific issue. Those capacities utilize a few parameters to give distinctive metaheuristic implementations. Table 2 shows employed parameters.

Table2:The Parameters used in the Parameterized Metaheuristic Schema

Metaheuristics parameters	Description
INEIni	Number of initial ligand conformations.
PEIIni	Best conformations that are improved in the function Initialize.
IIEIni	The intensification of the improvement in the function Initialize.
PBEIni	Best conformations to be included in the initial set for the next iterations.
PWEIni	Percentage of worst conformations to be included in the initial set for the next iterations.
PBESel	Best conformations to be selected for combination.
PWESel	Worst conformations to be selected for combination.
PBBCom	Best-best conformations to be combined.
PWWCom	Worst-worst conformations to be combined.

PBWCom	Best-worst conformations to be combined between them.

The composition is connected at every spot, with the equivalent metaheuristics parameters in Table 2 (the equivalent meta-heuristic). Fundamental capacities dealing with various subsets are also shown in Table 2. We summarize the results for various proteins below:

1. Initialize returns an underlying arrangement of arrangements. INEIni compliances are created arbitrarily for every one of the m spots. When they have been produced, for each (PEIIni) of the underlying adaptations of each spot is improved. The increase of the improvement is shown by the parameter IIEIni. At long last, (PBEIni + PWEIni) INEIni conformations from each spot are chosen for the execution of the accompanying capacities. PBEIni and PWEIni speak to the level of best and most exceedingly terrible conformations to be incorporated. The best adaptations are those with the best estimation of the scoring function, and the "most exceedingly terrible" compliances are chosen from the staying ones. To be sure, this approach does not choose simply the best compliances, in order to broaden the inquiry and abstain from falling into neighborhood optima.
2. Termination condition decides the exit criteria. Either, the greatest number of steps without progress of the best arrangement from every one of the spots, NIREnd, or the most extreme number of cycles, MNIEnd can be used to observe the results.
3. Select picks a few compliances to work with for the following stages. A level of the best and most noticeably awful compliances with respect to each spot is chosen, for example PBESel and PWESel.
4. Combine blends adaptations two by two, contingent upon their scoring. Most parameters speak to the level of best–best, most exceedingly terrible most exceedingly awful and best–most noticeably terrible compliances to be consolidated: PBBCom, PWWCom and PBWCom blends are per-framed among compliances at a similar spot.
5. Improvement is seen when nearby search is carried out by the recent protein fold surface. Each spot can be characterized by two parameters. These parameters show the ligand approach and also the binding factor involved

The parameter PBEInc sets up the level of best compliances related to each spot to be incorporated into its reference set. The remainder of the compliances to be incorporated into this set is randomly chosen from the rest of the adaptations at the spot. The incorporation of conformations adds to expand the hunt, so abstaining from slowing down in local area in minima.

3.1.2 GPU Implementation of the Scoring Function

SF depends on the pertinent non-fortified possibilities regularly utilized in VS figures recently portrayed in the 'Foundation' segment. They are Coulomb, electrostatic, the Lennard–Jones possibilities and the hydrogen-limits connections kernel. A discourse about the fundamental terms incorporated into the scoring capacity is past the extent of this paper. Algorithm 2 is used to calculate scoring using GPU.

```
pos = atom_position
individual = get_individual()
for i=1 to r do
  Energy = 4*epsilon*(term12(i,pos) - term6(i,pos))
  Scoring + = Energy
end for
synchronize_threads()
S_energy[individual] =
Reduction_atoms_individual()
```

Algorithm 2: Method to Calculate Scoring on GPU

SF is implemented in a kernel where all terms are determined simultaneously. We distinguish every competitor arrangement (for example adaptation) to a CUDA twist, and twists are assembled into squares relying upon the CUDA string square granularity. Some performance methodologies that we have connected to our codes to use NVIDIA GPU models are presented.

1. The utilization of shared memory encourages the re-usability of information by strings of a similar square. For our situation, the compound is stacked into the common memory so strings inside a similar square can share this data, so sparing expensive memory gets to. Along these lines, each string figures the scoring capacity comparing to the components that are related to every one of them, in this way expanding the general application data transmission [31-33].
2. The possibility of using shuffle instructions is available in devices with 3.X or higher compute capability, and their use can improve application performance substantially. These instructions enable information sharing within threads that belong to the same warp without using either shared or device memory.

