Applications of High Performance Computing in Bioinformatics, Computational Biology and Computational Chemistry

Horacio Péréz-Sánchez[1], Afshin Fassihi[1], José M. Cecilia[1], Hesham H. Ali[2], and Mario Cannataro[3]

[1] Bioinformatics and High Performance Computing Research Group (BIO-HPC)
Computer Science Department
Universidad Católica San Antonio de Murcia (UCAM), Guadalupe E30107, Spain
{hperez,afassihi,jmcecilia}@ucam.edu
[2] Department of Computer Science
College of Information Science and Technology
University of Nebraska at Omaha, USA
hali@unomaha.edu
[3] Bioinformatics Laboratory
Department of Medical and Surgical Sciences
University Magna Graecia of Catanzaro, Italy
cannataro@unicz.it

Abstract. In the last 10 years, we are witnessing one of the major revolutions in parallel systems. The consolidation of heterogeneous systems at different levels -from desktop computers to large-scale systems such as supercomputers, clusters or grids, through all kinds of low-power devices- is providing a computational power unimaginable just few years ago, trying to follow the wake of Moore's law. This landscape in the high performance computing arena opens up great opportunities in the simulation of relevant biological systems and for applications in Bioinformatics, Computational Biology and Computational Chemistry. This introductory article shows the last tendencies of this active research field and our perspectives for the forthcoming years.

Keywords: HPC, Parallel Computing, Heterogeneous Computing, Bioinformatics, Computational Biology, Computational Chemistry.

1 Introduction

The integration of the latest breakthroughs in biochemistry and biotechnology from one side and high performance computing and computational modelling from the other, enables remarkable advances in the fields of healthcare, drug discovery, genome research, systems biology and so on. By integrating all these developments together, scientists are creating new exciting personal therapeutic strategies for living longer and having healthier lifestyles that were unfeasible not that long ago.

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 527–541, 2015.

Those efforts have created new research fields such as *Bioinformatics and Computational Biology*, defined most broadly as informatics in the domains of biology and biomedical research. Bioinformatics spans to many different research areas, such as life sciences, where there are many examples of scientific applications for discovering biological and medical unknown factors that could greatly benefit from increased computational resources.

Indeed, as computing resources available on current systems are limited, this limitation becomes a serious constraint, then hindering it from successfully taking the next step forward. For instance, applications such programs of Molecular Dynamics (MD) [1], employed to analyse the dynamical properties of macromolecules such as folding and allosteric regulations, or software used for solving atom-to-atom interactions for drug discovery, such as AutoDock [2] and FlexScreen [3], could clearly benefit from enhanced computing capabilities and also from some novel algorithms approaches like those inspired by nature such as Genetic algorithms [4] or Ant Colony Optimization techniques [5].

There are other applications, which are actually working well, but their execution is too slow for providing feedback in real-time to the users, and thus limiting the effectiveness and comfort of such applications. In this group, we may cite biomedical image processing applications, such as X-ray computed tomography or mammography for breast cancer detection. The low-performance of these applications can drastically affect the patient's health. For instance, imagine a patient who is waiting for mammography results. She is actually waiting for a breast cancer diagnostic, thus the acceleration of the diagnosis time becomes paramount for the patient's health.

High Performance Computing technologies are at the forefront of those revolutions, making it possible to carry out and accelerate radical biological and medical breakthroughs that would directly translate into real benefits for the society and the environment. In this regard, Graphics Processing Units (GPUs) are providing a unique opportunity to tremendously increase the effective computational capability of the commodity PCs, allowing desktop supercomputing at very low prices. Moreover, large clusters are adopting the use of these relatively inexpensive and powerful devices as a way of accelerating parts of the applications they are running. Since June 2011, when the fastest supercomputer was the Tianhe-1A, placed in the National Supercomputing Centre at Tianjin (China)[1], including up to 7.168 NVIDIA® Tesla™ M2050 general purpose GPUs, several supercomputers have followed this trend.

However, the inclusion of those accelerators in the system has a great impact on the power consumption of the system, as a high-end GPU may increase the power consumption of a cluster node up to 30% which is actually a big issue. This is a critical concern especially for very large data centres, where the cost dedicated to supply power to such computers represents an important fraction of the Total Cost of Ownership (TCO) [6]. The research community is also aware of this and it is making efforts in striving to develop reduced-power installations.

[1] `http://www.top500.org/`

Thus, the GREEN500 list[2] shows the 500 most power efficient computers in the world. In this way, we can see a clear shift from the traditional metric FLOPS (FLoating point Operations Per Second) to FLOPS per watt.

Virtualization techniques may provide significant energy savings, as they enable a larger resource usage by sharing a given hardware among several users, thus reducing the required amount of instances of that particular device. As a result, virtualization is being increasingly adopted in data centres. In particular, cloud computing is an inherently energy-efficient virtualization technique [7], in which services run remotely in a ubiquitous computing cloud that provides scalable and virtualized resources. Thus peak loads can be moved to other parts of the cloud and the aggregation of a cloud's resources can provide higher hardware utilization [8]. Public cloud providers offer their services in a *pay as you go* fashion, and provide an alternative to physical infrastructures. However, this alternative only becomes real for a specific amount of data and target execution time.

The rest of the paper is organized as follows. We briefly introduce some HPC architectures we think are at the forefront of this revolution in Section 2. In Section 3 we present some relevant Bioinformatics applications that are using HPC alternatives to deal with their computational issues before we summarize our findings and conclude with suggestions for future work.

2 HPC Architectures

Traditionally, high performance computing has been employed for addressing bioinformatics problems that would otherwise be impossible to solve. A first example was the preliminary assembly of the human genome, a huge effort in which the Human Genome Project was challenged by Celera Genomics with a different approach, consisting in a bioinformatics post analysis of whole sets of shotgun sequencing runs instead of the long-standing vector cloning technique, arriving very close to an unpredictable victory [9].

In this Section, we briefly summarize the high performance systems that has been commonly used in Bioinformatics. Among them, we highlight throughput-oriented architectures such as GPUs, large-scale heterogeneous clusters and clouds or distributed computing systems, with a discussion about how the latest breakthroughs in these architectures are being used in this field.

2.1 GPU Computing

Driven by the demand of the game industry, graphics processing units (GPUs) have completed a steady transition from mainframes to workstations to PC cards, where they emerge nowadays like a solid and compelling alternative to traditional computing platforms. GPUs deliver extremely high floating point performance and massively parallelism at a very low cost, thus promoting a new

[2] `http://www.green500.org/`

concept of the high performance computing (HPC) market; i.e. *heterogeneous computing* where processors with different characteristics work together to enhance the application performance taking care of the power budget. This fact has attracted many researchers and encouraged the use of GPUs in a broader range of applications, particularly in the field of Bioinformatics, where developers are required to leverage this new landscape of computation with new programming models which ease the developers task of writing programs to run efficiently on such platforms altogether [10].

The most popular microprocessor companies such as NVIDIA, ATI/AMD or Intel, have developed hardware products aimed specifically at the heterogeneous or massively parallel computing market: Tesla products are from NVIDIA, Firestream is AMDs product line and Intel Xeon Phi comes from Intel. They have also released software components, which provide simpler access to this computing power. CUDA (Compute Unified Device Architecture) is NVIDIAs solution as a simple block-based API for programming; AMDs alternative was called Stream Computing and Intel relies on X86-based programming. More recently (in 2008), the OpenCL[3] emerged as an attempt to unify all of those models with a superset of features, being the best broadly supported multi-platform data-parallel programming interface for heterogeneous computing, including GPUs, accelerators and similar devices.

Although these efforts in developing programming models have made great contributions to leverage the capabilities of these platforms, developers have to deal with a massively parallel and high throughput-oriented architecture[11], which is quite different than traditional computing architectures. Moreover, GPUs are being connected with CPUs through PCI Express bus to build heterogeneous parallel computers, presenting multiple independent memory spaces, a wide spectrum of high speed processing functions, and communication latency between them. These issues drastically increase scaling to a GPU-cluster, bringing additional sources of latency. Therefore, programmability on these platforms is still a challenge, and thus many research efforts have provided abstraction layers avoiding to deal with the hardware particularities of these accelerators and also extracting transparently high level of performance, providing portability across operating systems, host CPUs and accelerators. For example, libraries interfaces for programming with popular programming languages like OMPSs for OpenMP [4] or OpenACC[5] API, which describes a collection of compiler directives to specify loops and regions of code in standard programming language such as C, C++ or Fortran.

3 Applications

This section describes some bioinformatics applications from the High Performance computing point of view.

[3] `http://www.khronos.org/opencl/`
[4] `http://openmp.org/wp/`
[5] `http://www.openacc-standard.org/`

3.1 Virtual Screening

In this Section, we summarize the main technical contributions for the parallelization of Virtual Screening (VS) methods on GPUs available on the bibliography. Concretely, we pay special attention to the parallelization of docking methods on GPUs.

In terms of implementations, the trend seems to be reusing available libraries when possible and implement the achievements into existing simulation packages for VS. Among the most-used strategies are either implementing the most time-consuming parts of previously designed codes for serial computers, or redesigning the whole code from scratch. When porting VS methods to GPUs, we should realize that not all methods are equally amenable for optimization. Programmers should check carefully how the code works and whether it is suited for the target architecture. Irrespective of CUDA, most authors maintain that the application will be more accessible in the future thanks to new and promising programming paradigms which are still in the experimental stage or are not yet broadly used. Among them, we may highlight OpenCL or DirectCompute.

Dock6.2 In the work of Yang et al. [12] a GPU accelerated amber score in Dock6.2 is presented. They report up to 6.5x speedup factor with respect to 3,000 cycles during MD simulation compared to a dual core CPU. [6].

The lack of the single-precision floating point operations in the targeted GPU (NVIDIA GeForce 9800GT) produces small precision losses compared to the CPU, which the authors assume as acceptable. They highlight the thread management utilizing multiple blocks and single transferring of the molecule grids as the main factor that dominates the performance improvements on GPU.

They use another optimization technique, such as dealing with the latency attributed to thread synchronization, divergence hidden and shared memory through tiling, that authors state may double the speedup of the simulation. We miss a deeper analysis on the device memory bandwidth utilization. It is not clear whether the pattern accesses to device memory in the different versions of the designs presented here are coalesced or not, which may drastically affect the overall performance.

They finally conclude that the speedup of Amber scoring is limited by the Amdahl's law for two main reasons: (1) the rest of the Amber scoring takes a higher percentage of the run time than the portion parallelized on the GPU, and (2) partitioning the work among SMs will eventually decrease the individual job size to a point where the overhead of initializing an SP dominates the application execution time. However, we do not see any clear evaluation that supports these conclusions.

Autodock. In the paper of Kannan et al. [13] the migration to NVIDIA GPUs of part of the molecular docking application *Autodock* is presented. Concretely,

[6] `http://dock.compbio.ucsf.edu/DOCK_6/`

they only focus on the Genetic Algorithm (GA) which is used to find the optimal docking conformation of a ligand with respect to a protein. They use single-precision floating point operation arguing that, "GA depends on relative goodness among individual energies and single precision may not affect the accuracy of GA path significantly". All the data relative to the GA state is maintained on the GPU memory, avoiding data movement through the PCI Express bus.

The GA algorithms need random numbers for the selection process. They decide to generate the random numbers on the CPU instead of doing it on the GPU. The explanation of that is two-fold according to the authors: (1) it enables one-to-one comparisons of CPU and GPU results, and (2) it reduces the design, coding and validation effort of generating random numbers on GPU.

A very nice decision is what the authors call *CGPU Memory Manager* that enables alignment for individual memory request, support for pinned memory and join memory transfer to do all of them in just one transfer. Regarding the fitness function of the GA, authors decide to evaluate all the individuals in a population regardless of modifications. This avoids warp divergences although it makes some redundant work.

Three different parallel design alternatives are discussed in this regard. Two of them only differ in the way they calculate the fitness function, assigning the calculation of the fitness of an individual either to a GPU thread or GPU block. A good comparison between them is provided. The last one includes an extra management of the memory to avoid atomic operations which drastically penalizes the performance.

All of these implementations are rewarded with up to 50x in the fitness calculation, but they do not mention anything about global speedup of the Autodock program.

Genetic Algorithms Based Docking. In literature is also available [14] an enhanced version of the PLANTS [15] approach for protein-ligand docking using GPUs. They report speedup factors of up to 50x in their GPU implementation compared to an optimized CPU based implementation for the evaluation of interaction potentials in the context of rigid protein. The GPU implementation was carried out using OpenGL to access the GPU's pipeline and Nvidia's Cg language for implementing the shaders programs (i.e. Cg kernels to compute on the GPU). Using this way of programming GPUs, the developing effort is too high, and also some peculiarities of the GPU architecture may be limited. For instance, the authors say that some of the spatial data structures used in the CPU implementation can not directly be mapped to the GPU programming model because of missing support for shared memory operations [14].

The speedup factors observed, especially for small ligands, are limited by several factors. First, only the generation of the ligand-protein conformation and the scoring function evaluation are carried out on the GPU, whereas the optimization algorithm is run on the CPU. This algorithmic decomposition implies time-consuming data transfers through PCI Express bus. The optimization algorithm used in PLANTS is the Ant Colony Optimization (ACO) algorithm [16].

Concretely, authors propose a parallel scheme for this algorithm on a CPU cluster, which use multiple ant colonies in parallel, exchanging information occasionally between them [17]. Developing the ACO algorithm on the GPU as it has been shown in [18] can drastically reduce the communications overhead between CPU and GPU.

3.2 Parallel Processing of Microarray Data

High throughput experimental platforms, such as mass spectrometry, microarray, and next generation sequencing, are producing an overwhelming amount of data yielding to the so called omics world. Among the many omics disciplines, genomics, proteomics, and interactomics and are gaining an increasing interest in the scientific community for the study of diseaases at the molecular level.

The increasing availability of omics data poses new challenges for efficient data storage and integration as well as data preprocessing and analysis: managing omics data requires both space for data storing as well as procedures for data preprocessing and analysis. Main challenges are: (i) the efficient storage, retrieval and integration of experimental data; (ii) their efficient and high-throughput preprocessing and analysis; (iii) the building of reproducible "in silico" experiments; (iv) the annotation of omics data with pre-existing knowledge (e.g. extracted from ontologies like Gene Ontology, or from specialized databases); (v) the integration of omics and clinical data.

Well-known high performance computing techniques, such as Parallel and Grid Computing, and emerging computational models such as Graphics Processing and Cloud Computing, are more and more used in bioinformatics and life sciences [19], with the main aim to face the complexity of bioinformatics algorithms and to allow the efficient analysis of huge data.

Affymetrix Power Tools. Affymetrix Power Tools[7] are a set of command line programs provided by Affymetrix that implement different algorithms for preprocessing Affymetrix arrays. In particular **apt-probeset-summarize** implements summarization and normalization methods (e.g. RMA, RMA-SKETCH and PLIER) for gene expression arrays, while **apt-dmet-genotype** supports probe-set summarization of binary CEL files and the management of resulting preprocessed files (.CHP) related to genotyping arrays, i.e. microarray detecting single nucleotide polymorphisms (SNPs) and copy number viariations (CNVs). The system developed in [20] uses a master/slave approach, where the master node computes partitions of the input dataset (i.e. sets a list of probesets intervals) and calls in parallel several slaves each one wrapping and executing the apt-probeset-summarize program, that is applied to the proper partition of data. Such system showed a nearly linear speedup up to 20 slaves.

[7] `www.affymetrix.com`

DMET Console. DMET Console[8] is a graphical software provided by Affymetrix that supports probe-set summarization of a complete dataset of binary .CEL files, the management of resulting preprocessed files (.CHP), and the building of a tabular dataset containing the genotype call for all the probesets and all the samples of an experiment [21,22].

DMET-Analyzer. DMET-Analyzer [23] is a platform-independent software built in Java that supports the automatic statistics test of the association between SNPs and sample conditions. It has a simple graphical user interface that allows users to upload DMET files produced by DMET Console and produces as output a list of candidate SNPs. DMET-Analyzer supports the visualization of the SNPs detected on the entire dataset as a heatmap to give an immediate visual feedback to the user. It implements a Hardy-Weinberg equilibrium calculator that can be used for testing the genetic model. Finally, it annotates significant SNPs with information provided by Affymetrix libraries and with links to the dbSNP database (for basic information about SNPs) and to the PharmaGKB pharmacogenomics knowledge base, giving various information (e.g. pathways) related to pharmacogemomics.

coreSNP. coreSNP is a parallel software tool [24] for the parallel preprocessing and statistical analysis of DMET data that extends the DMET-Analyzer sequential algorithm presented in [23]. It allows to statistically test, in a parallel and automatic way and for each probe, the significance of the presence of SNPs in two classes of samples using the well known Fisher test. It automatizes the workflow of analysis of SNPs data avoiding the use of multiple tools and optimizes the execution of statistical tests on large datasets. Efficiency in executing statistical tests is obtained by parallelizing the core algorithm of DMET-Analyzer on multicore architectures and by defining efficient ad hoc data structures. The scalable multi-threaded implementation of coreSNP allows to handle the huge volumes of experimental pharmacogenomics data in a very efficient way, while its easy to use graphical user interface and its ability to annotate significant SNPs, allow biologists to interpret the results easily.

Cloud BioLinux. Cloud BioLinux[9] is a publicly accessible Virtual Machine that provides high-performance bioinformatics computing on different Cloud platforms [25]. It provides both pre-configured command line applications and graphical-enabled applications. Users may access Cloud BioLinux applications on the Amazon EC2 cloud, or they can upload Cloud BioLinux images to private Eucalyptus clouds. Moreover, users can run Cloud BioLinux on a local desktop

[8] The Affymetrix DMET (drug metabolism enzymes and transporters) Plus Premier Pack is a novel microarray platform for gene profiling, designed specifically to detect in human samples the presence/absence of SNPs on 225 genes that are related with drug absorption, distribution, metabolism and excretion (ADME)

[9] `cloudbiolinux.org`

computer using VirtualBox, a virtualization software available for main operating systems.

3.3 Big Data Analytics and Network Models in Biomedical Informatics

The increasing availability of big biological and medical data continues to present major challenges as well as great opportunities in biomedical research. Such challenges and opportunities are only expected to continue to grow. New technologies, such as next-generation sequencing, and new health-related policies, such as the ones associated with electronic medical records; promise to produce even more data in the near future. Such explosion of biomedical data requires an associated increase in the scale and sophistication of the automated systems and intelligent tools needed to enable the researchers to take full advantage of available data. This comes during a time when the notion of big data analytics is emerging as a significant research area in computing and information technology to address the problem of mining useful knowledge from raw data in various application domains, with big data defined loosely as data that is high in volume, velocity, variety, and veracity. One of the main research questions in biomedical sciences has been how to extract useful knowledge from such massive raw data. The development of innovative tools to integrate, analyze and mine such data sources is a key step towards achieving large impact levels. In addition, advanced tools for visualizing biomedical data at various processing stages are critical in maximizing the value of its utilization. As a result, attention has been shifting recently from a pure focus on data generation technologies to a more balanced approach significant emphasize on data analysis and data visualization tools. Such tools are critical in taking full advantage of the public and private data currently available to all biomedical researchers. In the realm of biomedical informatics and systems biology, there is a growing need for innovative multidisciplinary research that can handle the continuously growing big data while retaining the capability to analyze and predict anomalies in the system, be the system on the cellular, organismal, or social level. The use of advanced network modeling and analysis has shown to be particularly effective in integrating different types of data elements in the biomedical domain and explore the interrelationships among such elements. In addition, due to the large size of most biological and medical data, the need to utilize high performance computing systems is also emerging as an essential component of how to efficiently analyze raw data in biomedical research. Hence, the proper integration of carefully selected/developed algorithms along with efficient utilization of high performance computing systems form the key ingredients in the process of reaching new discoveries from biological data.

Network Models in Biomedical Informatics. Since the explosive influx of biological data obtained from high-throughput medical instruments, the ability to leverage the currently available data to extract useful knowledge has become

one of the most challenging and exciting aspects in biomedical research. The analysis of such data is particularly complex not only due to its massive size but also due to its heterogeneity and inherent noise associated with several data gathering steps. The utilization of biological networks to model and integrate large-scale heterogeneous biomedical data continues to grow, especially with the systems biology approach taking center stage in many bioinformatics applications. Networks have been used to model various types of relations in Bioinformatics including gene interactions, protein-protein interactions, gene-disease relationship, and correlations among gene expressions. Networks constructed from high-throughput biological data are rising in popularity and importance in systems biology for their ability to effectively model relationships between entities. In addition, since networks and essentially are graphs, the ability to take full advantage of existing graph theoretic properties and algorithms makes network modeling a reliable approach for examining large complex systems. More importantly, critical elements and structures obtained from networks that model biological data have been shown to correspond to critical biological elements and cellular functions associated with the domain from which the data is obtained. Network structure has been tied to cellular function since the discovery of the link between high degree nodes and essential proteins in the interactome of yeast [26]. Initial studies performed on protein-protein interaction networks indicated that these networks follow the rather useful power-law degree distribution, meaning that most nodes in the network are poorly connected with a few nodes are very well connected; such nodes are known informally as hubs [27] [26]. Hubs have been found in the yeast protein-protein interaction network (also known as an interactome) to correspond to essential genes and have been found to be critical for maintenance of structure in other biological networks as well, such as the metabolome and the correlation network [26].