The execution time of each autonomous execution can differ, since it depends on:

(a) The metaheuristics when executed takes a determined time and also it is unknown during the run time.
(b)The number of solutions is affected by the GPU heterogeneity and also the level of kernel detail.

3.2 Scaling to a Heterogeneous Node

The CPU threads can be managed by using OpenMP. Each thread controls a GPU instance. Each GPU is assigned with calculation of scoring function for a given set of candidates. These candidates are assigned in a homogenous way, where they are equally distributed GPUs in the CUDA thread model.

IV. EXPERIMENTAL RESULTS

We now present the results from our work on multicore and multi-GPU frameworks. The primary target of these trials is two-overlay. In the first place, we investigate our heap dispersion systems to improve execution on heterogeneous hubs dependent on CPU and multi-GPU.

Second, we study the nature of the outcomes with a few synthetic mixes to discuss the viability of our methodology.

4.1 Performance Results

Our method sets up the test set-up progressively, the outcomes appeared underneath are platform structures dependent. Thus, we give a comprehensive investigation on the two heterogeneous frameworks recently portrayed. Table 3 demonstrates the time efficiency (single-point accuracy execution) and relative accelerate factor among various usage and metaheuristics setups for target dataset in systems. They demonstrate the execution times for OpenMP execution on CPUs as a kind of perspective for the enhancements.

Table 3: Execution time (s) and speed-up for the metaheuristics in Jupiter

Metaheuristics	multicore CPU (s)	Speed-up GPU Tesla C2075 vs. multicore CPU	Speed-up homogeneous distribution vs. multicore CPU	Speed-up heterogeneous distribution vs. multicore CPU
M1	140.48	10.42	38.72	39.35
M2	193.67	13.81	39.88	43.55
M3	1911.52	9.62	53.25	54.16
M4	209.56	9.59	33.86	34.43
M5	262.65	8.55	34.81	35.51
M6	1379.93	10.39	53.61	53.89
DUD:SRC target				
M1	639.32	7.45	36.43	39.35
M2	678.41	7.86	37.89	40.97
M3	10,670.57	8.55	49.34	49.41
M4	1150.81	8.49	40.21	43.75
M5	1574.79	8.27	43.08	44.62
M6	7422.66	8.93	50.08	50.27
DUD:GPB target				
M1	910.42	9.24	45.19	46.21
M2	964.82	9.44	49.01	53.01
M3	15,050.81	10.75	60.98	62.28
M4	1654.85	10.94	52.88	57.58
M5	2449.27	11.47	58.71	62.47
M6	10,186.09	10.94	61.57	62.41

It is trivial to notice that the metaheuristics parameters are crucial for performance. This is clearly visible in metaheuristics M3 for compounds that are considerably small or perhaps range up to medium sizes. And also a considerable improvement can be seen in M6.

Table 3 shows performance numbers in Jupiter. The GPUs available in the aforementioned ecosystem is more or less homogeneous and hence the heterogeneous strategy doesn't put up a good performance in this ecosystem. It can also be noticed from Table 3 that the speed-up is directly proportional to problem size thus concluding that the multi-GPU versions facilitate more scalability. Figure 2 shows scoring function evolution.

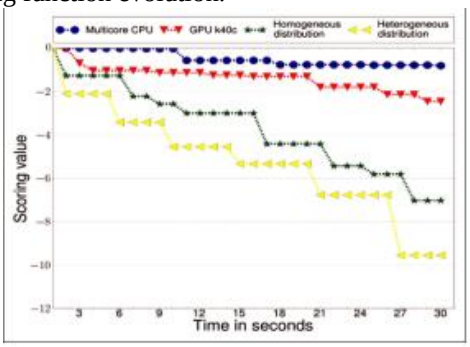

Figure 2: Scoring Function Evolution

Figure 2 shows the scoring function evaluation for about 30 seconds of docking. It is trivial to state here that the performance is directly proportional to the quality of results which only means the higher number of computation per time interval.