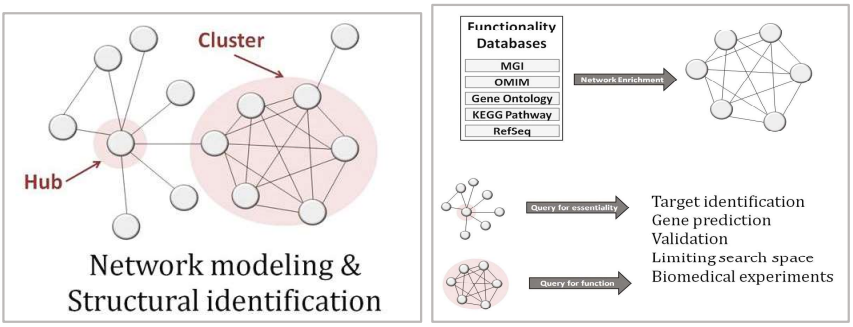


Fig. 1.

Other standard network/graph parameters such as clustering coefficient can point toward the modularity of the network4, and previous studies to identify modules in clustered networks indicate that when found, tend to correspond to genes or gene products working together toward some discrete function, such as

a protein complex in an interactome or as a regulatory cohort [27]. Many algorithms currently exist that are able to find clusters within networks that employ clustering via random seed selection and growing, spectral clustering, or clustering coefficient. It is worth nothing that while gene clusters tend to correspond to biological functions, the actual structures they form in the network can be mined based solely on network structure, often without the help of biological annotation data. Thus, the link between network structure and function can be exploited to identify known and unknown network elements.

Gene Expressions and Correlation Networks. Gene expression can be modeled as a network where nodes represent gene probes and edges represent relationships between expression arrays per each gene probe. Many studies have examined these so-called correlation or co-expression networks, particularly how they are built and validated in silico. In 2005, Horvath et al. proposed a method for creating correlation networks using a weighted gene expression schema, identifying a soft-thresholding approach that resulted in a scale-free network distribution [28]. Extensive research performed by Dempsey and Ali also found links between structure and function in networks built from correlation [29] [26]. In other words, there is a growing body of research that indicates the correlation network is a valid method for identifying co-expression and potentially co-regulation from high-throughput gene expression data. Correlation networks can be effectively used to model complex biological data in which nodes represent genes or gene products and edges connecting the nodes represent degrees of correlation associated with their expression levels. Although loaded with biologically-relevant signals, correlation networks do contain noise plus they are too large for simple data mining tools. In this project, we implement different types of filters to reduce the network size and sort out signals from noise. The use of gene correlation networks has emerged to assist in the discovery of previously unknown genetic relationships and the identification of significant biological functions. Such networks provide a useful mechanism to model experimental results obtained from expression data and capture a snapshot of the expression as well as the temporal changes in various experiments. In addition, gene Ontology is often integrated with biological networks within the analysis process as a source of domain knowledge. The application of correlation networks to characterize biological data has been and remains a unique method for determining changes in biological relationships over time. The correlation network is an all vs. all graphical model that examines the degree of relation over pairs of data points, such as genes in microarray expression data. Certain characteristics of the network, such as density, hubs, cliques, and pathways, can be used to filter noise and to identify causative sub-networks of biologically relevant genes or complexes. More recently correlation networks are being used to model biological relationships over time, for example in the progression of disease, the effect of pharmaceuticals on systems of the body, and in the process of aging [30].

Biological Networks and High Performance Computing. Although extremely effective, modeling big biomedical data using networks can be computationally extensive. To start with the number of nodes in such networks could be very large, however, the main computational intensity results from the large size of the solution space. For a network of size n, the number of possible relationships is of order $O(n^2)$, while the search space for the clusters or dense subgraphs is bounded only by the $O(2^n)$ exponential function. While heuristics are naturally used to conduct the clustering analysis, dealing with large networks remain expensive computationally, hence the need for high performance computing. The figure below show the impact of using up to 1000 processors to create and analyze a correlation network use to model brain tissues in aging mice. Due to the high degree of parallelism associate with most graph operation, significant speedup can be obtain by increasing the number of processors; in this case from 2 weeks to few minutes.

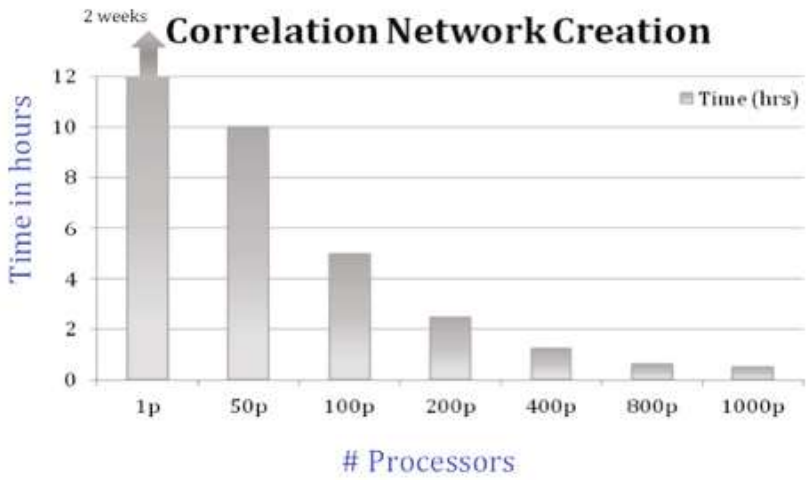


Fig. 2.

The need for high performance computing approaches to effectively solve critical problems in Bioinformatics present an exciting set of opportunities for computational researchers to get involved in Bioinformatics research. The next generation of Bioinformatics tools will be expected to take advantage of parallel computing techniques, the availability of computational resources via the clouds, and many ideas developed in the high performance computing domain to address big data analytics challenges in biomedical research.

4 Conclusions and Outlook

Applications with a real impact on the society, such as those in the field of Bioinformatics and Computational Biology, can take advantage from improvement in

high performance computing to overcome computational limitations they have by definition. Those applications are developed to give the opportunity to create new exciting personal therapeutic strategies for living longer and having healthier lifestyles that were unfeasible not that long ago.

This work summarizes the main trends in HPC applied to Bioinformatics. We show several successful stories and application fields, in which relevant biological problems have been solved (or are being targeted) thanks to the computational power available in current processors. We have also pointed out the main drawbacks in the HPC arena, which may limit this good alliance between Bioinformatics and HPC systems. Among them we may highlight power consumption, high learning-curve in emerging programming models to leverage their computational power and the total cost of ownership. We think that the investigations on improvement Bioinformatics application's performance on HPC systems will be also of high technological interest as those applications have novel computational patterns that can lead the next generation of heterogeneous computing systems.

Acknowledgements. This work was supported by the Fundación Séneca la Región de Murcia under Project 18946/JLI/13. This work has been funded by the Nils Coordinated Mobility under grant 012-ABEL-CM-2014A, in part financed by the European Regional Development Fund (ERDF). This work was partially supported by the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA−CIEMAT), funded by the European Regional Development Fund (ERDF). CETA−CIEMAT belongs to CIEMAT and the Government of Spain. The authors also thankfully acknowledge the computer resources and the technical support provided by the Plataforma Andaluza de Bioinformática of the University of Málaga. We also thank NVIDIA for hardware donation under CUDA Teaching Center 2014. This work has been partially supported by the projects PRIN 2010-2011 2010NFEB9L_003 and PON04a2_D DICET-INMOTO-ORCHESTRA funded by MIUR.

References

1. Klepeis, J.L., Lindorff-Larsen, K., Dror, R.O., Shaw, D.E.: Long-timescale molecular dynamics simulations of protein structure and function. Current Opinion in Structural Biology 19(2), 120–127 (2009)
2. Morris, G.M., Goodsell, D.S., Huey, R., Olson, A.J.: Distributed automated docking of flexible ligands to proteins: Parallel applications of AutoDock 2.4. Journal of Computer-Aided Molecular Design 10(4), 293–304 (1996)
3. Fischer, B., Merlitz, H., Wenzel, W.: Increasing Diversity in In-Silico Screening with Target Flexibility. In: Proceedings of Computational Life Sciences, First International Symposium, CompLife. CompLife 2005, pp. 186–197. Springer (2005)
4. Halperin, I., Ma, B., Wolfson, H., Nussinov, R.: Principles of docking: An overview of search algorithms and a guide to scoring functions. Proteins 47(4), 409–443 (2002)

5. Korb, O., Stützle, T., Exner, T.E.: Accelerating Molecular Docking Calculations Using Graphics Processing Units. Journal of Chemical Information and Modeling 51, 865–876 (2011)
6. Fan, X., Weber, W.-D., Barroso, L.A.: Power provisioning for a Warehouse-Sized Computer. In: Proceedings of the 34th annual International Symposium on Computer Architecture, ISCA 07, pp. 13–23. ACM (2007)
7. Hewitt, C.: ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing. IEEE Internet Computing 12(5), 96–99 (2008)
8. Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M.Q., Pentikousis, K.: Energy-efficient cloud computing. The Computer Journal 53(7), 1045–1051 (2010)
9. Venter, J.C., Adams, M.D., Myers, E.W., Li, P.W., Mural, R.J., Sutton, G.G., Smith, H.O., Yandell, M., Evans, C.A., Holt, R.A.: The Sequence of the Human Genome. Science 291(5507), 1304–1351 (2001), http://dx.doi.org/10.1126/science.1058040
10. Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V.: Parallel Computing Experiences with CUDA. IEEE Micro 28, 13–27 (2008)
11. Garland, M., Kirk, D.B.: Understanding throughput-oriented Architectures. Communications of the ACM 53, 58–66 (2010)
12. Yang, H., Zhou, Q., Li, B., Wang, Y., Luan, Z., Qian, D., Li, H.: GPU Acceleration of Dock6's Amber Scoring Computation. Advances in Computational Biology 680, 497–511 (2010)
13. Kannan, S., Ganji, R.: Porting Autodock to CUDA. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2010)
14. Korb, O., Stützle, T., Exner, T.E.: Accelerating molecular docking calculations using graphics processing units. Journal of Chemical Information and Modeling 51(4), 865–876 (2011)
15. Korb, O., Stützle, T., Exner, T.E.: PLANTS: Application of Ant Colony Optimization to Structure-Based Drug Design. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 247–258. Springer, Heidelberg (2006)
16. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. dissertation, Politecnico di Milano, Italy (1992)
17. Manfrin, M., Birattari, M., Stützle, T., Dorigo, M.: Parallel ant colony optimization for the traveling salesman problem. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 224–234. Springer, Heidelberg (2006)
18. Cecilia, J.M., García, J.M., Ujaldón, M., Nisbet, A., Amos, M.: Parallelization strategies for ant colony optimisation on gpus. In: NIDISC 2011: 14th International Workshop on Nature Inspired Distributed Computing. Proc. 25th International Parallel and Distributed Processing Symposium (IPDPS 2011), Anchorage, Alaska, USA (May 2011)
19. Cannataro, M.: Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare. IGI Global Press, Hershey (2009)
20. Guzzi, P.H., Cannataro, M.: Parallel pre-processing of affymetrix microarray data. In: Guarracino, M.R., Vivien, F., Träff, J.L., Cannatoro, M., Danelutto, M., Hast, A., Perla, F., Knüpfer, A., Di Martino, B., Alexander, M. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 225–232. Springer, Heidelberg (2011)

21. Sissung, T.M., English, B.C., Venzon, D., Figg, W.D., Deeken, J.F.: Clinical pharmacology and pharmacogenetics in a genomics era: the DMET platform. Pharmacogenomics 11(1), 89–103 (2010), http://dx.doi.org/10.2217/pgs.09.154
22. Burmester, J.K., Sedova, M., Shapero, M.H., Mansfield, E.: Dmet microarray technology for pharmacogenomics-based personalized medicine. Microarray Methods for Drug Discovery, Methods in Molecular Biology 632, 99–124 (2010)
23. Guzzi, P.H., Agapito, G., Di Martino, M.T., Arbitrio, M., Tagliaferrri, P., Tassone, P., Cannataro, M.: DMET-analyzer: Automatic analysis of affymetrix DMET data. BMC Bioinformatics 13, 258 (2012), http://dx.doi.org/10.1186/1471-2105-13-258
24. Guzzi, P.H., Agapito, G., Cannataro, M.: coresnp: Parallel processing of microarray data. IEEE Transaction on Computers 63(12), 2961–2974 (2014)
25. Krampis, K., Booth, T., Chapman, B., Tiwari, B., Bicak, M., et al.: Cloud biolinux: Pre-configured and on-demand bioinformatics computing for the genomics community. BMC Bioinformatics 13(42), 3448–3449 (2012)
26. Dempsey, K., Ali, H.: On the robustness of the biological correlation network model. In: International Conference on Bioinformatics Models, Methods and Algorithms (BIOINFORMATICS 2014), pp. 186–195 (2014)
27. Dempsey, K., Currall, B., Hallworth, R., Ali, H.H.: A new approach for sequence analysis: Illustrating an expanded bioinformatics view through exploring properties of the prestin protein. In: Handbook of Research on Computational and Systems Biology, pp. 202–223 (2011)
28. Zhang, B., Horvath, S.: A general framework for weighted gene co-expression network analysis. Statistical Applications in Genetics and Molecular Biology 4(1) (January 2005), http://dx.doi.org/10.2202/1544-6115.1128
29. Dempsey, K., Currall, B., Hallworth, R., Ali, H.: An intelligent data-centric approach toward identification of conserved motifs in protein sequences. In: Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology. BCB 2010, pp. 398–401. ACM, New York (2010), http://doi.acm.org/10.1145/1854776.1854839
30. Dempsey, K., Bonasera, S., Bastola, D., Ali, H.: A novel correlation networks approach for the identification of gene targets. In: 2011 44th Hawaii International Conference on System Sciences (HICSS), pp. 1–8 (January 20011)

Computing Biological Model Parameters by Parallel Statistical Model Checking*

Toni Mancini, Enrico Tronci, Ivano Salvo, Federico Mari, Annalisa Massini, and Igor Melatti

Computer Science Department, Sapienza University of Rome, Italy

Abstract. Biological models typically depend on many *parameters*. Assigning suitable values to such parameters enables *model individualisation*. In our clinical setting, this means finding a model for a given patient. Parameter values cannot be assigned arbitrarily, since *inter-dependency* constraints among them are not modelled and ignoring such constraints leads to biologically meaningless model behaviours. Classical parameter identification or estimation techniques are typically not applicable due to scarcity of clinical measurements and the huge size of parameter space. Recently, we have proposed a statistical algorithm that finds (almost) all biologically meaningful parameter values. Unfortunately, such algorithm is computationally extremely intensive, taking up to months of sequential computation. In this paper we propose a parallel algorithm designed as to be effectively executed on an arbitrary large cluster of multi-core heterogenous machines.

1 Introduction

Systems biology models aim at providing quantitative information about time evolution of biological species. One of the main goals of systems biology in a health-care context is to *individualise* models in order to compute patient-specific predictions (see, e.g., [24]) for the time evolution of species (e.g., hormones).

Depending on the system at hand, many modelling approaches are currently investigated. For example, see [22,21] for an overview on discrete as well as continuous modelling approaches, and [49] for a survey on stochastic modelling approaches. In biological networks modelled with a system of *Ordinary Differential Equations (ODEs)* depending on a set of parameters (as in, e.g., [35,50,39]) model individualisation can be done by assigning suitable values to the model parameters. Such biological models depend on many (easily hundreds of) parameters, whose values cannot be chosen arbitrarily because of *inter-dependency* constraints among them (see, e.g., [26]) that, usually, are not explicitly known and thus are not modelled. If model parameter values are chosen ignoring such constraints, then the resulting model behaviour is biologically meaningless.

Model identification (see, e.g., [27]) techniques are typically used to compute values for model parameters so that a suitable error function measuring mismatch between model predictions and experimental data is minimised (*parameter estimation*). If such a value exists and is unique, the model is said *identifiable*. In a

* This work has been partially supported by the EC FP7 project PAEON (Model Driven Computation of Treatments for Infertility Related Endocrinological Diseases, 600773).

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 542–554, 2015.

clinical setting, for each patient, only a small number of measurements is available, since they can be costly, invasive and time-consuming. Therefore, although in principle model identification techniques could be used to compute *patient-specific* model parameters, in practice, because of the large amount of measurements needed (see, e.g., [9]), they are typically used to compute a *default parameter value* that averages among the behaviours of many patients (as, e.g. in [39]). *Parameter estimation* approaches cannot be used either, since with such a few data they would not take into due consideration inter-dependencies among model parameters [26].

Motivations. To overcome scarcity of measurements, we proposed a two-phase approach [47]. First, an *off-line* phase that accounts for parameter inter-dependencies [26] greatly narrows down the search space to a set S of parameter values yielding *biologically meaningful* model behaviours. Second, an *on-line* phase computes a patient-specific model by selecting in S those parameter values that minimise mismatch with respect to patient measurements. This enables fast patient-specific predictions for the time evolution of each species of interest.

In general, to decide if time evolution of species concentration is biologically meaningful takes a domain expert. To build a general purpose tool that can automatically search through millions of model parameter values, in [47] we proposed a criterion which regards as *Biologically Admissible (BA)* those parameter values entailing time evolutions with a second order statistics *close enough* to that of the model default parameter values.

The computation of the set S of BA values for the model parameters requires to explore the set of all possible values for the parameter vector, that is typically huge. Since an exhaustive exploration would be unfeasible, in [47] a Statistical Model Checking (SMC) based approach is proposed. Nonetheless, the exploration of the parameter space may take *months* of sequential computation.

Main Contributions. Our main contribution is a SMC parallel algorithm and its distributed multi-core implementation to compute the set of all BA parameter values. We propose a master-slave architecture where a single master process (*Orchestrator*) implements the SMC algorithm and delegates to a high number of slaves (*BA Verifier*s) the numerical integration of the ODE system defining the model. As a consequence, our parallel algorithm will benefit from the availability of many heterogeneous computational units (e.g., a data-centre in the cloud).

The SMC algorithm proposed in [47] would require too much synchronisation in a parallel context. Here, we define a new random sampling process at the basis of our SMC algorithm, which enables massive parallelisation of BA Verifiers.

We developed our distributed multi-core tool in the C language using Message Passing Interface (MPI) [42]. We evaluate effectiveness of our approach by using it on the *GynCycle* model in [39], an ODE model which predicts blood concentration of several species during female menstrual cycle. We show that our implementation achieves high efficiency even when using dozens of computational cores (e.g., efficiency is 74% when using 80 cores).

Related Work. The input to our algorithm consists of a system model along with the *default value* for its parameters. The *GynCycle* model considered in

our case study has been presented in [39] and the default (inter-patient) values for its parameters have been computed in [11] using model identification (often referred to as *parameter identification* in our setting) techniques [27].

In recent years, parallel and distributed computing has received attention in order to cope with the complexity of biological systems. See [5] for a survey of parallel methods to solve ODEs, parallel model checking, and parallel simulations in biological applications.

Statistical Model Checking mainly addresses system verification of stochastic systems with respect to probabilistic temporal properties or continuous stochastic properties (see, e.g., [41]). Several parallel and distributed approaches to SMC have been introduced (see, e.g., [3,40]), some of them motivated by the complexity of biological models [4]. Here, we focus on deterministic biological systems modelled with ODEs, and we apply SMC techniques (along the lines of [18]) to infer statistical completeness of our set of BA parameters.

Parallel approaches close to ours are those in [7,44,8], where the problem of computing all (discretised) model parameter values meeting given LTL properties has been investigated. We extend such works in two directions. First, the above mentioned papers focus on piecewise affine ODE systems, whereas we can handle any (possibly non-linear) ODE system. Second, they aim at computing a maximal set of parameters satisfying a given LTL property. Thus, when the model changes, a new LTL property has to be provided by domain experts. Our approach infers such a system property by the default value for the model parameters, thus decreasing the amount of input needed from domain experts.

A key feature of parameter identification approaches is their ability to give information about parameter *identifiability* (see, e.g., [9] and citations thereof). Gradient-based methods, as, e.g., the classical one in [25], provide a local optimum solution to the parameter estimation problem. Global methods, such as [28], provide a global optimum solution whereas heuristics approaches as evolutionary algorithms (see, e.g., [6,45]), provide near-global optimal solutions. All such approaches do not provide information about parameter identifiability. When observations are scarce, parameters usually become non-identifiable. Studying the correlation among system parameters can reduce the number of data needed for identifiability, see for example [37,26]. Our goal here is to support model individualisation from clinical measurements. This means that we need to compute model parameters from a few (say, 3) observations about a small subset (4 in our case study) of the species occurring in the model (33 in our case). Because of scarcity of measurements, neither model identification approaches nor parameter estimation approaches can be used in our setting.

Model checking based parameter estimation approaches have been investigated for example in [20,12,38]. Such approaches differ from ours, since they do not address the problem of automatically restricting the search space. Model checking techniques have been widely used in systems biology, to verify time behaviours. Examples are in [23,19,14,16,35]. Such approaches focus on verifying a given property for the model trajectories, whereas our main problem here is to compute *all* biologically plausible values for the model parameters.

We note that computing the set of *all* model parameter values that satisfy a given property is closely related to that of computing *all* control strategies

satisfying a given property. In a discrete time setting this problem has been addressed, for piecewise affine systems and safety properties, in [33,2,1,34].

2 Background

Unless otherwise stated, all forthcoming definitions are based on [47,43]. Throughout the paper, we denote with $[n]$ the set $\{1, 2, \ldots, n\}$ of the first n natural numbers and with $\mathbb{R}^+$, $\mathbb{R}^{\geq 0}$ and $\mathbb{R}$ the sets of, respectively, positive, non-negative and all real numbers. We also denote with $(\mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0})^*$ the set of pairs $(a, b) \in \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$ such that $a \geq b$.

2.1 Parametric Dynamical Systems

We model biological systems using dynamical systems. Usually, a dynamical system comes equipped with a function space that models both *controllable* (e.g., treatments) and *uncontrollable* inputs (*disturbances*). Here, we do not address treatments or disturbances and accordingly we omit inputs from Def. 1.

Definition 1 (Parametric Dynamical System). *A* Parametric Dynamical System *(or, simply, a Dynamical System) $\mathcal{S}$ is a tuple $(\mathcal{X}, \mathcal{Y}, \Lambda, \varphi, \psi)$, where:*

- $\mathcal{X} = X_1 \times \ldots \times X_n$ *is a non-empty set of* states *(*state space *of $\mathcal{S}$);*
- $\mathcal{Y} = Y_1 \times \ldots \times Y_p$ *is a non-empty set of* outputs *(*output value space*);*
- Λ *is a non-empty set of* parameters *(*parameter value space*);*
- $\psi : \mathbb{R}^{\geq 0} \times \mathcal{X} \to \mathcal{Y}$ *is the* observation function *of $\mathcal{S}$;*
- $\varphi : (\mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0})^* \times \mathcal{X} \times \Lambda \to \mathcal{X}$ *is the* transition map *of $\mathcal{S}$. Intuitively, $\varphi(t_2, t_1, x, \lambda)$ is the state reached by the system (with parameter values λ) at time t_2 starting from the state $x \in \mathcal{X}$ at time t_1.*

Remark 1. To simplify notation, unless otherwise stated, we assume that the set of parameters Λ has the form $\mathcal{X} \times \Gamma$ (where Γ is a non-empty set). Therefore, a parameter $\lambda = (x_0, \gamma) \in \Lambda$ embodies information about the initial state x_0 of a system trajectory. Such a system trajectory is a function of time $x(\lambda)(t)$, which, for each $t \in \mathbb{R}^{\geq 0}$, evaluates to $\varphi(t, 0, x_0, \gamma)$. In the following, abusing notation, we write $x(\lambda, t)$ instead of $x(\lambda)(t)$. Analogously, we write $x_i(\lambda, t)$ $[y_i(\lambda, t)]$ for the time evolution $x_i(\lambda)(t)$ $[y_i(\lambda)(t)]$ of the i^{th} state [output] component with parameters γ starting in x_0 from time 0.

Example 1. Dynamical systems whose dynamics is described by a system of Ordinary Differential Equations (ODEs) depending on parameters are currently of great interest as a mathematical model for biological networks (see, e.g., [15,39]). In this paper, we will use as a case study the *GynCycle* model presented in [39]. It is a ODE model for the feedback mechanisms between Gonadotropin-Releasing Hormone (GnRH), Follicle-Stimulating Hormone (FSH), Luteinizing Hormone (LH), development of follicles and corpus luteum, and the production of Estradiol (E2), Progesterone (P4), Inhibin A (IhA), and Inhibin B (IhB) during the female menstrual cycle. The model aims at predicting blood concentrations of LH, FSH, E2, and P4 during different stages of the menstrual cycle. The model is intended as a tool to help in preparing and monitoring clinical trials with new drugs that affect GnRH receptors (*quantitative and systems pharmacology*).

In our *black-box* approach, the system transition map models our call to a solver (namely, *Limex* [13]) computing a solution to the ODEs defining our dynamical system. This is along the lines of simulation based system level formal verification as in [29,31,30,32,46,10].

2.2 Biological Admissibility

In general, given a value λ for the (vector of) model parameters, it takes a domain expert to decide if a time evolution $x(\lambda, t)$ is *biologically meaningful*. Indeed, many parameter values lead to time evolutions for the model species that are not compatible with the laws of biology. Our goal is to build a general purpose tool that automatically filters out biologically meaningless parameter values. Following [47], we provide a formal criterion for biological admissibility, by asking that the time evolution of $x(\lambda, t)$ is *similar enough* to that of $x(\lambda_0, t)$, that is the one entailed by the model default parameter value λ_0. To this end, we introduce three measures of how similar two trajectories are.

Given a function f from $\mathbb{R}$ to $\mathbb{R}$ and $\alpha, \tau \in \mathbb{R}$, we denote with $f^{\alpha,\tau}$ the function defined by $f^{\alpha,\tau}(t) = f(\alpha(t+\tau))$ for all t. Here, α and τ are used to model, respectively, a stretch and a shift of f. Given two functions f and g from $\mathbb{R}$ to $\mathbb{R}$, the *cross-correlation* (see, e.g., [48]) $\langle f, g\rangle(\xi)$ between f and g is a function of ξ (where $\xi \in \mathbb{R}$ is the *time lag*) defined as: $\langle f, g\rangle(\xi) = \int_{-\infty}^{+\infty} f(t)g(t+\xi)\mathrm{d}t$. We consider the *normalised zero-lag cross-correlation* function $\rho_{f,g}$, defined as $\rho_{f,g} = \frac{\langle f,g\rangle(0)}{\|f\|\|g\|}$, where, for any f, $\|f\|$ is the L^2 norm of f, i.e., $\sqrt{\langle f, f\rangle(0)}$. The higher $\rho_{f,g}$ the more *similar* f and g (e.g., f and g have the same peaks). In particular, $\rho_{f,g}$ is 1 if f is equal to g up to an amplification factor.

Let $\mathcal{S}$ be dynamical system with n state variables and a default parameter value λ_0. Given a parameter value λ and a finite horizon $h \in \mathbb{R}^{\geq 0}$, let $x_i(\lambda_0, t)$ and $x_i(\lambda, t)$ be the time evolutions of species x_i (for each $i \in [n]$) under parameters λ_0 and λ respectively. Being time evolutions, both $x_i(\lambda_0, t)$ and $x_i(\lambda, t)$ are defined for $0 \leq t \leq h$. Anyway, to easily match the above general definition of cross-correlation, we define such functions on the whole set of real numbers, as being 0 for any $t < 0$ or $t > h$. In order to model biological admissibility, we define the following three functions (i ranges over $[n]$, $\alpha, \tau \in \mathbb{R}$):

$$\rho_{\lambda_0,\lambda,i}(\alpha,\tau) = \rho_{x_i(\lambda_0),x_i^{\alpha,\tau}(\lambda)} \qquad \mu_{\lambda_0,\lambda,i}(\alpha,\tau) = \left| \frac{\int_0^h (x_i(\lambda_0,t) - x_i^{\alpha,\tau}(\lambda,t))\mathrm{d}t}{\int_0^h x_i(\lambda_0,t)\mathrm{d}t} \right|$$

$$\chi_{\lambda_0,\lambda,i}(\alpha) = \left| (\|x_i(\lambda_0)\|^2 - \|x_i^{\alpha,\tau}(\lambda)\|^2) \right| \;/\; \|x_i(\lambda_0)\|^2$$

The *normalised zero-lag cross-correlation* $\rho_{\lambda_0,\lambda,i}(\alpha,\tau)$ measures the similarity of the trajectories $x_i(\lambda_0, t)$ and $x_i(\lambda, t)$ as for qualitative aspects (for example, if they have the same peaks), when $x_i(\lambda, t)$ is subject to stretch α and time-shift τ. The *normalised average differences* $\mu_{\lambda_0,\lambda,i}(\alpha,\tau)$ and the *normalised squared norm differences* $\chi_{\lambda_0,\lambda,i}(\alpha,\tau)$ are two measures of the average distance between $x_i(\lambda_0, t)$ and $x_i(\lambda, t)$, when $x_i(\lambda, t)$ is subject to stretch α and time-shift τ.

In Def. 2, we use these functions to formalise the notion of Biologically Admissible (BA) parameter λ with respect to a default parameter λ_0. Intuitively, λ is BA if the three measures above are all above or below certain thresholds.

Definition 2 (Biologically Admissible parameter). *Let* λ_0, $\lambda \in \mathcal{X} \times \Lambda$ *be two parameters. Let* $\mathbb{A} \subseteq \mathbb{R}^+$, $\mathbb{B} \subseteq \mathbb{R}$ *be two sets of real numbers such that* $1 \in \mathbb{A}$ *and* $0 \in \mathbb{B}$. *Given a tuple* $\Theta = (\theta_1, \theta_2, \theta_3)$ *of positive real numbers, we say that* λ *is* Θ-biologically admissible *with respect to* λ_0, *notation* $\mathrm{adm}_{\mathbb{A},\mathbb{B}}(\lambda_0, \lambda, \Theta)$, *if there exist* $\alpha \in \mathbb{A}$ *and* $\tau \in \mathbb{B}$ *such that, for all* $i \in [n]$: $(\rho_{\lambda_0,\lambda,i}(\alpha,\tau) \geq \theta_1) \wedge (\mu_{\lambda_0,\lambda,i}(\alpha,\tau) \leq \theta_2) \wedge (\chi_{\lambda_0,\lambda,i}(\alpha,\tau) \leq \theta_3)$. □

3 Computation of Admissible Parameters

Our goal is to compute the set S of (with high confidence) all Biologically Admissible (BA) parameter values with respect to a default parameter value λ_0 validated by the model designer as biologically meaningful.

Since small differences in values are meaningless from a biological point of view, we consider a (grid-shaped) *discretised parameter space* $\hat{\Lambda}$ that is a finite subset of the set of possible parameter values Λ. An exhaustive search on $\hat{\Lambda}$ would be unfeasible, due to the large number of parameters to identify (75 in our case study) that makes $\hat{\Lambda}$ huge (10^{75} elements if we consider 10 possible values for each parameter). To overcome such an obstruction, we follow an approach inspired by Statistical Model Checking (SMC) [18,17]. Statistical Hypothesis Testing is used in [47] to compute, with high statistical confidence, the set S of all BA values with respect to a default value λ_0 for the model parameters.

Given arbitrary values in $(0, 1)$ for ε (probability threshold) and δ (confidence threshold), the SMC algorithm in [47] computes the set S of BA parameters by randomly sampling the discretised parameter space $\hat{\Lambda}$ and adding to S those parameter values $\lambda \in \hat{\Lambda}$ which are shown (by simulation) to be BA. The algorithm terminates when set S remains unchanged after $N = \lceil \ln \delta / \ln(1-\varepsilon) \rceil$ attempts. At this point, following [18], in [47] it is proved, that, with statistical confidence $1 - \delta$, the probability that the sampling process will extract a BA parameter vector value not already in S is less than ε.

Unfortunately, the SMC algorithm proposed in [47] *cannot* be extended to work in a parallel context efficiently, as too much synchronisation would be required. Here, we define a new random sampling process at the basis of our SMC approach, which enables massive parallelisation of the computation of S. Our new algorithm has been explicitly designed as to be easily deployed on a cluster of heterogeneous multi-core machines connected by a network.

3.1 Algorithm Outline

An overall high-level view of our algorithm deployed on multiple machines connected by a network is shown in Fig. 1. The parallel algorithm that we present here consists of one *Orchestrator* and many *BA Verifiers*.

Orchestrator. The orchestrator initialises the set S of BA parameter values to the singleton set $\{\lambda_0\}$. Then, at each iteration, it randomly chooses N parameter values $\lambda_1, \ldots, \lambda_N \in \hat{\Lambda}$ independently (where $N = \lceil \ln \delta / \ln(1-\varepsilon) \rceil$) and delegates the verification of each of them to an idle BA Verifier. After having collected all the N answers, the Orchestrator adds to the set S those parameter values

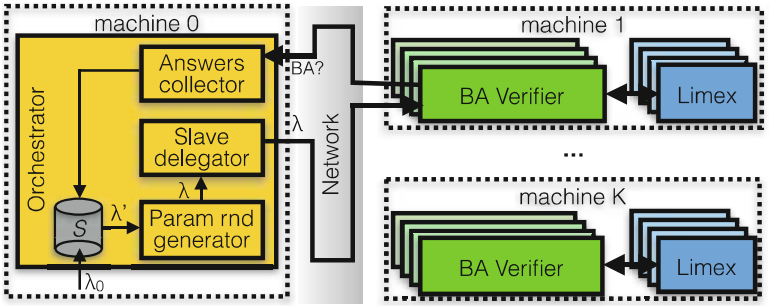


Fig. 1. Parallel algorithm Architecture

returned as BA. If S changes (i.e., at least one of the N randomly generated parameters is BA and not already in S), a new round of this process starts, otherwise the set S computed so far is returned as the final result.

The sampling space $\hat{\Lambda}$ is given by the set of discretised values for the model parameters. Our sampling strategy (Sect. 3.2) guarantees that any parameter value $\lambda \in \hat{\Lambda}$ can be extracted with non-zero probability, as required by [18,47]. To speed up our procedure, we give a higher probability to parameter values that differ from some parameters already in S for a small number of components.

Note that, at each iteration, the Orchestrator adds up to N parameters to S. Thus, increasing the number of parallel BA Verifiers helps a faster growth of the set of BA parameter values S.

BA Verifiers. Each BA Verifier repeatedly takes a parameter λ as input from the Orchestrator, checks if it is Biologically Admissible, and sends back the answer (consisting of the result and the parameter value) to the Orchestrator. To check whether parameter λ is admissible, the BA Verifier in charge runs its *own* instance of the *Limex* solver to compute the time evolutions of all species under parameter λ and checks whether the normalised zero-lag cross-correlation, the normalised average differences, and the normalised squared norm differences for all species are above or below the given thresholds $\Theta = (\theta_1, \theta_2, \theta_3)$, as prescribed by Def. 2.

We observe that the computation distributed to the BA Verifiers is the heaviest part, since it entails to numerically solve the system of differential equations (for a given discretisation of the time output period $[0, h]$ into a finite set T of time-points) and to compute the functions defined in Sect. 2.2 by numerical integration. In order to speed up their computation, BA Verifiers invoke the *Limex* solver just once for each parameter value λ: given the requested finite output time set T and the sets $\mathbb{A}$ and $\mathbb{B}$ for the allowed stretch and time-shift factors, they simulate the system $\mathcal{S}$ computing the trajectory $(t, x(\lambda, t))$ for all time points in a set $T_{\mathbb{A},\mathbb{B}}$ defined as $T \cup \{t' \mid t' = \alpha(t + \tau), t \in T, \alpha \in \mathbb{A}, \tau \in \mathbb{B}\}$. The set $T_{\mathbb{A},\mathbb{B}}$ contains all time instants in which species values are to be known in order to evaluate whether parameter λ satisfies Def. 2.

3.2 Parameter Probability Space

The probability distribution over the discretised parameter space $\hat{\Lambda}$ used by the Orchestrator to generate new parameter values to examine is parametric to the

set S of BA parameter values found so far. To speed up the finding of new BA parameter values (with respect to, e.g., uniform sampling), parameter values that are close to those in S are most likely to be chosen.

Given a set S, we extract the N values $\lambda_1, \ldots, \lambda_N$ to examine at each iteration of the Orchestrator independently as follows. For all $i \in [1, N]$: 1) We randomly choose $\lambda'_i \in S$ considering a uniform probability distribution over S. 2) We randomly choose the maximum number h_i of components in which λ_i will differ from λ'_i. In this case, the set $[n]$ is considered distributed as a power-law of the form $\mathbf{Pr}[h] = a{h_i}^{-b}$, with $b > 1$ and a being a normalisation constant. This implies that, with high probability, λ_i will differ from λ'_i in a small number of components. 3) We randomly choose a subset H_i of h_i different components in $[n]$, assuming a uniform distribution over the set of subsets of cardinality h_i. 4) Finally, the parameter value λ_i is such that for all $j \in H_i$ $\lambda_{i,j}$ is choosen in $\hat{\Lambda}_j$ uniformly at random and $\lambda_{i,j} = \lambda'_{i,j}$ for all $j \in [n] \setminus H_i$.

This sampling technique defines a probability space $(\hat{\Lambda}, \mathcal{P}(\hat{\Lambda}), \mathbf{Pr}^S)$ parametric with respect to a set $S \subseteq \hat{\Lambda}$. By multiplying the (conditional) probabilities of steps 1)–4) above, we have: $\mathbf{Pr}^S[\lambda] = \frac{1}{|S|} \sum_{\lambda' \in S} a\ |d(\lambda, \lambda')|^{-b} \binom{n}{|d(\lambda,\lambda')|}^{-1} \prod_{i \in d(\lambda,\lambda')} \frac{1}{|\hat{\Lambda}_i|}$, where $d(\lambda, \lambda')$ is the set of the components on which λ and λ' differ. Note that $\mathbf{Pr}^S[\lambda]$ is non-zero for all λ.

3.3 Algorithm Correctness

The guarantee that, upon termination, with high statistical confidence, all BA parameter values are in S depends only on the fact that the sampling process consecutively fails N times to find a BA parameter value outside S, and not on how the set S has been populated in the previous iterations of the algorithm.

Stemming from the above considerations, we show the following theorem, stating the correctness of our parallel algorithm.

Theorem 1. *Given a dynamical system $\mathcal{S}$ as in Def. 1, a finite subset $\hat{\Lambda}$ of Λ, a value $\lambda_0 \in \hat{\Lambda}$, a tuple Θ of biological admissibility thresholds, two real numbers ε and δ in $(0, 1)$, and two finite sets of real numbers $\mathbb{A}$ and $\mathbb{B}$ (with $1 \in \mathbb{A}$ and $0 \in \mathbb{B}$), our parallel algorithm is such that:*

1. it terminates;

2. upon termination, it computes a set $S \subseteq \hat{\Lambda}$ of Θ-Biologically Admissible parameter values;

3. with confidence $1 - \delta$: $\mathbf{Pr}^S[\{\lambda \in \hat{\Lambda} \setminus S \mid adm_{\mathbb{A},\mathbb{B}}(\lambda_0, \lambda, \Theta)\}] < \varepsilon$. □

4 Experimental Results

The computational effectiveness of our distributed multi-core implementation has been evaluated on the *GynCycle* model [39]. Such a model has 114 parameters, 75 of which are patient-specific (at least for our purposes), and consists of 41 differential equations defining the time evolution of 33 species.

We implemented our tool in the C programming language using Message Passing Interface (MPI) [42] to enable the communication between the Orchestrator and BA Verifiers spread on multiple machines connected by a network.

4.1 Experimental Setting

Experiments have been carried out on a cluster of 7 Linux heterogeneous machines: 1 machine equipped with 2 × Intel(R) Xeon(R), 2.83 GHz and 8GB of RAM (category A), 2 machines equipped with 2 × Intel(R) Xeon(R), 2.66 GHz and 8GB of RAM (cat. B), and 4 machines equipped with 2 × Intel(R) Xeon(R), 2.27 GHz and 16GB of RAM (cat. C). We used a maximum number of 81 CPU cores (7 out of the 8 available cores for machines of categories A and B and 15 out of the 16 available cores for machines of cat. C). The single Orchestrator process was always run on a core of the machine in cat. A.

We set both ε and δ to 10^{-3}. The stretch factor α (see Def. 2 in Sect. 2.2) ranges in the set $\mathbb{A} = \{0.90, 0.95, 1.00, 1.05, 1.10\}$, while the set $\mathbb{B}$ for the shift factor τ (see Def. 2 in Sect. 2.2) consists of all values from -3 to 3 days multiple of 6 hours. The discretisation $\hat{\Lambda}$ of Λ has been obtained by uniformly discretising the range of each parameter into 5 values. We set Limex to compute time evolutions for all species over $h = 90$ days, returning values with a time step of 15 minutes. Integrals for cross-correlation and norms have been computed numerically with a time step of 15 minutes.