V. CONCLUSIONS AND FUTURE WORK

VS strategies are computational systems that guide or supplement the test tranquilize disclosure process yet they are all around computationally requesting applications. This paper presents a VS strategy, in light of a combined parameterized metaheuristics outline that can produce a wide assortment of metaheuristics, thus gives a completely adaptable edge work for medication revelation, and in this way encourages upgraded execution and expectation exactness. This is custom-made for heterogeneous PCs dependent on CPU and various GPUs. Even though the heterogeneity confines acceleration, using metaheuristics one can leverage the platform in a better way. It can be achieved in two ways:

1. CPU-GPU heterogeneity - Here some sections of computation are assigned to CPU and remaining assigned to GPU.

2. GPU-GPU heterogeneity - GPUs with different characteristics, the kernels part are again split among the available GPUs which are run concurrently and controlled in master slave architecture.

The results obtained by applying our technique indicates that using metaheuristics one can leverage the GPUs for optimal docking of protein-ligand. Also, it is evident that it is suitable where real time constraints need to be fulfilled along with the quality requirements. This technique serves as a primary aid in early stages of drug discovery.

This strategy is useful for such applications with stochastic behavior where real time constrains are to be met with accuracy. AUC results indicate that this technique s useful in drug discovery and also VS. Also from the results it is clear that metaheuristics technique improves overall efficiency in docking.

Our strategy is particularly useful for non-deterministic algorithms and stochastic behaviors, where real-time constraints must be fulfilled. Performance gains are translated into quality improvements that are a decisive factor in virtual screening. AUC results obtained with this technique support that its parallel, metaheuristics-based schema makes it a useful tool in the early stages of drug discovery.