In [47], suitable values for the biological admissibility thresholds $\theta_1, \theta_2, \theta_3$ have been considered, in order to largely cover the set of model meaningful biological behaviours. Here we are interested in evaluating the *speedup* and the *efficiency* of our distributed multi-core algorithm. To this end, in order to execute multiple experiments in reasonable time, we set the biological admissibility thresholds $\theta_1, \theta_2, \theta_3$ to, respectively, 0.99, 0.01, 0.01. Such values are way overly restrictive from a biological point of view, and allow us to compute only a *tiny fraction* (only 8 parameter values) of the set of the BA parameters shown in [47] (which consists of several thousands of Biologically Admissible (BA) parameter values). Anyway, the overall number of random parameter values generated and examined in our case (27620) is sufficiently large to let us correctly evaluate the computational performance of our algorithm.

4.2 Experimental Results

Table 1 shows the overall computation time (column "*time*") when varying the number of BA Verifiers (col. "*# proc.*") used in parallel by our algorithm. Each BA Verifier runs on a *different* core of a machine in our cluster. To make the different values comparable (given the stochastic nature of our algorithm and the heterogeneity of our cluster machines), we started all runs using the *same* random seed and used the *same* proportion of machines of each category in all runs (col. "*# cores*"). To neutralise biases due to the heterogeneity of our cluster machines, we determined the computation time of our algorithm when using a *single* BA Verifier (sequential time) by carrying out three runs allocating the (single) BA Verifier on a core of a machine of each category. Such computation times are listed in Table 2. From such data we have computed the completion time in the first line of Table 1 by averaging the three sequential execution times, using the proportion of the number of cores for each machine category as weights.

Column "*speedup*" in Table 1 shows the speedup achieved by our algorithm. For each number v of parallel BA Verifiers, the speedup is the ratio t_v/t_1, where t_v and t_1 are the computation times shown in Table 1 when using, respectively,

Table 1. Computation times

#proc.	# cores			time	speedup	eff.
	A	B	C	(h:m:s)		
1	–	–	–	*238:16:55*	1×	100%
26	2	4	20	9:16:57	25.67×	98.73%
52	4	9	39	5:16:25	45.18×	86.88%
80	6	14	60	4:1:12	59.27×	74.09%

Table 2. Sequential time

machine cat. for the sequential alg.	time (h:m:s)
A	194:47:45
B	206:19:15
C	250:5:18

v and 1 BA Verifiers. Column "*eff.*" shows the efficiency of our algorithm and is computed, as typically done in the evaluation of parallel algorithms, by dividing the speedup by the number of the parallel BA Verifiers used.

From Table 1 we can see that our distributed multi-core implementation scales well with the number of used parallel BA Verifier instances. The observed lack of efficiency, mostly due to network delays, is typical in a cluster setting. We note that high-performance parallel simulation typically has efficiency values in the range 40%-80% (e.g., see [36]). Accordingly, an efficiency of 74% (last row of Table 1) is to be considered state-of-the-art.

5 Conclusions

We presented a parallel algorithm which efficiently computes the set of Biologically Admissible (BA) parameters for an ODE-based biological model. In our approach, this is a crucial step to enable fast computation of patient-specific predictions from clinical trials. The main ingredient of our parallel algorithm is a novel random sampling process which allows the parallel execution of an arbitrarily high number of processes to check their biological admissibility (which is the most computationally demanding part). Such processes are independent and communicate only with an orchestrator. Our results show that our distributed multi-core implementation scales well with the number of available cores.

References

1. Alimguzhin, V., Mari, F., Melatti, I., Salvo, I., Tronci, E.: A map-reduce parallel approach to automatic synthesis of control software. In: Bartocci, E., Ramakrishnan, C.R. (eds.) SPIN 2013. LNCS, vol. 7976, pp. 43–60. Springer, Heidelberg (2013)
2. Alimguzhin, V., Mari, F., Melatti, I., Salvo, I., Tronci, E.: On-the-fly control software synthesis. In: Bartocci, E., Ramakrishnan, C.R. (eds.) SPIN 2013. LNCS, vol. 7976, pp. 61–80. Springer, Heidelberg (2013)
3. AlTurki, M., Meseguer, J.: pVeStA: A parallel statistical model checking and quantitative analysis tool. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) CALCO 2011. LNCS, vol. 6859, pp. 386–392. Springer, Heidelberg (2011)
4. Ballarini, P., Forlin, M., Mazza, T., Prandi, D.: Efficient parallel statistical model Checking of Biochemical Networks. In: Proc. of PDMC, EPCTS 2014, pp. 47–61 (2009)
5. Ballarini, P., Guido, R., Mazza, T., Prandi, D.: Taming the complexity of biological pathways through parallel computing. Briefings in Bioinformatics 10(3), 278–288 (2009)

6. Balsa-Canto, E., Peifer, M., Banga, J.R., Timmer, J., Fleck, C.: Hybrid optimization method with general switching strategy for parameter estimation. BMC Systems Biology 2, 26 (2008)
7. Barnat, J., Brim, L., Černá, I., Dražan, S., Šafránek, D.: Parallel model checking large-scale genetic regulatory networks with DiVinE. ENTCS 194(3), 35–50 (2008)
8. Barnat, J., Brim, L., Šafránek, D., Vejnár, M.: Parameter scanning by parallel model checking with applications in systems biology. In: Proc. of HiBi/PDMC, pp. 95–104. IEEE (2010)
9. Chis, O.-T., Banga, J.R., Balsa-Canto, E.: Structural identifiability of systems biology models: A critical comparison of methods. PLoS ONE, 6(11) (2011)
10. Della Penna, G., Intrigila, B., Tronci, E., Venturini Zilli, M.: Synchronized regular expressions. Acta Inf. 39(1), 31–70 (2003)
11. Dierkes, T., Röblitz, S., Wade, M., Deuflhard, P.: Parameter identification in large kinetic networks with BioPARKIN. CoRR, abs (2013)
12. Donaldson, R., Gilbert, D.: A model checking approach to the parameter estimation of biochemical pathways. In: Heiner, M., Uhrmacher, A.M. (eds.) CMSB 2008. LNCS (LNBI), vol. 5307, pp. 269–287. Springer, Heidelberg (2008)
13. Ehrig, R., Nowak, U., Oeverdieck, L., Deuflhard, P.: Advanced extrapolation methods for large scale differential algebraic problems. In: High Performance Scient. and Eng. Comp. LNCSE (1999)
14. Gong, H., Zuliani, P., Komuravelli, A., Faeder, J.R., Clarke, E.M.: Analysis and verification of the hmgb1 signaling pathway. BMC Bioinform. 11(S-7), S10 (2010)
15. Gong, H., Zuliani, P., Komuravelli, A., Faeder, J.R., Clarke, E.M.: Computational modeling and verification of signaling pathways in cancer. In: Horimoto, K., Nakatsui, M., Popov, N. (eds.) ANB 2010. LNCS, vol. 6479, pp. 117–135. Springer, Heidelberg (2012)
16. Gong, H., Zuliani, P., Wang, Q., Clarke, E.M.: Formal analysis for logical models of pancreatic cancer. In: Proc. of 50th CDC, pp. 4855–4860. IEEE (2011)
17. Grosu, R., Smolka, S.A.: Quantitative model checking. In: Preliminary Proc. of ISoLA, pp. 165–174 (2004)
18. Grosu, R., Smolka, S.A.: Monte carlo model checking. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 271–286. Springer, Heidelberg (2005)
19. Heath, J., Kwiatkowska, M.Z., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. Theor. Comput. Sci. 391(3), 239–257 (2008)
20. Hussain, F., Dutta, R.G., Jha, S.K., Langmead, C.J., Jha, S.: Parameter discovery for stochastic biological models against temporal behavioral specifications using an sprt based metric for simulated annealing. In: Proc. of 2nd ICCABS, pp. 1–6. IEEE (2012)
21. Ingalls, B., Iglesias, P.: Control Theory and Systems Biology. MIT Press (2009)
22. De Jong, H.: Modeling and simulation of genetic regulatory systems: A literature review. Journal of Computational Biology 9, 67–103 (2002)
23. Kwiatkowska, M., Norman, G., Parker, D.: Using probabilistic model checking in systems biology. ACM SIGMETRICS Perf. Eval. Rev. 35(4), 14–21 (2008)
24. Langmead, C.J.: Generalized queries and bayesian statistical model checking in dynamic bayesian networks: Application to personalized medicine. In: Proc. of CSB, pp. 201–212 (2009)
25. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. The Quarterly of Applied Math 2, 164–168 (1944)
26. Li, P., Vu, Q.D.: Identification of parameter correlations for parameter estimation in dynamic biological models. BMC Systems Biology 7(1), 91 (2013)

27. Ljung, L.: System Identification (2Nd Ed.): Theory for the User. Prentice Hall PTR, Upper Saddle River (1999)
28. Stahl, S., Brusco, M.: Branch-and-Bound Applications in Combinatorial Data Analysis. Statistics and Computing. Springer (2005)
29. Mancini, T., Mari, F., Massini, A., Melatti, I., Merli, F., Tronci, E.: System level formal verification via model checking driven simulation. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 296–312. Springer, Heidelberg (2013)
30. Mancini, T., Mari, F., Massini, A., Melatti, I., Tronci, E.: Anytime system level verification via random exhaustive hardware in the loop simulation. In: Proc. of DSD, pp. 236–245 (2014)
31. Mancini, T., Mari, F., Massini, A., Melatti, I., Tronci, E.: System level formal verification via distributed multi-core hardware in the loop simulation. In: Proc. of PDP (2014)
32. Mancini, T., Mari, F., Massini, A., Melatti, I., Tronci, E.: SyLVaaS: System level formal verification as a service. In: Proc. of PDP. IEEE (2015)
33. Mari, F., Melatti, I., Salvo, I., Tronci, E.: Synthesis of quantized feedback control software for discrete time linear hybrid systems. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 180–195. Springer, Heidelberg (2010)
34. Mari, F., Melatti, I., Salvo, I., Tronci, E.: Model based synthesis of control software from system level formal specifications. ACM TOSEM 23(1), 1–42 (2014)
35. Miskov-Zivanov, N., Zuliani, P., Clarke, E.M., Faeder, J.R.: Studies of biological networks with statistical model checking: Application to immune system cells. In: Proc. of BCB, pp. 728–729. ACM (2007)
36. Phillips, J.C., Sun, Y., Jain, N., Bohm, E.J., Kalé, L.V.: Mapping to irregular torus topologies and other techniques for petascale biomolecular simulation. In: Proc. of SC14, pp. 81–91. IEEE (2014)
37. Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., Timmer, J.: Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. Bioinformatics 25(15), 1923–1929 (2009)
38. Rizk, A., Batt, G., Fages, F., Soliman, S.: On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In: Heiner, M., Uhrmacher, A.M. (eds.) CMSB 2008. LNCS (LNBI), vol. 5307, pp. 251–268. Springer, Heidelberg (2008)
39. Röblitz, S., Stötzel, C., Deuflhard, P., Jones, H.M., Azulay, D.-O., van der Graaf, P., Martin, S.W.: A mathematical model of the human menstrual cycle for the administration of GnRH analogues. Journ. of Theor. Biology 321, 8–27 (2013)
40. Sebastio, S., Vandin, A.: Multivesta: statistical model checking for discrete event simulators. In: Proc. of ValueTools, pp. 310–315 (2013)
41. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 266–280. Springer, Heidelberg (2005)
42. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J.: MPI-The Complete Reference, Vol. 1: The MPI Core, 2nd edn. MIT Press (1998)
43. Sontag, E.D.: Mathematical Control Theory: Deterministic Finite Dimensional Systems (2nd Edition). Springer, New York (1998)
44. Streck, A., Krejci, A., Brim, L., Barnat, J., Safranek, D., Vejnar, M., Vejpustek, T.: On parameter synthesis by parallel model checking. IEEE/ACM Trans. on Comput. Biology and Bioinf. 9(3), 693–705 (2012)
45. Sun, J., Garibaldi, J.M., Hodgman, C.: Parameter estimation using metaheuristics in systems biology: A comprehensive review. IEEE/ACM Trans. Comput. Biology Bioinform. 9(1), 185–202 (2012)

46. Tronci, E., Mancini, T., Mari, F., Melatti, I., Salvo, I., Prodanovic, M., Gruber, J.K., Hayes, B., Elmegaard, L.: Demand-aware price policy synthesis and verification services for smart grids. In: SmartGridComm, pp. 236–245. IEEE (2014)
47. Tronci, E., Mancini, T., Salvo, I., Sinisi, S., Mari, F., Melatti, I., Massini, A., Davì, F., Dierkes, T., Ehrig, R., Röblitz, S., Leeners, B., Krüger, T.H.C., Egli, M., Ille, F.: Patient-specific models from inter-patient biological models and clinical records. In: Proc. of FMCAD, pp. 207–214 (2014)
48. Vaseghi, S.V.: Advanced Digital Signal Processing and Noise Reductio. John Wiley & Sons (2006)
49. Wilkinson, D.J.: Stochastic Modelling for Systems Biology. Chapman & Hall (2006)
50. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to Stateflow/Simulink verification. Formal Methods in System Design 43(2), 338–367 (2013)

Mobile Access to On-line Analytic Bioinformatics Tools

Sergio Díaz Del Pino, Tor Johan Mikael Karlsson, Juan Falgueras Cano, and Oswaldo Trelles

[1] Computer Architecture Department, Málaga University, Louis Pasteur 35, 29071 Málaga, Spain
[2] Integromics S.L, Parque Tecnológico de Ciencias de la Salud, Avenida de la Innovación, nº 1, 18100 Armilla, Granada, Spain
[3] Computer Sciences Department, Malaga University, Louis Pasteur 35, 29071 Malaga, Spain
{sergiodiazdp,tjkarlsson,juanfc,ortrelles}@uma.es

Abstract. The use of mobile devices grow continuously in many aspects of our everyday lives. It is essential that bioinformatics and biomedicine adapt to this trend because these platforms can provide universal access to computational resources independently of their location. We report the implementation of a light-weight client which allows researchers to launch complex analysis experiments and monitor the progress using their mobile devices. Starting from the analysis of existing functionality in current bioinformatics clients for web-services, we have selected, designed and implemented a complete set of functions for accessing computational resources in bioinformatics and biomedicine through mobile devices.

Keywords: Bioinformatics, mobile-based client, Web Services, Cloud Computing.

1 Introduction

It is a commonplace statement that bioinformatics is mostly a web-based domain. The major players in the field; i.e. EBI (EMBL European Bioinformatics Institute, 2013), NCBI (National Center for Biotechnology Information, 2013), INB (The Spanish Institute for Bioinformatics, 2013), etc. provide web-based services to access databases and data analysis applications located in their servers through some form of web-based interface.

On the other hand, the rapid proliferation of mobile devices, including smartphones and tablets, makes the development of new scientific applications for these platforms an urgent issue. Since most mobile devices are endowed with some sort of web-browser it would be reasonable to expect that resources and clients from bioinformatics would be easily adapted to mobile devices. However experience dictates that, from the bioinformatics user perspective, the transition is anything but satisfactory. Moving or adapting web-based applications to mobile devices requires a detailed study on the capabilities of these new devices to define the best way to cover the required functionality.

In this document we describe our experience in client development for mobile devices in bioinformatics. In particular, web-based bioinformatics platforms (e.g. MOWServ 2 (Mateos, 2010) was used as the starting point to move into the mobile's environment

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 555–565, 2015.

(see Section 2). In Section 3, we will describe the developed prototype and finally we will discuss and conclude in Section 4. The prototype is being evaluated in the context of the Mr.SymBioMath EU-project and will become part of the services offered by the Spanish National Institute for Bioinformatics.

2 Background

Although there are several bioinformatics applications already ported to mobile devices, such as: Biocatalogue (http://bit.ly/15m2Rtp), SimAlign (http://bit.ly/188GlGo), Oh BLAST it!... (http://bit.ly/1yYS32q) (full list in supplementary material). However, although such applications are certainly useful, the amount of effort needed to develop specific applications for all bioinformatics software would be unreasonable. This paper reports a general approach in which is only needed to register a new application describing its metadata information to make it available for the general user.

2.1 Considerations for Mobile Interfaces

Many studies address the appropriate transition between WIMP (Windows, Icons, Menus, and Pointer) interfaces and mobile (or touch user) interfaces (GUI), and the details, such as the substitution of the legacy hovering effects with distinct types of animation (Cheung *et al.*, 2012). The interaction style and the metaphor to use must be to redesign.

In particular, biomedical informatics applications require specific considerations regarding the distraction issue in order to be suitable in complex settings (Deegan, Robin, 2013); and they also require particularly suited graphic design to ease handling and avoid cluttering, especially in genetic related displaying (Pfeifer, 2012; Plumlee, 2006). In this environment, achieving trustworthy and usable interactive devices requires the application of Human Computer Interaction (HCI) research techniques, and the awareness of HCI issues throughout the lifecycle, from design through procurement, training and use (Acharya, 2010).

Development of mobile applications is an emerging area and follows few standards, if any. Due to the success of the iOS platform, Android and others, smartphones account for half of all mobile phones in 2013, http://bit.ly/1DTeK9e). It is possible to characterize the development of mobile applications in the following programming models:

- Native applications are developed entirely in the device's native language. This kind of applications is highly coupled to a specific platform, but has access to all the internals of the device (GPS, accelerometer, contacts, calendar, etc.).
- Web-based applications are platform independent and only require a web browser. An important drawback is that they do not have access to some parts of the device.
- Hybrid applications are a merge of the two previous models. There provide an application container developed in the native language, which is in charge of

loading the HTML5 application and displaying it to the user. This model shares some of the advantages of native applications such as access to device's APIs and App Store distribution and also some advantages of HTML5 applications (cross-platform).

2.2 Repositories of Bioinformatics Services: Browsing, Discovering and Invocation

In the very dynamic world of bioinformatics and biomedicine, the number of services is not only high but growing continuously as new tools and data types appear. The large number of such available resources suggests the need for some type of intelligent software organization to facilitate the integrated exploitation of tools (i.e. to avoid the construction of specific interfaces for the services or help the user to discover appropriate tools).

Repositories or catalogues of services and datatypes appear as the option of choice to organise this information. Repositories store the meta-data of the services (parameters, data types, documentation, etc.) in a centralized way. Currently, there are a number of these meta-data repositories in the bioinformatics field, with BioCatalogue (Bhagat et al., 2010) being one of the most representative.

Several tools for browsing, discovering and service invocation from metadata repositories are available (i.e. jORCA (V. Martín-Requena et al, 2010), MOWServ, Taverna (T. Oinn, et al. 2004). In general, these applications offer solutions for typical laptops and desktops computers which are not suitable for mobile devices whose screen is much smaller. For example, searching for a resource can be an issue, especially on large ones lists. Navigating through the entire tool/data type list searching for a specific item can be a tedious and inefficient task. In addition, data entry will be done by touching the screen and not by clicking with a mouse.

2.3 Additional Functionality

The previously mentioned functionality is the most common one in a bioinformatics platform, but additional functionality is also needed:

- File up- and down-load: the system requires a component which allows the users to up- and down-load their files into/from the system (decoupling the data transfer from the invocation call). The component should be able to work for multiple files at the same time; cancel/resume on-going processes and to provide progress information.
- User accounting: the system requires also a component to manage user authentication
- Multi-repository manager: this component should allow the user to select a repository among a list of available repositories. New catalogues can be added by modifying configuration files.

3 System and Methods

3.1 The Proposed Architecture

In the previous section, we have specified the traditional functionality of a Web based application in the bioinformatics and biomedical application domain. This section outlines the development of a mobile application to satisfy such requirements under the current capabilities of these devices.

This work extends our previous developments for desktop and laptops computers, in particular MOWServ 2, by replacing, adapting or re-designing components in MOWServ 2 (the client side) by specific mobile modules sharing the same server side (see Figure 1). The client side is composed of different modules, developed in Javascript, whose main functionality is:

- Browsing the service catalogue.
- Discovering services.
- Invocate services
- File management

Regarding the functionality, our starting point was an initial analysis of what MOWServ2 provided. Due to the successful performance of this software, the decision was to maintain the original server-side architecture. The main advantage is that we were able to use the same server side and thereby reduced the time required for developing the new mobile application.

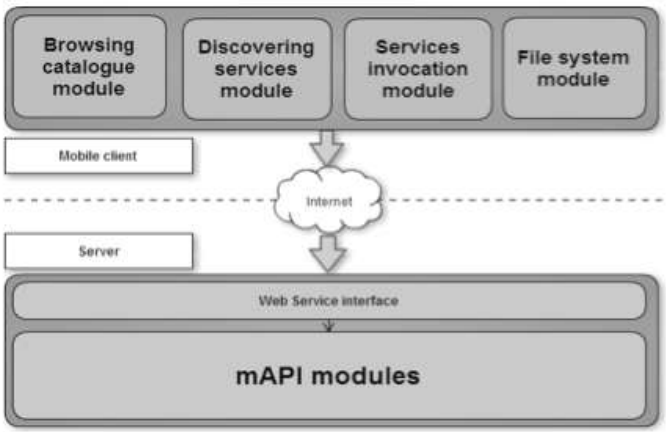


Fig. 1. System Architecture. The application, divided in modules, use a web service to connect with mAPI.