REFERENCES

1. Almeida F, Gime´nez D, Lo´pez-Espı´n JJ, et al. (2013) Parame-terised schemes of metaheuristics: Basic ideas and applica-tions with genetic algorithms, scatter search and GRASP. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 43(3): 570–586.
2. Asanovic K, Bodik R, Catanzaro BC, et al. (2006) The land-scape of parallel computing research: A view from Berke-ley. Report, Report no. UCB/EECS-2006-183, University of California at Berkeley, USA, Electrical Engineering and Computer Sciences.
3. Austin T (2015) Bridging the Moore's law performance gap with innovation scaling. In: Proceedings of the 6th ACM/ SPEC international conference on performance engineering, Austin, Texas, 31 January–4 February, pp.1. ACM.
4. Bianchi L, Dorigo M, Gambardella LM, et al. (2009) A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing: An International Journal 8(2): 239–287.
5. Blum C, Puchinger J, Raidl GR, et al. (2011) Hybrid meta-heuristics in combinatorial optimization: A survey. Applied Soft Computing 11(6): 4135–4151.
6. Blum C and Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35(3): 268–308.
7. Cecilia JM, Garcı´a JM, Nisbet A, et al.(2013) Enhancing data parallelism for ant colony optimization on GPUs. Journal of Parallel and Distributed Computing 73(1): 42–51.
8. Chapman B, Jost G and Van Der Pas R (2008) Using OpenMP: Portable Shared Memory Parallel Programming, vol. 10. Massachusetts, United States: MIT press.
9. Cutillas-Lozano JM and Gime´nez D (2013) Determination of the kinetic constants of a chemical reaction in hetero-geneous phase using parameterized metaheuristics. In: 7th Workshop on Computational Chemistry and Its Appli-cations on International Conference on Computational Sci-ence (ICCS 2013), Barcelona, Spain, 5–7 June.
10. Cutillas-Lozano LG, Cutillas-Lozano JM and Gime´nez D (2012) Modeling shared-memory metaheuristic schemes for electricity consumption. In: Distributed computing and artificial intelligence – 9th international conference, Salamanca, Spain, 28–30 March, pp.33–40.
11. De Michell G and Gupta RK (1997) Hardware-software co-design. Proceedings of the IEEE 85(3): 349–365.
12. Dolezal R, Ramalho TC, Francxa TC, et al. (2015) Parallel flexible molecular docking in computational chemistry on high performance computing clusters. In: Computational Collective Intelligence, vol 9330. Berlin, Heidelberg: Springer, pp.418–427.
13. Dre´o J, Pe´trowski A, Siarry P, et al. (2005) Metaheuristics for Hard Optimization. New York, United States: Springer Science & Business Media.
14. Drews J (2000) Drug discovery: A historical perspective. Science 287(5460): 1960–1964.
15. DUD (2006) Directory of Useful Decoys. Available at: http:// dud.docking.org/ (accessed 4 October 2016).
16. Ewing TJA, Makino S, Skillman AG, et al. (2001) DOCK 4.0: Search strategies for automated molecular docking of flexible molecule databases. Journal of Computer-Aided Molecular Design 15(5): 411–428.
17. Franco AA (2013) Multiscale modelling and numerical simula-tion of rechargeable lithium ion batteries: Concepts, methods and challenges. RSC Advances 3(32): 13027–13058.
18. Friesner RA, Banks JL, Murphy RB, et al. (2004) Glide: A new approach for rapid, accurate docking and scoring: Method and assessment of docking accuracy. Journal of Medicinal Chemistry 47(7): 1739–1749.
19. Glover F and Kochenberger GA (2003) Handbook of Meta-heuristics. New York, United States: Kluwer Academic Publishers.
20. Guerrero GD, Cebria´n JM, Pe´rez-Sa´nchez H, et al. (2014) Toward energy efficiency in heterogeneous processors: Findings on virtual screening methods. Concurrency and Computation: Practice and Experience 26(10): 1832–1846.
21. Hromkovic˘J (2003) Algorithmics for Hard Problems. 2nd ed.Berlin: Springer.
22. Huang SY and Zou X (2010) Advances and challenges in protein-ligand docking. International Journal of Molecular Sciences 11(8): 3016–3034.
23. Irwin JJ and Shoichet BK (2005) ZINC–a free database of commercially available compounds for virtual screening. Journal of Chemical Information and Modeling 45(1): 177–182.
24. Jain AN (2006) Scoring functions for protein-ligand docking.
25. Current Protein and Peptide Science 7(5): 407–420.
26. Jorgensen WL (2004) The many roles of computation in drug discovery. Science 303: 1813–1818.
27. Kirk DB and Wen-Mei WH (2013) Programming Massively Parallel Processors: A Hands-On Approach. Boston, MA, USA: Morgan Kaufmann Publishers Inc.
28. Kitchen DB, Decornez H, Furr JR, et al. (2004) Docking and scoring in virtual screening for drug discovery: Methods and applications. Nature Reviews Drug Discovery 3(11): 935–949.
29. Kuntz SK, Murphy RC, Niemier MT, et al. (2001) Petaflop computing for protein folding. In: Proceedings of the tenth SIAM conference on parallel processing for scientific computing, Porstmouths, Virginia, USA, 12–14 March, pp. 12–14.
30. Li Y, Han L, Liu Z, et al. (2014a) Comparative assessment of scoring functions on an updated benchmark: 2. evaluation methods and general results. Journal of chemical informa-tion and modeling 54(6): 1717–1736.
31. Li Y, Liu Z, Li J, et al. (2014b) Comparative assessment of scoring functions on an updated benchmark: 1. Compilation of the test set. Journal of Chemical Information and Modeling 54(6): 1700–1716.
32. Lionta E, Spyrou G K, Vassilatis D, et al. (2014) Structure-based virtual screening for drug discovery: Principles, applications and recent advances. Current Topics in Medic-inal Chemistry 14(16): 1923–1938.
33. Lo´pez-Camacho E, Garcı´a-Godoy MJ, Garcı´a-Nieto J, et al. (2015) Solving molecular flexible docking problems with metaheuristics: A comparative study. Applied Soft Com-puting 28: 379–393.
34. McIntosh-Smith S, Price J, Sessions RB, et al. (2014) High performance in silico virtual drug screening on many-core processors. International Journal of High Performance Computing Applications 29(2): 119–134.
35. Michalewicz Z and Fogel DB (2002) How to Solve It: Modern Heuristics. Berlin: Springer.