In short, our design consists of a bottom part where MAPI and its web-service act as the main communicators with the repositories and services themselves, and a client-side which process the information and offers the user experience through different modules.

MAPI (Karlsson, 2013) -a framework for the development of Web Services- allow us to use service inputs and outputs in a different format than the one accepted by the native service in a transparent way. The use of MAPI for the invocation also permits the execution of any kind of service that has a corresponding worker (Service invocation module).

3.2 Implementation

After considering the three possible programming models explained in Section 2.1 and benchmarking the available programming options, we determined that web-based development is the best option for our application. Web-based developments are the best choice for those applications that require portability. The state of web-based development tools using HTML5 and JavaScript have matured and offer functional and well-designed frameworks while at the same time is compatible for all major platforms (iOS, Android, Windows Phone).

For this reasons this kind of development is cheaper and, perhaps, faster but results are more limited when it comes to native characteristics of the platform. Native development environments and the amount of available libraries, on the other hand, allow really fast coding and testing but such developments are limited to the specific platform.

3.3 User Interface

The transition between an application thought for a WIMP (Windows, Icons, Menus and Pointer) desktop environment to a touch screen on mobile devices implies losing some of the abilities in the former model. There are also strong limitations such as screen sizes but especially the human-machine interaction should be re-evaluated to improve user experience.

All the previous considerations must be taken into account, due to the importance of the user interface in the relationship between users and electronic devices. In fact, to do some kind of work you normally have to use more than one application. Therefore, we evaluated the best option to present the information from MOWServ 2 in a mobile device.

3.3.1 Browsing the Catalogue

A "TableView" has been used to represent the tree as a list of folders and services. This provides the user a quick overview of the main categories of the entire tree. Then the user is able to navigate through the tree by selecting the categories to follow a path to a specific service. Once a category is selected, a new panel appears from right to left containing the children categories and services.

3.3.2 Discovering Services

A text-box component has been used to discover services filtering the tree depending on user input, showing only the matching services and categories Since the Magallanes tool also demands a string to be used as the search criteria, the idea is the same (results not shown) but invoking a web-service to complete the discovering.

3.3.3 Invocation

The invocation of a Web Service with the mobile application is done in the same way as for MOWServ 2 (i.e. through the MAPI service). Naturally, before making the service call, the user needs to fill the parameters data out (strings and numeric values), paths to data files (including special characters such as backslashes, dots, etc.), or select one entry from a list of files stored remotely. Secondary parameters will be filled with default values as described in the Web Service catalogue. Once the user has filled all the required parameters, it is possible to invoke the service. Following a similar criteria as for the input file, the final result is stored in a remote endpoint. Our philosophy is to provide mechanisms to download results from external storage systems.

3.3.4 File System

For the file system, the solution we have devised is to decouple the up- and down-load of data from the Web Services call. Typically, data for the service calls will not be stored on the mobile devices due to the large data sizes involved in current bioinformatics analysis. In a first step, users must upload the data files to the data storage endpoints using services such as Globus Online. The result is a reference that is used later in the call to the service (i.e. call-by-reference). By following this approach, the user does not need to send the entire input data-files to the service during the call, but sends it before the execution so that it is stored in the server before performing the call.

4 Results

Life Cycle. The typical life-cycle for browsing, discovering, and invoking services in the cloud environment from external clients is illustrated in Figure 2. The process consists –in general- of several steps; The starting point (1) is to register the service(s) into the repository using the Flipper (Torreño, 2014) application. Once the service is registered, its metadata information becomes available for the Client. In order to (2) discover the appropriated service and once users select a given service (3) they provide interfaces to complete the service parameters by accept the user parameter for tuning the application behaviour, based on the metadata retrieved from the repository.

Data consumed and results produced by services are stored in the data storage and can be used instantly as data entry for other services.

Diving into the mORCA application

In order to demonstrate the potential of providing access through mobile devices (i.e. developing such applications), we present two different exercises: The first one is focused on service discovery and the second on enacting a typical service (blast). In fact, these two exercises together conform a simple workflow. In the first service, we will retrieve a sequence that will be used as input for the Blast service.

The mobile client is available at http://bit.ly/1yfr8NC

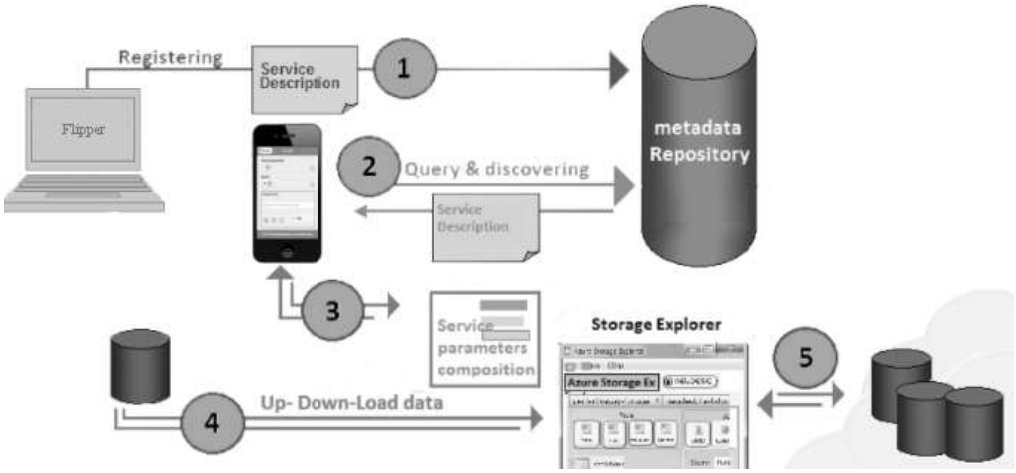


Fig. 2. Life-cycle for (1) service metadata registering; (2) service or datatype discovering,(3) service parameters composition; (4) up- and down-load data into/from the data storage (5) and to communicate to enact the Web Services. More details in the main text.

The first example involves the discovering and retrieval of a given service: 'GetAminoAcid' service. This service can retrieve a biological aminoacid sequence from a database. Initially, it is necessary to (a) log into the system for security reasons. (b) Select the repository where we are going to work (i.e. Bitlab repository). (c) Locate the service by browsing the catalogue (Step 2 in figure 2). After the service is selected, a fast query to the web-service is done and the (d) interface is generated in the client (Step 3 in fig 2). Once the parameters are filled correctly the service can be (e) invoked. Finally, the results of the invocation are showed in the screen and stored in the server for future uses.

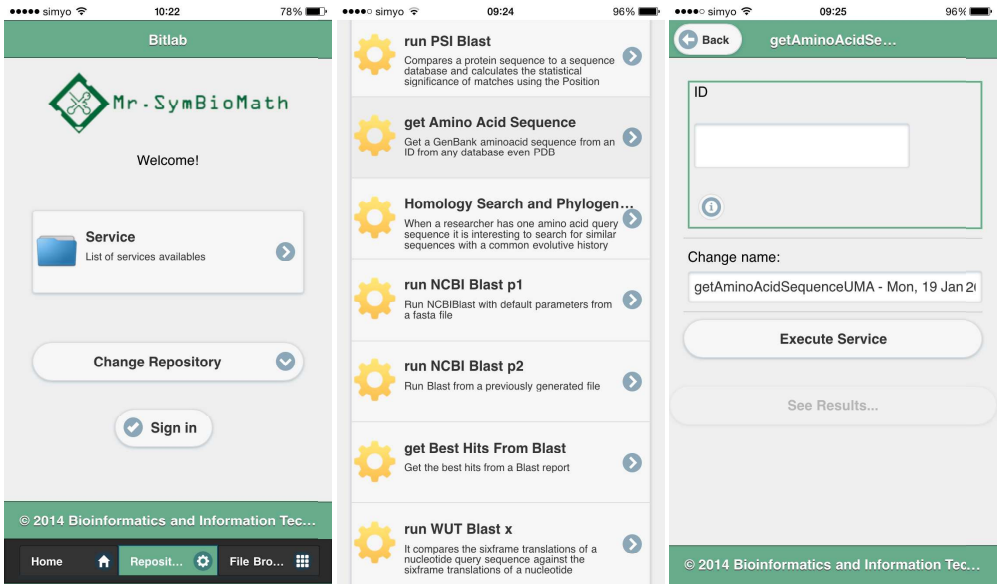


Fig. 3. Dynamically generated interface for the GetAminoacidSequence

In the second exercise, we will execute the well-known Blast application to compare the retrieved sequence against the SwissProt database. For this example, the sequence is already available in the server by our previous exercise but other options are available, i.e. uploading the data or copy and paste the sequence manually.

We use the 'cloud' icon to select our previously obtained sequence and after filling the rest of parameters, we can launch the Blast service.

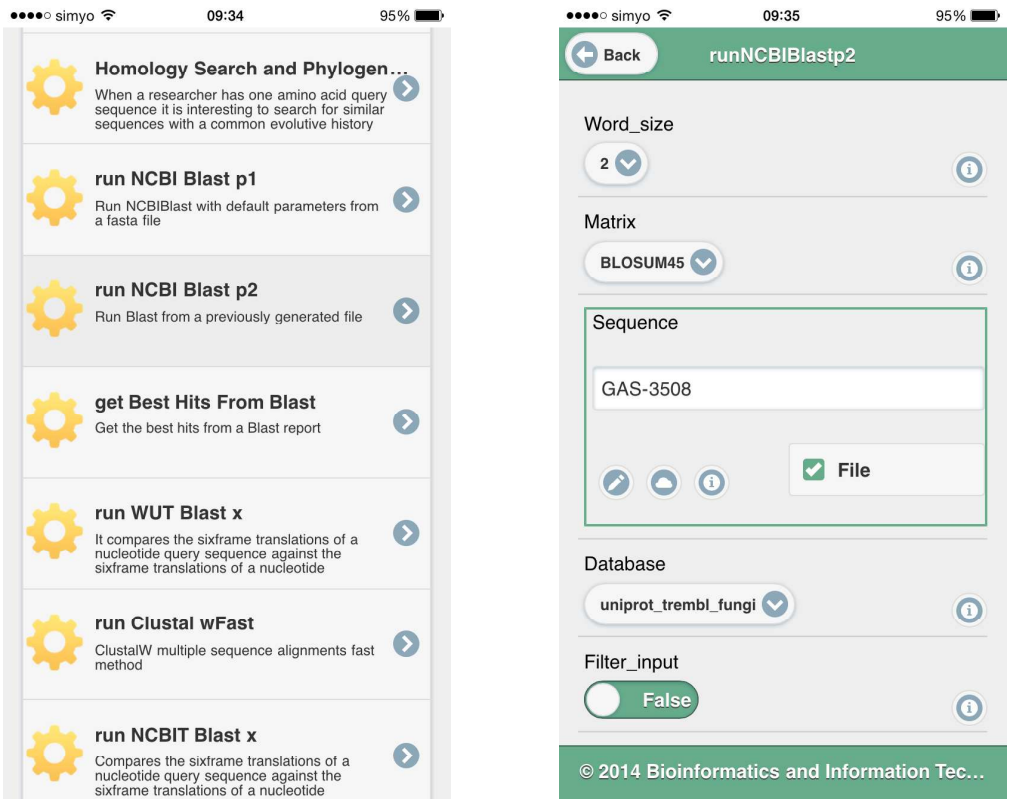


Fig. 4. Dynamically generated interface for Blast

5 Discussion and Conclusions

Mobile devices are increasingly important, as we have said in Section 1, not only because their social environment and human interaction but also due to the possible strong impact on scientific applications. We must capitalize their main advantages such as ubiquity and universal availability.

Our main contribution lies in the experience we got from the development and provision of bioinformatics and biomedical web-services. These application domains are well known to be based on the remote provisioning of services that work over big data collections. This is the framework in which the migration from web to mobiles has been addressed.

Some of the abilities of the interaction language are lost in the translation process, but other possibilities emerge in the new environment. These possibilities do not exist nor were possible / necessary in the legacy software. One of the most notorious is the hovering effect -impossible in the touching interaction- that gives the user an informative feedback when their mouse pointer is over some interactive item. On the other side, effects such as "pinching" (thumb and finger approaching), rotating, two-finger moving, etc. are difficult on WIMP interfaces. However, they are not impossible; we have the example of Apple's Magic Trackpad which can provide this kind of effects in a desktop environment.

Screen sizes are also much more limited and constrain the amount of information that can be displayed, not only because of size but also because the user attention is not only on the device. Another consideration is related to the precision the input devices (fingers) have compared to the exactness of the mouse pointer.

Although some Android-based and Windows Phone-based mobiles have file managers, they are not enough for big data files upload (even if the user is under a Wi-Fi connection). For this reason we decided to separate the file transfer from the service invocation.

We have made an initial evaluation based on real users whom we have asked to test the application. The users were not trained in advance. They did not know any details neither of the interface nor the steps to follow. There were only given a two paragraph description of the goal and the file with the data needed. The first exercise was fairly simple and only intended to be a first step and to provide familiarity with the interface. The second one required the users to change parameters before launching a "Clustal" multi-sequence comparison and implied a two step approach using the corresponding tools from the application

After the completion of the experiments, the user filled out a questionnaire form with 21 points, answering questions about experience with mobile devices, knowledge of the field. In particular, the question about "tasks in which you felt lost" was particularly relevant and helpful for developers.

In summary, this paper reports a prototype mobile application able to browse various service repositories, generate user interfaces dynamically based on service metadata and execute/monitor service progress. The development of the application was preceded by an exhaustive evaluation of an existing web-service client for traditional laptops / desktop computers. The user interface was re-designed to fit the usability aspects of mobile devices.

Acknowledgements. This work has been partially financed by the National Institute for Bioinformatics (www.inab.org) (INB-GN5) PT13-0010012, RIRAAF: Research network of allergies and drugs adverse reactions (RD12/0013/006) and Mr.SymBioMath (IAPP-code 324554) funded by the EU and the JDA-P10-TIC6108. The authors would like to thank the Bitlab research team for invaluable help and contribution with pieces of software.

References

Mateos, J.M., Martinez, A., Trelles, O.: MOWSERV 2: Friendly and extensible web platform for bioinformatics tools integration. In: Proceedings of X Spanish Bioinformatics Symposium (international), pp. 253–256 (2010) ISBN-978-84-614-4481-6

Martín-Requena, V., Rios, J., García, M., Ramírez, S., Trelles, O.: jORCA: Easily integrating bioinformatics Web Services. Bioinformatics 26(4), 553–559 (2010)

Wilkinson, M.D., Links, M.: BioMOBY: An open source biological web services proposal. Briefings in Bioinformatics 3(4), 331–341 (2002)

Karlsson, J., Trelles, O.: MAPI: A software framework for distributed biomedical applications. J. Biomedical Semantics 4, 4 (2013)

Ríos, J., et al.: Magallanes: A web services discovery and automatic workflow composition tool. BMC Bioinformatics 10, 334 (2009)

Martínez, A., Gordon, P., Sensen, C., Trelles, O.: Towards closing the gap between user data and standardized input. In: Network Tools and Applications in Biology (NETTAB 2009) (2009)

Plumlee, M.D., Ware, C.: Zooming versus multiple window interfaces: Cognitive costs of visual comparisons. ACM Trans. Comput.-Hum. Interact. 13(2), 179–209 (2006)

Vardoulakis, L.P., Karlson, A., Morris, D., Smith, G., Gatewood, J., Tan, D.: Using mobile phones to present medical information to hospital patients. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2012, pp. 1411–1420. ACM, New York (2012)

Deegan, R.: Managing distractions in complex settings. In: Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services, MobileHCI 2013, pp. 147–150. ACM, New York (2013)

Iacovides, I., Cox, A.L., Blandford, A.: Supporting learning within the workplace: Device training in healthcare. In: Proceedings of the 31st European Conference on Cognitive Ergonomics, ECCE 2013, pp. 30:1–30:4. ACM, New York (2013)

Acharya, C., Thimbleby, H., Oladimeji, P.: Human computer interaction and medical devices. In: Proceedings of the 24th BCS Interaction Specialist Group Conference, BCS 2010, pp. 168–176. British Computer Society, Swinton (2010)

Cheung, V., Heydekorn, J., Scott, S., Raimund, D.: Revisiting hovering: Interaction guides for interactive surfaces. In: Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces, ITS 2012, pp. 355–358. ACM, New York (2012)

Tirado, O.T., Trelles, O.: Easily registering bioinformatics services metadata. In: ECCB 2014: The 13th European Conference on Computational Biology. Methods and technologies for computational biology, France (2014)

Internet Trends (2013), http://www.kpcb.com/insights/2013-internet-trends (October 7, 2013)

The 'Mobile Only' Internet Generation (2010), http://www.slideshare.net/OnDevice/the-mobile-only-internet-generation (October 7, 2013]

Pew's Internet Mobile | Pew Research Center's Internet & American Life Project, EMBL European Bioinformatics Institute (2013), http://www.pewinternet.org/Commentary/2012/February/Pew-Internet-Mobile.aspx, http://www.ebi.ac.uk/ (October 7, 2013)

National Center for Biotechnology Information (2013), http://www.ncbi.nlm.nih.gov/ (October 7, 2013)

The Spanish institute of bioinformatics, Native, HTML5 or Hybrid, Understanding Your Mobile Application Development Options (2013), http://www.inab.org/, http://s3.amazonaws.com/dfc-wiki/en/images/c/c2/Native_html5_hybrid.png (October 7, 2013)

Globus Online | Reliable File Transfer. No IT required, Smartphones account for half of all mobile phones, dominate new phones purchase in the US (2013), https://www.globusonline.org/, http://www.nielsen.com/us/en/newswire/2012/smartphones-account-for-half-of-all-mobile-phones-dominate-new-phone-purchases-in-the-us.html (October 11, 2013)

Oinn, T., et al.: Taverna: A tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20(17), 3045–3054 (2004)

isDNA: A Tool for Real-Time Visualization of Plasmid DNA Monte-Carlo Simulations in 3D

Adriano N. Raposo and Abel J.P. Gomes

Instituto de Telecomunicações, Universidade da Beira Interior,
Av. Marquês D'Avila e Bolama, 6200-001 Covilhã, Portugal
anraposo@ubi.pt, agomes@di.ubi.pt
https://www.it.ubi.pt/medialab

Abstract. Computational simulation of plasmid DNA (pDNA) molecules, owning a closed-circular shape, has been a subject of study for many years. Monte-Carlo methods are the most popular family of methods that have been used in pDNA simulations. However, though there are many software tools for assembling and visualizing DNA molecules, none of them allows the user to visualize the course of the simulation in 3D. As far as we know, we present here the first software (called isDNA) allowing the user to visualize 3D MC simulations of pDNA in real-time. This is sustained on an adaptive DNA assembly algorithm that uses Gaussian molecular surfaces of the nucleotides as building blocks, and an efficient deformation algorithm for pDNA's MC simulations.

Keywords: pDNA, Monte-Carlo, simulation, visualization, software.

1 Introduction

As known, DNA plays an important role in life sciences research. DNA is made of four subsidiary molecules called *nucleotides*: *adenine* (A); *cytosine* (C); *guanine* (G); and *thymine* (T). A DNA molecule is usually represented by its base-pairs sequence because there is a unique correspondence between the nucleobases in the two strands, in addition to the fact that the atomic structure of the nucleotides is widely known. Thus, a DNA molecule can be assembled using either its atoms or, alternatively, its nucleotides. The advantage in using nucleotides as building blocks of DNA is that we have a speedup of 34×, since a nucleotide has about 34 atoms in general, what is relevant for visualisation purposes [1].

In turn, pDNA molecules own a closed-circular shape, being widely used in several fields including biotechnology, pharmaceutical sciences, and medical research. In a simple way, we can say that pDNA molecules are DNA molecules whose double helix's extremeties are tied up one to another. pDNA molecules are very dynamic and flexible, and the term "closed-circular" does not mean this type of molecules exhibits a perfect circular conformation. Instead, pDNA may acquire a large variety of conformations including *supercoiled*, or even *knotted* conformations. On the other hand, the production of pDNA in laboratory is done in two major steps: *fermentation* and *purification*. In the fermentation step, the

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 566–577, 2015.

pDNA that we want to produce is replicated by a bacterium. The purification step serves the purpose of separating the pDNA of interest in relation to the DNA of the bacteria and other contaminants. This is where pDNA computer simulations can play an important role.

In this sense, computational methods based on laboratory data have been developed to simulate possible conformations for pDNA molecules under certain thermodynamic conditions. The MC method, due to its reliability, is probably the most popular simulation tool for pDNA. The principle behind the MC method is to make the molecule achieve an elastic energy equilibrium state in as few iterations as possible without compromising the effectiveness and reliability of the method. The deformation method traditionally used in MC simulations, known as *crankshaft move*, has a very low acceptance ratio of trials, i.e., many trials are rejected. Even worse, it is the fact that the crankshaft move presents a very unnatural behavior, as it features very sudden motions along large portions of the molecule. To solve these problems, Raposo and Gomes developed a more efficient deformation algorithm for pDNA's MC simulations [2].

In this paper, we combine assembling and visualization algorithms together with the Raposo-Gomes move into a new software tool, called isDNA, which is seemingly the first to allow real-time 3D visualization of pDNA's MC simulations. For that purpose, this new software implements an adaptive assembly algorithm for DNA [1], and an efficient deformation algorithm of pDNA in MC simulations [2]. The pDNA deformation algorithm adopted by isDNA also contributes to the smoothness of the 3D animation with small and controlled changes in the molecule between iterations.

isDNA is implemented as a C++ package, which already includes a simple GLUI/OpenGL graphical user interface (GUI) capable of loading GBK (GenBank) files, allowing also 3D interaction with the user. The main canvas of the GUI shows in real-time the animated course of the pDNA simulation in 3D. Besides, pDNA conformations resulting from MC simulations can be saved into text files. These conformation files can later been used to re-assemble the pDNA molecules. The isDNA source code is publicly available at: `https://github.com/ISDNA`.

2 Related Work

Let us now briefly review DNA assembly methods, as well as traditional deformation algorithms used in Monte Carlo simulations of pDNA.

2.1 DNA Assembly Methods: Predictive versus Adaptive

The *predictive methods* for DNA assembly include two different approaches: the Cambridge meeting method [3]; and the wedge angle method [4]. These approaches try to predict possible trajectories of DNA molecules just from their base-pair sequences. This prediction is based on previously obtained geometric values between consecutive DNA nucleotides. This DNA assembling paradigm is specifically suitable for scenarios where we want to know the likely conformation

(or topology) of DNA base-pair sequences. It is important to clarify that neither the Cambridge methods [3] nor the wedge angle methods [4] are adequate for pDNA simulation purposes, in largely because they assume that the trajectory of the DNA axis in the process of dinucleotide stacking is determined by the sequence of nucleotides. Examples of Cambridge-based software packages are: FREEHELIX [5]; 3DNA [6]; w3DNA [7]; Curves [8]; and Curves+ [9]. In respect to wedge angle software packages, we have the following: CURVATURE [10]; DNACURVE (`http://www.lfd.uci.edu/~gohlke/dnacurve`); NAB [11]; ADN Viewer [12]. However, because they use predictive DNA assembly methods, none of these tools allows 3D visualization of Monte Carlo simulations of pDNA.

In turn, the *adaptive methods* include those due to Raposo and Gomes [1], and to Hornus et al. [13]. These methods are able to adapt specific base-pair sequences to arbitrary conformations, i.e., these methods allow the base-pair sequence to be assembled along an arbitrary DNA trajectory. In this sense, we can say that adaptive DNA assembling methods are particularly suitable for simulation scenarios of DNA, in particular pDNA. This is so because a simulation process produces a sequence of DNA conformations, so that we need to proceed to the DNA assembling for each conformation.

The software presented in this paper belongs to the family of adaptive assembly algorithms; more specifically, it implements the stacking algorithm previously proposed by Raposo and Gomes [1]. In the literature, we find only a few other codes that implement the DNA assembling for arbitrary conformations, namely: NAB [11]; and GraphiteLifeExplorer [13]. Currently, NAB is bundled as part of AmberTools v.13 [14]. Interestingly, GraphiteLifeExplorer enables the user to model DNA molecules of arbitrary length by modeling its helical axis as a quadratic or cubic Bézier curve in space. Finally, it is also important to clarify that, even presenting adaptive capabilities, neither NAB nor GraphiteLifeExplorer allow real-time 3D visualization of pDNA's MC simulations. The real-time capability is achieved by isDNA mainly because the DNA assembly algorithm adopted [1] uses pre-triangulated molecular surfaces of the nucleotides as building blocks (see Figure 1, instead of their atoms, dramatically reducing the number of graphical objects to be assembled and rendered. Taking the pUC19 molecule as an example, adopting an atomistic representation implies rendering approximately 180,000 atoms. In turn, using a nucleotide-based representation for the same molecule, we only have to render about 5,000 building blocks.

2.2 Deformation Methods for pDNA Simulations

In pDNA simulations, it is usual to simplify a DNA molecule through a linear skeleton (i.e., polygonal line consisting of with equal sized segments connecting consecutive nucleotides) that represents its topological conformation. Then, new pDNA conformations are randomly generated by applying deformations to molecule's skeleton. These trial conformations are then subject to evaluation for acceptance/rejection by the simulation method. However, the deformation method used to generate the new trials has been, in essence, the same since it was first introduced in the context of lattice polymer chains. This move was later

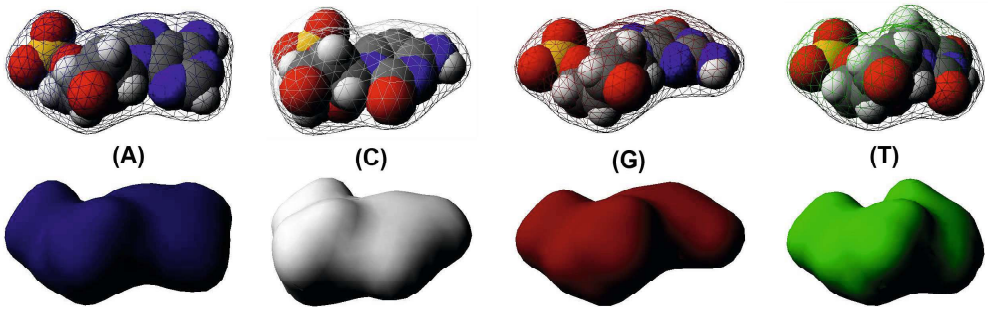


Fig. 1. DNA building blocks: (A) adenine, (C) cytosine, (G) guanine, and (T) thymine. (Image courtesy of Raposo and Gomes [2]).

adapted for pDNA simulation using MC methods. This deformation method is known as the *standard* crankshaft move, with its origins dating back to the early 1960s [15][16].

Later, Klenin et al. [17][18] proposed a biased crankshaft move that starts by randomly choosing two vertices v_m and v_n of the skeleton, rotating then randomly all the vertices —and, consequently, all connecting segments— between v_m and v_n by the angle θ around the axis defined by the line connecting v_m and v_n (Figure 2 (left)). The value of θ is uniformly distributed over a certain interval, and must be continuously adjusted during the simulation to guarantee that about half the steps are accepted [17].

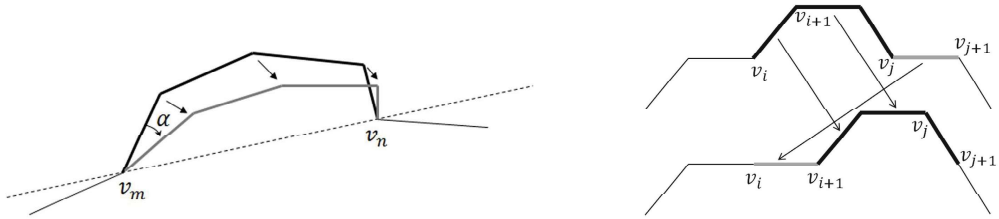


Fig. 2. Crankshaft move (left) and reptation move (right). (Image courtesy of Raposo and Gomes [2]).

In the literature, we can find an alternative deformation method that was thought to increase the acceptance ratio of the simulation trials, which is known as *reptation motion* [18] (Figure 2 (right)). In a simple way, this deformation move is just a sub-chain translation. First, two vertices v_i and v_j are randomly chosen. Then, the sub-chain between v_i and v_j is translated by one segment length along the chain contour. The segment placed immediately after v_j is also translated to fill the gap between v_i and v_{i+1}.

More recently, Raposo and Gomes [2] presented a more efficient unbiased move for pDNA, whose skeleton is a closed polyline. This move, implemented in the isDNA software package, not only preserves the size of each segment and its

connectivity, but also is very effective in maximizing the acceptance ratio of the trials and stabilizing the molecule, thereby allowing steady, gradual temperature changes during the simulation. Our method also generates natural and realistic animations that can be used in real-time simulation and visualization.

Other types of motion can be adopted if Metropolis' microscopic reversibility is satisfied, i.e., if the probability of each trial conformation is the same as that one of the reverse movement [19].

3 Real-Time Adaptive DNA Assembly

For 3D visualization of pDNA simulations in real-time, isDNA implements the DNA assembly algorithm introduced by Raposo and Gomes [1]. This DNA assembly algorithm uses four three-dimensional building blocks representing DNA nucleotides (Figure 1), namely, adenine (A), cytosine (C), thymine (T), and guanine (G). Each building block is a pre-triangulated isosurface generated by a triangulation algorithm for molecules [20].

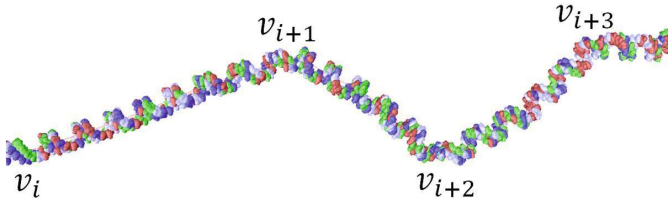


Fig. 3. Piece of assembled DNA. (Image courtesy of Raposo and Gomes [2]).

The DNA axis is approximated by a polyline whose segments have a length of $H = 3.3$ Å (the axial distance between two consecutive nucleotides) [21]. Then, the assembly procedure for nucleotides can be thought of as the operation of wrapping helicoidal DNA backbones around cylinders along the DNA skeleton segments as follows (Figure 3):

1. Given a nucleobase n_i, two geometric instances of nucleotides must be generated, the first for the building block b_i and the second for the mate building block B_i.
2. The base pair b_iB_i is aligned and positioned at the origin laying on the plane $z = 0$.
3. The base pair b_iB_i is aligned with the plane perpendicular to segment i. This alignment is also done about the origin o of the coordinate system.
4. The base pair b_iB_i is translated to the plane perpendicular to segment i.
5. Finally, b_i and B_i are displaced to their correct positions in relation to the midpoint of the corresponding segment i of the DNA axis.

For complete details on the adaptive DNA assembly algorithm implemented in isDNA, the reader is referred to [1].

4 Plasmid DNA Monte-Carlo Simulation Method

The MC simulation method, originally introduced by Metropolis et al. [19], generates pDNA conformations combining energy calculations, random conformational changes, and statistics. This simulation method is considered the standard in pDNA simulations.

4.1 Monte-Carlo Method

MC simulation methods are iterative methods that try to reach a thermodynamic equilibrium state of pDNA molecules based on elastic energy calculations and statistics. So, in each iteration, these methods perform random deformations on the pDNA skeleton until one of the conformation trials is accepted. The acceptance of a new conformation is based on the principle of elastic energy minimization, but the conformation trial can be also accepted if it has a certain probability of occurrence. In isDNA, we used the same MC simulation method and parameters as those used in [22]. For more details on the Monte-Carlo methods, the reader is referred to [2].

4.2 Deformation Algorithm

The pDNA deformation algorithm that was implemented in isDNA has been recently presented as a more efficient way of generating pDNA conformation trials for pDNA's MC simulation procedures [2]. When compared to the traditional deformation methods used in MC simulations, this new deformation method has a higher acceptance ratio of trials, and generates smoother and more controlled deformations, what enhances the real-time animation of the simulation course.

The deformation algorithm implemented in isDNA uses a linear skeleton (i.e., a polyline) with equal sized segments (corresponding to approximately to 30 base pairs of the double helix [23]), henceforth called the DNA skeleton. The pDNA skeleton can assume any closed unknotted conformation, being the completely relaxed circular conformation the simplest of those conformations. Then, the first step of the algorithm is to determine the number of segments of the DNA skeleton, ensuring around 30 base pairs per segment.

Let $\mathbf{P}_k$ a three-dimensional closed polyline representing the DNA skeleton. Now, we need to find a new polyline $\mathbf{P}_{k+1}$ by deforming $\mathbf{P}_k$, but keeping the same number s of equal sized segments and connectivity. From the set of vertices $\{\mathbf{v}_i\}$, $i = 0, ..., s-1$ of the polyline, we choose a random vertex $\mathbf{v}_m$, $0 \leq m \leq s-1$ as the current *mobile vertex*, i.e., the vertex that most moves in the current trial conformation. It happens that any move of $\mathbf{v}_m$ implies moving its closest neighbors $\mathbf{v}_{m-1}$ and $\mathbf{v}_{m+1}$, here called *semi-mobile vertices*. The remaining neighbors $\mathbf{v}_{m-2}$ and $\mathbf{v}_{m+2}$ remain fixed, i.e., they do not move in a deformation step. Therefore, in each deformation step, only three vertices will be displaced: $\mathbf{v}_m$, $\mathbf{v}_{m-1}$ and $\mathbf{v}_{m+1}$.

But, $\mathbf{v}_m$ cannot be freely moved around, unless within the sphere $\mathbf{N}_m$ centered at $\mathbf{v}_m$ itself; the radius of its sphere is $r = 2\,\Delta$, where $\Delta = 3.3$ Å stands for the

distance between two consecutive base pairs. In true, the new position of $\mathbf{v}_m$ is obtained randomly in the intersection of the three spheres, $\mathbf{N}_m$, $\mathbf{S}_{m-2}$ and $\mathbf{S}_{m+2}$, with the latter two spheres with radius $2l$ centered on the fixed vertices $\mathbf{v}_{m-2}$ and $\mathbf{v}_{m+2}$, respectively, where l is the length of each DNA skeleton segment. Note that the small radius r of sphere $\mathbf{N}_m$ guarantees a transition from $\mathbf{P}_k$ to $\mathbf{P}_{k+1}$ without noticeable jumps.

In order to calculate the new position of $\mathbf{v}_m$, we first need to convert the Cartesian coordinates (x, y, z) to spherical coordinates (d, θ, ϕ) relative to $\mathbf{v}_{m-2}$, where d is the distance between $\mathbf{v}_{m-2}$ and $\mathbf{v}_m$. Then, one randomly generates a new position for $\mathbf{v}_m$ as $(d + \Delta d, \theta + \Delta\theta, \phi + \Delta\phi)$, where $\Delta d \in [-r, r]$ and $\Delta\theta, \Delta\phi \in [-\pi, \pi]$. The new position of $\mathbf{v}_{m+1}$ is calculated in a similar manner.

So, the algorithm also converts the Cartesian coordinates of $\mathbf{v}_{m-1}$ to spherical coordinates (l, α, β) relative to $\mathbf{v}_{m-2}$, where l is the radius of the three spheres $\mathbf{s}_m$, $\mathbf{s}_{m-1}$, and $\mathbf{s}_{m-2}$ centered on $\mathbf{v}_m$, $\mathbf{v}_{m-1}$, and $\mathbf{v}_{m-2}$, respectively. Moving $\mathbf{v}_{m-1}$ to a new position must be done without changing its distance l to $\mathbf{v}_{m-2}$ and $\mathbf{v}_m$. That is, the new $\mathbf{v}_{m-1}$ must lie on the circumference on the intersection of the two surfaces bounding $\mathbf{s}_m$ and $\mathbf{s}_{m-2}$. If $\Delta d = 0$, the new position of $\mathbf{v}_{m-1}$ relative to $\mathbf{v}_{m-2}$ is given by $(l, \alpha + \Delta\theta, \beta + \Delta\phi)$; otherwise, the new location of $\mathbf{v}_{m-1}$ is $(l, \alpha + \Delta\theta + \Delta\psi, \beta + \Delta\phi)$, where $\Delta\psi$ is the angle of the angular motion of $\mathbf{v}_{m-1}$ on $\mathbf{s}_{m-2}$ resulting from the translational displacement Δd of $\mathbf{v}_m$ along the line defined by $\mathbf{v}_m$ and $\mathbf{v}_{m-2}$. We compute $\Delta\psi$ by rearranging the equation that describes the reciprocal motion of the piston with respect to the crank angle (cf. [24], p.44).

Being aware that knots can occur when random deformations are applied to pDNA conformations, we must check for the existence of knots and reject the deformation if we find one or more knots. For knot detection isDNA uses the method of Harris and Harvey [25].

As an example of isDNA output, Figure 4 shows a snapshot of a 3D animated MC simulation performed by isDNA using the pUC19 molecule. The MC simulation conditions were the same as the ones presented in [2] when the temperature of the experiments decreases. This snapshot was taken about 1 minute after the beginning of the simulation, with a circular conformation used as the initial conformation of the simulation. For more details on the MC deformation algorithm implemented in isDNA, the reader is referred to [2].

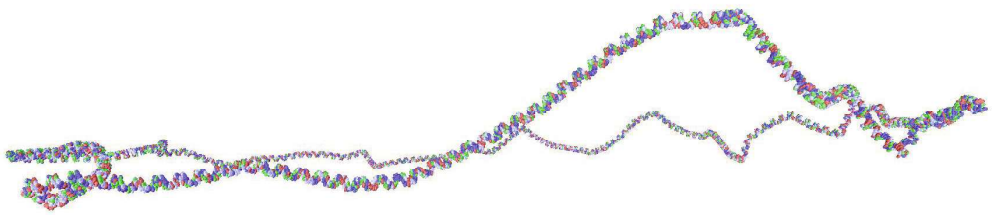


Fig. 4. Snapshot of a 3D animated MC simulation of the pUC19 molecule performed by isDNA

5 The Software

isDNA aims at helping users to perform real-time MC-based pDNA simulation and 3D visualization. However, we can see isDNA as a bundle with two different components: the isDNA GUI (Graphical User Interface) and the isDNA API (Aplication Programming Interface). The idea is to satisfy the needs of two different categories of end users: (a) those needing a simulation tool with embedded 3D visualization capabilities; and (b) those wanting to integrate the isDNA API into third-party 3D visualization tools.

In terms of system requirements, both isDNA GUI and API can be used in general-purpose personal computers without high-performance requirements (i.e., CPU multi-threading and GPU computing) or graphics acceleration. The results presented in this paper were obtained using a laptop equipped with an Intel Core i5-2430M CPU, 2.4GHz clock, 4GB of RAM, and an Nvidia GEFORCE GT 520MX graphics card with 1GB of memory, running Microsoft Windows 7.

5.1 The GUI

The isDNA GUI is very simple and spartan, because it only provides the essential functionalities for 3D visualization and interaction with the pDNA molecules. More specifically, the user interface includes: a 3D canvas where the molecules are rendered; one button that allows the user to rotate de molecules; two buttons to translate the user point of view (one of them for zooming purposes); a select box that allows the user to choose the pDNA molecule; two check buttons that hide and show the molecule components, one for the skeleton and the other for the nucleotides; and, finally, a button to close the application. It it worthy to mention that the user interface was implemented using the GLUI User Interface Library, a GLUT-based C++ user interface library that provides controls such as buttons, checkboxes and radio buttons to OpenGL applications. This means that those who want to recompile isDNA will need the GLUI library that is available at `http://glui.sourceforge.net`.

The aim of this user interface is just to provide the users with a *ready-to-use* solution for pDNA simulation and real-time visualization in 3D. In the future we intend to enhance isDNA user interface with much more functionalities that, at this moment, are just possible for users that integrate isDNA API into their own, or into third-party, visualization tools. Among those new functionalities, we count having a way of saving and loading pDNA conformations and GBK files directly from the user interface. The isDNA API is presented in more detail in the following section.

5.2 The API

The API briefly described in this section is the core of the isDNA software package. This section is particularly useful for those users who want to know more details about how the algorithms are implemented, and also for those users who want to include the isDNA functionalities into their own software tools. However,

at this point, it is important to remember that this is a C++ API, and thus it is only usable in a C++/OpenGL development context. As previously mentioned, the isDNA API source code is available at `https://github.com/ISDNA`.

The two essential classes in the isDNA API are `Dna` and `BuildingBlock`. The `Dna` class is the core class of the API. In this class, we use constants to define DNA geometric properties and MC simulation parameters. The most important methods implemented in the `Dna` class are the following:

- `Dna(char*,char*)` - The constructor of the class. The first parameter is the path to the GBK containing the DNA base pairs sequence. The second parameter is the path to the file that contains the conformation to be loaded.
- `void draw(void)` - Draws the molecule.
- `void drawSkeleton(void)` - Draws the skeleton of molecule.
- `void drawAxis(void)` - Draws the axis of the molecule.
- `void circularConformation(void)` - Builds a circular conformation.
- `void buildAxis(void)` - Builds the axis of the molecule.
- `double randomMoveNew(void)` - Implements the deformation method used by the MC simulation.
- `double twist(void)` - Calculates and returns the *twist* value of the molecule.
- `double writhe(void)` - Calculates and returns the *writhe* value of the molecule.
- `long double bendingEnergy(void)` - Calculates and returns the *bending energy* value of the molecule used in MC simulations.
- `long double torsionalEnergy(void)` - Calculates and returns the *torsional energy* value of the molecule used in MC simulations.
- `int countKnots(void)` - Counts and returns the number of knots in the molecule.
- `void saveConformation(char* file)` - Saves the current conformation of the molecule to the file specified in the parameter.
- `int loadConformation(char* file)` - Load a conformation for the molecule from the file specified in the parameter.
- `void translate(double x, double y, double z)` - Translates the molecule.
- `void rotate(double angle, Point p, Point q)` - Rotates de molecule.

The `BuildingBlock` class is very important because the building blocks used to assemble the molecules are instances of it. The methods implemented in this class are the following:

- `BuildingBlock(char)` - The constructor of the class. The parameter is the nucleotide letter (A, C, G or T).
- `char getNucleotide(void)` - Returns the nucleotide of the building block.
- `GLfloat* getVertex(void)` - Returns the array of vertices of the building block surface mesh.
- `GLfloat* getNormals(void)` - Returns the array of normal vectors of the building block mesh triangles.

- `GLint getVertexCount(void)` - Returns the number of vertices in the building block mesh.
- `GLfloat* getColor(void)` - Returns the color of the building block according to its nucleotide.
- `void draw(void)` - Draws the building block.

isDNA API also implements other generic classes for 3D geometric purposes like, for example, `Vector`, `Point` and `Tuple`, which are not presented here for sake of brevity. Note that isDNA API does not include molecular surface triangulation functionalities. Instead, isDNA API includes four pre-triangulated mesh templates, one for each type of nucleotide building block.

5.3 Features and Future Work

This section presents a summary of the key features of isDNA, as well as an example of a simulation performed with it. At this point, it is also worthy to say that all the simulations presented in [2] were performed using isDNA, which was also used to generate the pictures published therein. Summing up, the most relevant features of isDNA are:

- Real-time visualization of pDNA MC simulations in 3D;
- Loading of GBK (GenBank) files containing the base-pairs sequences of pDNA molecules;
- The user can choose between starting the simulations from a circular conformation or, alternatively, from a previously saved conformation generated, for example, by a former isDNA simulation;
- The user can save conformations into a text file at any point of the pDNA MC simulations;
- The user is allowed to adjust the MC simulation parameters according to their own needs;
- The isDNA API can be fully integrated as a library in the source code of any C++/OpenGL software;
- The isDNA GUI allows the user to choose between a 3D representation and a simplified skeleton representation of a given pDNA molecule;
- The isDNA GUI allows the user to interact with the pDNA molecules in 3D.

In the future, we intend to improve the GUI with more user-friendly functionalities like menus and buttons to save or load pDNA conformations and GKB files, which at this point are only accessible to the API users. We also intend to include adequate functionalities to export the graphical results to image files with illustration quality. Another improvement can be the implementation of a 3D curve smoothing algorithm (like the one due to Kummerle and Pomplun [22]), in order to eliminate sharp kinks that may occur in the simulated conformations.

6 Conclusions

This paper presents isDNA, a novel software tool specifically developed for real-time 3D visualization of pDNA molecules subject to Monte Carlo simulations. As far as the authors know, this is the first software that has these funcionalities, i.e., the existing tools for DNA simulation do not include realistic 3D representation of the molecules and the real-time visualization of the simulation course. These objectives were achieved with the implementation of an adaptive algorithm for DNA assembly that uses instances of the molecular surfaces of the nucleotides as building blocks [1], together with the adoption of an efficient deformation algorithm for pDNA MC simulations [2]. isDNA includes a GUI with basic 3D interaction functionalities that provides a ready-to-use pDNA simulation and visualization tool. Nevertheless, the isDNA API can be integrated into third-party C++/OpenGL sofwtare tools.

References

1. Raposo, A.N., Gomes, A.J.: 3D molecular assembling of B-DNA sequences using nucleotides as building blocks. Graphical Models 74(4), 244–254 (2012)
2. Raposo, A.N., Gomes, A.J.P.: Efficient deformation algorithm for plasmid DNA simulations. BMC Bioinformatics 15(1), 301 (2014)
3. Dickerson, R.E.: Definitions and nomenclature of nucleic acid structure components. Nucleic Acids Research 17(5), 1797–1803 (1989)
4. Bolshoy, A., McNamara, P., Harrington, R.E., Trifonov, E.N.: Curved dna without a-A: experimental estimation of all 16 dna wedge angles. Proceedings of the National Academy of Sciences 88(6), 2312–2316 (1991)
5. Dickerson, R.E.: DNA bending: The prevalence of kinkiness and the virtues of normality. Nucleic Acids Research 26(8), 1906–1926 (1998)
6. Lu, X.J., Olson, W.K.: 3DNA: A software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures. Nucleic Acids Research 31(17), 5108–5121 (2003)
7. Zheng, G., Lu, X.J., Olson, W.K.: Web 3DNA–a web server for the analysis, reconstruction, and visualization of three-dimensional nucleic-acid structures. Nucleic Acids Research 37(web server issue), W240–W246 (2009)
8. Lavery, R., Sklenar, H.: The definition of generalized helicoidal parameters and of axis curvature for irregular nucleic acids. Journal of Biomolecular Structure and Dynamics 6(1), 63–91 (1988)
9. Lavery, R., Moakher, M., Maddocks, J.H., Petkeviciute, D., Zakrzewska, K.: Conformational analysis of nucleic acids revisited: Curves+. Nucleic Acids Research 37(17), 5917–5929 (2009)
10. Shpigelman, E.S., Trifonov, E.N., Bolshoy, A.: CURVATURE: Software for the analysis of curved DNA. Computer Applications in the Biosciences 9(4), 435–440 (1993)
11. Macke, T.J., Case, D.A.: 25. In: Modeling Unusual Nucleic Acid Structure, pp. 379–393. American Chemical Society (1998)
12. Herisson, J., Ferey, N., Gros, P.E., Gherbi, R.: ADN-Viewer: A 3D approach for bioinformatic analyses of large DNA sequences. Cellular and Molecular Biology 52(6), 24–31 (2006)

13. Hornus, S., Levy, B., Lariviere, D., Fourmentin, E.: Easy DNA modeling and more with GraphiteLifeExplorer. PLoS One 8(1), e53609 (2013)
14. Salomon-Ferrer, R., Case, D.A., Walker, R.C.: An overview of the AMBER biomolecular simulation package. Wiley Interdisciplinary Reviews: Computational Molecular Science 3(2), 198–210 (2013)
15. Verdier, P., Stockmayer, W.: Monte Carlo calculations on the dynamics of polymers in dilute solution. The Journal of Chemical Physics 36(1), 227–235 (1962)
16. Hilhorst, H., Deutch, J.: Analysis of Monte Carlo results on the kinetics of lattice polymer chains with excluded volume. The Journal of Chemical Physics 63(12), 5153–5161 (1975)
17. Klenin, K., Vologodskii, A., Anshelevich, V., Dykhne, A., Frank-Kamenetskii, M.: Computer simulation of DNA supercoiling. Journal of Molecular Biology 63(3), 413–419 (1991)
18. Vologodskii, A.V., Levene, S.D., Klenin, K.V., Frank-Kamenetskii, M., Cozzarelli, N.R.: Conformational and thermodynamic properties of supercoiled DNA. Journal of Molecular Biology 227(4), 1224–1243 (1992)
19. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics 21(6), 1087–1092 (1953)
20. Raposo, A.N., Queiroz, J.A., Gomes, A.J.P.: Triangulation of molecular surfaces using an isosurface continuation algorithm. In: Proceedings of the 2009 International Conference on Computational Science and its Applications, pp. 145–153. IEEE Computer Society (2009)
21. Bates, A., Maxwell, A.: DNA Topology, 2nd edn. Oxford University Press (2005)
22. Kummerle, E.A., Pomplun, E.: A computer-generated supercoiled model of the pUC19 plasmid. European Biophysics Journal 34(1), 13–18 (2005)
23. Vologodskii, A.: Monte Carlo simulation of DNA topological properties. In: Monastyrsky, M. (ed.) Topology in Molecular Biology. Biological and Medical Physics, Biomedical Engineering, pp. 23–41. Springer, Heidelberg (2007)
24. Heywood, J.: Internal Combustion Engine Fundamentals. McGraw-Hill, Inc., New York (1988)
25. Harris, B.A., Harvey, S.C.: Program for analyzing knots represented by polygonal paths. Journal of Computational Chemistry 20(8), 813–818 (1999)

Transport Properties of RNA Nanotubes Using Molecular Dynamics Simulation

Shyam R. Badu, Roderik Melnik, and Sanjay Prabhakar

MS2Discovery Interdisciplinary Research Institute, M NeT Laboratory,
Wilfrid Laurier University, Waterloo, ON, N2L 3C5 Canada
{sbadu,rmelnik}@wlu.ca

Abstract. We present novel molecular dynamics studies of transport properties of RNA nanotubes. Specifically, we determine the velocity trajectories for the phosphorous atom at the phosphate backbone of the RNA nanotube, the oxygen atom at sugar ring, and the $^{23}Na^{+}$ and $^{35}Cl^{-}$ ions in physiological solutions. At the constant temperature simulation it has been found that the fluctuation of the velocities is small and consistent with simulation time. We have also presented the velocity autocorrelation function for the phosphorous atom in RNA nanotubes that provides better insight into the diffusion direction of the system in physiological solution. We compare our results calculated computationally with the available experimental results.

1 Introduction

The RNA nanoclusters have a wide range of current and potential applications in a variety of fields, and in particular in nanomedicine. As a result, it is becoming increasingly important to study the properties of these systems in solutions. Particularly, in studies of transport phenomena, properties, and characteristics, including diffusion coefficients and velocity autocorrelation functions for these systems, are the subject of interest. Several experimental studies have been performed to analyze the diffusion coefficients of the biomolecular systems [1,2,11,12]. Furthermore, substantial efforts were devoted to computational studies where molecular dynamics simulations and other methodologies were applied to DNA polymers (e.g. [8,14]). In some such cases the average diffusion coefficient for the solvent as a function of the distance from the solute was reported. By using the molecular dynamics simulation the diffusion coefficient for the single strand RNA can also be calculated [15].

The self-diffusivity of the system of molecules under such studies in molecular dynamics simulations is defined by the random motion of the molecules in the media in which the change of the mass flux is zero. The self-diffusion coefficient can be calculated in two different ways. One is from the root mean square deviation and the other one is from the velocity autocorrelation function. More specifically, once the mean square deviation of the system is defined as

$$MSD(t_1,t_2) = \frac{1}{N}\sum_{i=1}^{N} \|x_i(t_2) - x_i(t_1)\|^2, \tag{1}$$

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 578–583, 2015.

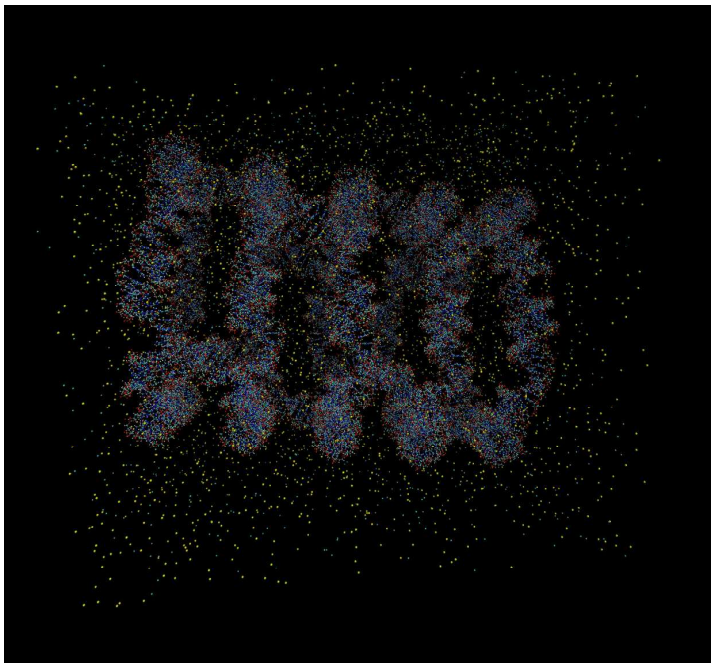


Fig. 1. (Color online.)The VMD generated structure of the RNA nanotube with 5 rings in a physiological solution (water molecules are not shown)

then the self-diffusion coefficient can be expressed as

$$D_s = \lim_{t\to 0} \frac{1}{6Nt} \sum_{i=1}^{N} \|x_i(t_2) - x_i(t_1)\|^2. \tag{2}$$

The autocorrelation function of the system can be expressed as

$$VACF(t) = \frac{< v(t)v(0) >}{(v(0))^2}. \tag{3}$$

In the study of the dynamic properties of the system consisting of RNA nanoclusters we use the structures modeled in the earlier studies [4,9,16]. However, unlike these earlier papers our focus here is on the transport properties such as velocity trajectories, autocorrelation functions and the diffusion coefficient of the nanocluster in physiological solutions The building blocks for these RNA nanoclusters are based on the RNAIi/RNAIIi complexes which are taken from the protein data bank with the pdb code (2bj2.pdb) [6]. A typical example of the RNA nanocluster in the physiological solution is presented in Figure 1.

This contribution is organized as follows. In section 2, we describe computational details used in our analysis of RNA nanoscale systems and highlight the main features of this analysis. The results are presented and discussed in Section 3, while concluding remarks are found in Section 4.

2 Computational Details

In order to perform all atom molecular dynamics simulations on the RNA nanoclusters we used CHARMM27 force field [10] implemented by NAMD package [7] The potential of the system used during the molecular dynamics simulation using CHARMM force filed can be expressed as follows:

$$V_{total} = \sum_{bond} K_b(r-r_0)^2 + \sum_{angle} K_\theta(\theta-\theta_0)^2 + \sum_{dihedral} K_\phi(1+cos(n\phi-\gamma)) + \sum_{Hbond}(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}) + \sum_{Vanderwaals}(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{10}}) + \sum \frac{q_{ij}}{\varepsilon r_{ij}}, \quad (4)$$

where the first term corresponds to bonds, second corresponding to angle parameters, the third term corresponds to the potential energy and interactions arised from the dihedral angles in the molecular system, the fourth term defines the interaction coming from the hydrogen bonds which includes the base pairing as well as the hydrogen bonding between the RNA and the water molecules. Finally the last term in the potential expression represents the long distance interactions known as the (van der Waals' interactions). As it was done for the nanoclusters in [4,9] the modeling of RNA nanotube including pre and post processing of the input-output files have been performed by using the visualization software software VMD and gnuplot. The main features of our analysis here are to calculate the velocity trajectory along the path of molecular dynamics simulation and then to calculate the autocorrelation function which can later be used for calculation of the diffusion properties of the RNA nanocluster in physiological solutions. We note that the RNA-nanotube was solvated by the water in a water box. The size of the box is taken in such a way that the distance wall of the water box is at a distance larger than the cut off radius used in the MD simulation. In order to make the system neutral we have added 594, 924, 1254 and 1584 $^{23}Na^+$ for two ring, three ring, four ring and five ring nanotubes, respectively. Furthermore, in order to make the solution equivalent to the physiological solution we have added extra $^{23}Na^+$ and $^{35}Cl^-$ ions. This system was simulated at constant temperature and pressure using NAMD software. The temperature in the system is controlled by the Langevin method [5] with damping $\eta = 5\ ps^{-1}$.

3 Results and Discussion

Recently, we succeeded in describing RNA nanoclusters of variable sizes by using the molecular dynamics techniques [3,4]. Specifically, we used the RNA building blocks and self assembled them to construct the RNA nanotube using the VMD and the protocols available in the software NAMD. Now, for these nanoclusters we have calculated the trajectories for the velocities, focusing on atoms that may influence substantially dynamical properties of these RNA nanoclusters. Hence, this contribution represents a new steps in the study of transport properties including diffusion phenomena in the RNA nanocluster that can be analyzed via molecular dynamics simulations. In particular the trajectories for the phosphorous atom in the phosphate backbone, the sodium ion in physiological solution, the chloride ion and the oxygen atom at sugar ring of the

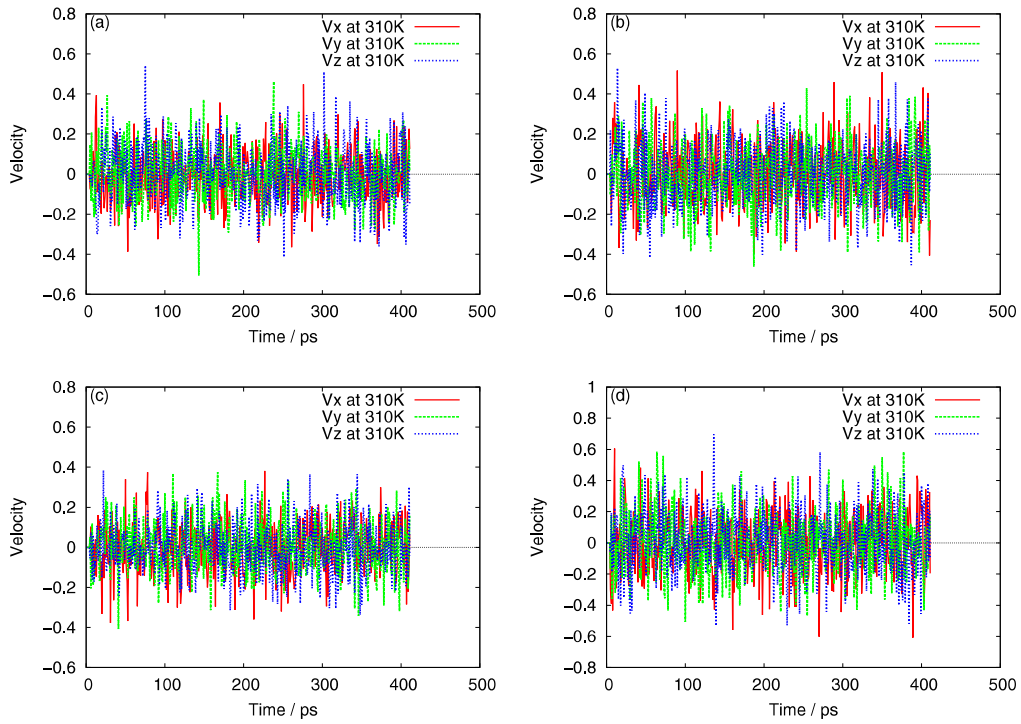


Fig. 2. (color online)The trajectories of the velocity for (a) Phosphorus at the phosphate backbone (b) Sodium ion in a physiological solution (c) Chloride ion in a physiological solution and (d) The oxygen atom at a sugar ring

five ring RNA nanotube are presented in Figure 2. For each of the atoms the velocities have been tracked consistently along all directions i.e x, y, and z. From the plots it is clear that the variation in the velocities during the molecular dynamics simulations is small and consistent in all directions, given constant temperature. This consistency of the velocity components during the MD simulation is due to the consistency of the temperature, along with some variations due to damping. The nature of the fluctuation of the velocities during molecular dynamics simulations is similar to the fluctuations observed for the temperature reported in earlier studies [3,4]. This feature has been observed because the classical velocities are proportional to the temperature of the system. In Figure 3, we have also presented the results for the velocity autocorrelation function in three different directions for the phosphorus atom in the RNA nanotube using molecular dynamics velocity trajectories in physiological solutions. From the plots in Figure 3, we conclude that the variations of the velocity autocorrelation function(VACF) in x direction is significantly different than those in the y and z directions. These velocity autocorrelation functions are primary characteristics for the estimates of diffusion coefficients of the molecular systems under considerations.

Up till now, very limited experimental studies have been done for the diffusion properties of the RNA nanoclusters. However, it is worthwhile noting that some experimental studies on the conformational diffusion coefficient for a typical biopolymer has recently been performed by using the experimental technique force spectroscopy [13]. This shed further light on the inter chain motion in complex biological polymers.

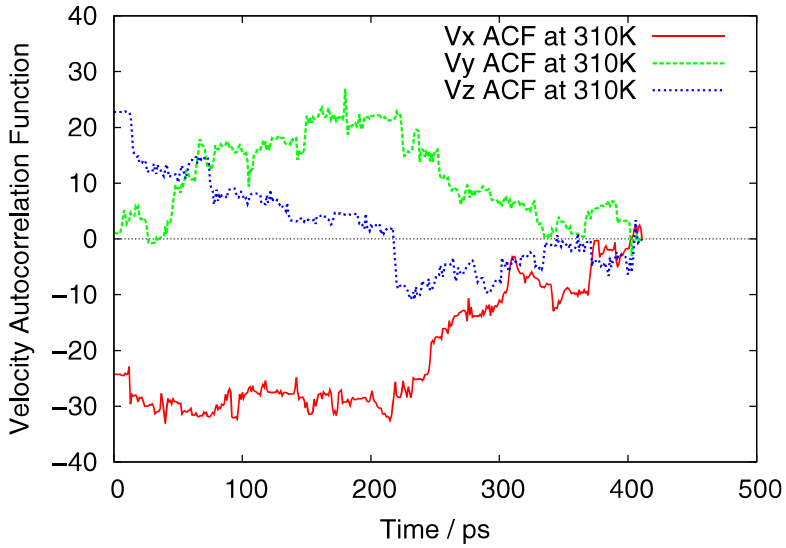


Fig. 3. (color online) Velocity autocorrelation function for the phosphorous atom in the RNA nanotube using molecular dynamics simulation

4 Conclusions and Outlook

In this contribution, we presented new results on dynamic properties of RNA nanotubes, in particular the velocity trajectories and velocity autocorrelation functions for P, O, $^{23}Na^{+}$, and $^{35}Cl^{-}$atoms during molecular dynamics simulations of RNA nanotubes in physiological solutions. Such systems are of particular interest in nanomedical applications [4,9]. In typical NVT runs of constant temperature molecular dynamics simulations for these RNA nanotubes, we found that the velocity is fluctuating in all directions rather uniformly. We have presented the VACFs, focusing on the phosphorous atom at the phosphate backbone of the RNA nanotubes. At the same time, it would be very interesting to calculate the VACFs for other atoms in order to better understand the trend of their velocity variations in different directions. Using these autocorrelation functions deduced from velocity trajectories, a detailed study of diffusion characteristics of such systems represents an important avenue of future work that should provide additional insight into transport phenomena in RNA nanotubes and their applications in biomedicine as well as other fields. Finally, we note that, despite efficient coarse-graining procedures, our current studies have been limited by severe computational challenges and were naturally limited in time scales smaller than realistically required for many biological processes. Longer simulations would provide a better understanding about the transport properties of the RNA nanoclusters. Undoubtedly, these studies should encourage experimentalists to do such kind of measurements that would provide the diffusion parameters on the RNA nanoclusters.

Acknowledgements. Authors are grateful to the NSERC and CRC Program for their support and Shared Hierarchical Academic Research Computing Network (SHARCNET: www.sharcnet.ca) for providing the computational facilities. Finally, we would

like to thank Dr. P. J. Douglas Roberts for helping with technical SHARCNET computational aspects.

References

1. Ando, T., Skolnick, J.: Crowding and hydrodynamic interactions likely dominate in vivo macromolecular motion. PNAS 107(43), 18457–18462 (2010)
2. Arrio-Dupont, M., Foucault, G., Vacher, M., Devaux, P.F., Cribier, S.: Translational diffusion of globular proteins in the cytoplasm of cultured muscle cells. Biophysical Journal 78(2), 901–907 (2000)
3. Badu, S.R., Melnik, R., Paliy, M., Prabhakar, S., Sebetci, A., Shapiro, B.A.: High performance computing studies of RNA nanotubes. In: Proceedings of IWBBIO-2014 (International Work-Conference on Bioinformatics and Biomedical Engineering), pp. 601–607 (2014)
4. Badu, S.R., Melnik, R., Paliy, M., Prabhakar, S., Sebetci, A., Shapiro, B.A.: Modeling of RNA nanotubes using molecular dynamics simulation. Eur. Biophys. J. 43(10-11), 555–564 (2014)
5. Feller, S.E., Zhang, Y., Pastor, R.W., Brooks, B.R.: Constant pressure molecular dynamics simulation: The langevin piston method. The Journal of Chemical Physics 103(11), 4613–4621 (1995)
6. Lee, A.J., Crothers, D.M.: The solution structure of an RNA looploop complex: The ColE1 inverted loop sequence. Structure 6(8), 993–1007 (1998)
7. MacKerell, Bashford, D., Bellott, Dunbrack, Evanseck, J.D., Field, M.J., Fischer, S., Gao, J., Guo, H., Ha, S., Joseph-McCarthy, D., Kuchnir, L., Kuczera, K., Lau, F.T.K., Mattos, C., Michnick, Ngo, T., Nguyen, D.T., Prodhom, B., Reiher, W.E., Roux, B., Schlenkrich, M., Smith, J.C., Stote, R., Straub, J., Watanabe, M., Wirkiewicz-Kuczera, J., Yin, D., Karplus, M.: All-atom empirical potential for molecular modeling and dynamics studies of proteins. J. Phys. Chem. B 102(18), 3586–3616 (1998)
8. Makarov, V.A., Feig, M., Andrews, B.K., Pettitt, B.M.: Diffusion of solvent around biomolecular solutes: A molecular dynamics simulation study. Biophysical Journal 75(1), 150–158 (1998)
9. Paliy, M., Melnik, R., Shapiro, B.A.: Molecular dynamics study of the RNA ring nanostructure: A phenomenon of self-stabilization. Phys. Biol. 6(4), 046003 (2009)
10. Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kal, L., Schulten, K.: Scalable molecular dynamics with NAMD. J. Comput. Chem. 26(16), 1781–1802 (2005)
11. Verkman, A.S.: Solute and macromolecule diffusion in cellular aqueous compartments. Trends in Biochemical Sciences 27(1), 27–33 (2002)
12. Wojcieszyn, J.W., Schlegel, R.A., Wu, E.S., Jacobson, K.A.: Diffusion of injected macromolecules within the cytoplasm of living cells. PNAS 78(7), 4407–4410 (1981)
13. Woodside, M., Lambert, J., Beach, K.: Determining intrachain diffusion coefficients for biopolymer dynamics from single-molecule force spectroscopy measurements. Biophysical Journal 107(7), 1647–1653 (2014)
14. Yang, X., Melnik, R.: Effect of internal viscosity on brownian dynamics of dna molecules in shear flow. Computational Biology and Chemistry 31, 110–114 (2007)
15. Yeh, I.C., Hummer, G.: Diffusion and electrophoretic mobility of single-stranded {RNA} from molecular dynamics simulations. Biophysical Journal 86(2), 681–689 (2004)
16. Yingling, Y.G., Shapiro, B.A.: Computational design of an RNA hexagonal nanoring and an RNA nanotube. Nano Lett. 7(8), 2328–2334 (2007)

Molecular Dynamics Simulations of Ligand Recognition upon Binding Antithrombin: A MM/GBSA Approach

Xiaohua Zhang[1], Horacio Péréz-Sánchez[2,*], and Felice C. Lightstone[1,*]

[1] Biosciences and Biotechnology Division, Physical and Life Sciences Directorate, Lawrence Livermore National Laboratory (LLNL), Livermore, CA, USA
{zhang30,lightstone1}@llnl.gov
[2] Bioinformatics and High Performance Computing Research Group, Department of Computer Science, Universidad Católica San Antonio de Murcia (UCAM), Spain
hperez@ucam.edu

Abstract. A high-throughput virtual screening pipeline has been extended from single energetically minimized structure Molecular Mechanics/Generalized Born Surface Area (MM/GBSA) rescoring to ensemble-average MM/GBSA rescoring. For validation, the binding affinities of a series of antithrombin ligands have been calculated by using the two MM/GBSA rescoring methods. The correlation coefficient (R^2) of calculated and experimental binding free energies has been improved from 0.36 (for single-structure MM/GBSA rescoring) to 0.69 (for ensemble-average one). Decomposition of the calculated binding free energy reveals the electrostatic interactions in both solute and solvent play an important role in determining the binding free energy. The increasing negative charge of the compounds provides a more favorable electrostatic energy change but creates a higher penalty for the solvation free energy. Such a penalty is compensated by the electrostatic energy change, which results in a better binding affinity. The best binder has the highest ligand efficiency.

Keywords: MM/GBSA, Molecular dynamics, Binding Affinity, Antithrombin, Heparin.

1 Introduction

High-throughput virtual screening is an important tool for computer-aided drug discovery. We have developed a high throughput virtual screening pipeline for *in-silico* screening of a virtual compound database using high performance computing (HPC) [1]. The previous pipeline consists of four modules: receptor/target preparation, ligand preparation, VinaLC docking calculation [2], and single-structure MM/GBSA rescoring. All modules are parallelized to exploit typical cluster-type supercomputers. The MM/GBSA method is selected for rescoring because it is the fastest force-field based method that computes the free energy of binding, as compared to the other computational free energy methods, such as free energy perturbation (FEP) or thermodynamic

* Corresponding author.

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 584–593, 2015.

integration (TI) methods [3]. The MM/GBSA method has been widely exploited in free energy calculations [4, 5]. One of the notable features of this pipeline is an automated receptor preparation scheme with unsupervised binding site identification, which enables automatically running the whole pipeline with little or no human intervention. Perez-Sanchez and co-workers have developed a similar approach to improve drug discovery using massively parallel GPU hardware instead of supercomputers [6]. Their GPU-based program, BINDSURF [7], takes advantage of massively parallel and high arithmetic intensity of GPUs to speed-up the calculation in low cost desktop machine.

In this study, we have extended our pipeline from single-structure to ensemble-average MM/GSBA rescoring. To validate the new approach, we have gathered a panel of antithrombin ligands (Figure 1), including heparin and non-polysaccharide scaffold compounds. For the purpose of comparison, both single-structure and ensemble-average MM/GSBA rescoring are employed in the binding affinity calculations of antithrombin ligands. We must point out that estimation/calculations of entropy term are tricky. In most scenarios, the entropy term is neglected in the calculation for relative free binding energies. Quite a few researchers dispute the benefits of including the entropy term, which can be a major source of error due to the drawback of the entropy calculation method [8, 9], despite others who advocate its usage [10]. We choose to neglect the entropy term in our calculations.

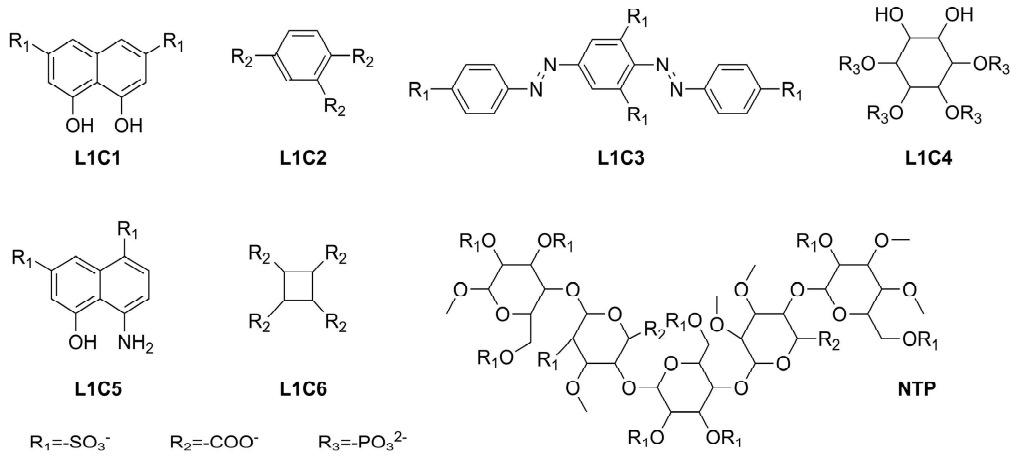


Fig. 1. Compounds targeting antithrombin. Compound NTP is a synthetic pentasaccharide compound from the crystal structure (PDB ID: 1AZX).

Antithrombin is a glycoprotein that plays a crucial role in the regulation of blood coagulation by inactivating several enzymes of the coagulation system and, thus, is an important drug target for the anticoagulant treatment. Antithrombin has two major isoforms, α and β, in the blood circulation [11]. α-Antithrombin is the dominant form of antithrombin and consists of 432 amino acids with 4 glycosylation sites, where an oligosaccharide occupyies each glycosylation site [12]. Heparin is the first compound that was identified and used as an anticoagulant and antithrombotic agent. It is a sulfated polysaccharide containing a specific pentasaccharide fragment (Figure 1, NTP)

that binds and activates the antithrombin [13]. This binding localized the function of antithrombin to inhibition of serine proteases in the coagulation cascade in the bloodstream, which allows coagulant activity in damaged tissue outside the vascular system [12].

Due to increasing interests in clinical applications, computational studies have been carried out to investigate the structure and behavior of antithrombin. Verli and co-workers performed molecular dynamics simulations to study the induced-fit mechanism of the antithrombin-heparin interaction and effects of glycosylation on heparin binding [14, 15]. Several detailed conformational changes associated with heparin binding to antithrombin were revealed. They also confirmed an intermediate state between the native and activated forms of antithrombin. Because of the weak surface complementarity and the high charge density of the sulfated sugar chain, the docking of heparin to its protein partners presents a challenging task for computational docking. Wade and Bitomsky developed a protocol that can predict the heparin binding site correctly [16]. Navarro-Fernandez and colleagues screened a large database *in silico* by using FlexScreen docking program [17] and identified a new, non-polysaccharide scaffold able to interact with the heparin binding domain of antithrombin [18]. They predicted D-myo-inositol 3,4,5,6-tetrakisphosphate (Figure1, L1C4) to strongly interact with antithrombin, which was confirmed by experimental binding affinity study.

Here, we carried out molecular dynamics (MD) simulations of ligand recognition upon binding antithrombin and calculate the ligand binding affinity, using the ensemble-average MM/GBSA rescoring method, as a complementary study to previous docking works [16, 18]. The advantage of long-time MD simulations is that more configurational space can be explored and dynamical properties of systems can be revealed.

2 Method

The initial structure for the antithrombin complex with Compound NTP is obtained from the PDB bank (PDB ID: 1AZX). The initial structures for the antithrombin complex with the other 6 ligands (Figure 1) are obtained from the FlexScreen docking program [17]. The MM/GBSA calculations are applied to these initial structures by using our in-house developed pipeline [1, 2] and Amber molecular simulation package [19]. The Amber forcefield f99SB [19] is employed in the calculation for the antithrombin receptor. Ligands use the Amber GAFF forcefield [20] as determined by the Antechamber program [21] in the Amber package. Partial charges of ligands are calculated using the AM1-BCC method [22]. The fourth module of the pipeline is employed for the single-structure MM/GBSA calculation, where the receptor-ligand complexes are energetically minimized by the MM/GBSA method implemented in the Sander program of the Amber package [23]. The atomic radii developed by Onufriev and coworkers (Amber input parameter igb=5) are chosen for all GB calculations [24]. For the ensemble-average MM/GBSA rescoring, energetically minimized structures from single-structure MM/GBSA rescoring are served as initial structures. The systems are heated from 0 K to room temperature, 300 K. The MD simulations

with a time step of 2 fs for the integration of the equations of motion are carried out at room temperature. The systems are equilibrated at room temperature for 500 ps. Each MD trajectory is followed to 100 ns after equilibrium. Binding affinities of antithrombin and its 7 ligands are calculated by post-processing the ensembles of structures extracted from MD trajectories using MM/GBSA calculations. In the MM/GBSA calculation, the binding free energy between a receptor and a ligand is calculated using the following equations:

$$\Delta G_{\text{bind}} = G_{\text{complex}} - G_{\text{receptor}} - G_{\text{ligand}} \quad (1)$$

$$\Delta G_{\text{bind}} = \Delta H - T\Delta S \approx \Delta E_{\text{gas}} + \Delta G_{\text{sol}} - T\Delta S \quad (2)$$

$$\Delta E_{\text{gas}} = \Delta E_{\text{int}} + \Delta E_{\text{ELE}} + \Delta E_{\text{VDW}} \quad (3)$$

$$\Delta G_{\text{sol}} = \Delta G_{\text{GB}} + \Delta G_{\text{Surf}} \quad (4)$$

The binding free energy (ΔG_{bind}) is decomposed into different energy terms. Because the structures of complex, receptor, and ligand are extracted from the same trajectory, the internal energy change (ΔE_{int}) is canceled. Thus, the gas-phase interaction energy (ΔE_{gas}) between the receptor and the ligand is the sum of electrostatic (ΔE_{ELE}) and van der Waals (ΔE_{VDW}) interaction energies. The solvation free energy (ΔG_{sol}) is divided into the polar and non-polar energy terms. The polar solvation energy (ΔG_{GB}) is calculated by using the GB model. The non-polar contribution is calculated based on the solvent-accessible surface area (ΔG_{Surf}). A value of 80 is used for the solvent dielectric constant, and the solute dielectric constant is set to 1. The calculated binding free energy (ΔG_{bind}) is the sum of the gas-phase interaction energy and solvation free energy because we neglect the entropy term. The experimental binding free energy is estimated from the experimental dissociation constant (Kd) by the equation:

$$\Delta G_{\text{Exp}} = \text{RT} \cdot \ln(\text{Kd}) \quad (5)$$

where R is the gas constant, and T is the temperature.

3 Results and Discussion

The calculated binding free energies of seven antithrombin ligands using the ensemble-average MM/GBSA rescoring are shown in Table 1 together with their corresponding experimental values. Each calculated binding free energy is averaged from snapshots extracted from 100 ns MD trajectories. Except for Compound L1C1, all the antithrombin ligands have experimental binding free energies. As determined experimentally, Compound L1C4 is the best binder with a Kd value of 0.088 uM [18]. As predicted by the MM/GBSA method, Compound L1C4 has the most negative binding free energy (-308.01 kcal/mol), which is in agreement with the experimental results. The second best binder as predicted by the MM/GBSA calculation is Compound NTP with a calculated binding free energy of -279.57 kcal/mol, confirming the experimental ranking relative

to Compound L1C4. Compound L1C2 is predicted to have the worst binding free energy of the six ligands, which is also in agreement with its experimental ranking value. In summary, the MM/GBSA calculations rank the binding affinities of all six antithrombin ligands in same exact order as that of experimental binding free energy rankings.

Table 1. Calculated and experimental binding free energies (kcal/mol) of antithrombin ligands

Cmpd	ΔE_{ELE}	ΔE_{VDW}	ΔE_{gas}	ΔG_{Surf}	ΔG_{GB}	ΔG_{GB-ELE}	ΔG_{Sol}	ΔG_{bind}	Kd(uM)	$\Delta G_{Exp.}$
L1C1	-552.67	-23.68	-576.35	-2.75	480.12	-72.55	477.37	-98.97	-	-
L1C2	-442.99	0.47	-442.52	-1.20	417.60	-25.39	416.41	-26.11	13700	-2.54
L1C3	-836.77	-39.96	-876.73	-4.06	781.94	-54.83	777.88	-98.85	10.02	-6.81
L1C4	-1599.09	33.02	-1566.07	-2.98	1261.05	-338.04	1258.07	-308.01	0.088	-9.62
L1C5	-613.30	-19.00	-632.31	-2.57	525.82	-87.48	523.25	-109.06	0.69	-8.40
L1C6	-818.73	8.21	-810.52	-1.54	752.94	-65.79	751.41	-59.11	17.52	-6.48
NTP	-2598.87	-60.89	-2659.76	-7.58	2387.77	-211.09	2380.20	-279.57	0.104	-9.52

The calculated binding free energies of six antithrombin ligands using the ensemble-average MM/GBSA rescoring have been plotted against the free energies derived from experimental dissociation constants. The correlation coefficient (R2) is 0.69, which indicates good correlation between the calculated and experimental values (Figure 2). By comparison, the correlation coefficient calculated by single-structure MM/GBSA is only 0.36, and Compound NTP is predicted to be the best binder, instead of Compound L1C4. Thus, using the ensemble-average MM/GBSA rescoring method significantly improves the accuracy of the prediction over the single-structure MM/GBSA rescoring.

As shown in Figure 1, all antithrombin ligands contain negatively charged groups, suggesting electrostatic interactions should be a key factor in the binding affinity. Compound NTP has a total charge of -11, and Compound L1C4 has a total charge of -8. By decomposing the binding free energy, Compound NTP and L1C4 have the largest electrostatic energy changes upon binding in both gas phase (ΔE_{ELE}) and GB solvent ($\Delta G_{GB\text{-}ELE}$). The energy change upon binding in gas phase is equivalent to the energy change upon binding for the solute. Thus, in other words, Compound NTP and L1C4 have the largest electrostatic energy changes upon binding in solute and solvent. In contrast, Compound L1C2 has the smallest electrostatic energy changes in solute and solvent. Although Compound L1C4 has the least favorable van der Waals energy change upon binding, the electrostatic energy

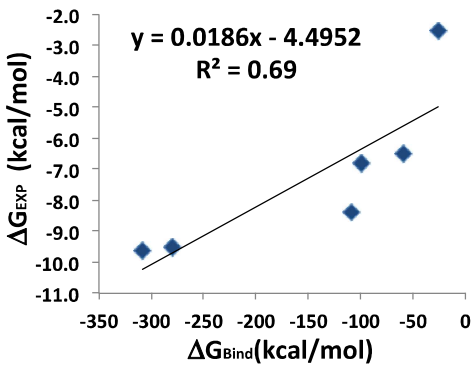


Fig. 2. The scatter plot of calculated MM/GBSA binding free energy versus experimental binding affinity estimated from dissociation constant

change compensates significantly. For all ligands, the van der Waals energy changes (ΔE_{VDW}) upon binding are less than the electrostatic energy changes (ΔE_{ELE}) by 1-2 orders of magnitude. The contribution of the van der Waals energy change has been overpowered by the electrostatic energy change. Non-polar contribution of solvation free energy of Compound NTP and L1C3 are more negative than that of the other compounds because the sizes of Compound NTP and L1C3 are larger than the other compounds. Nevertheless, non-polar contributions for all compounds are small. The non-polar contribution is overwhelmed by the polar contribution of solvation free energy. Thus, the two major factors to determine the binding affinity are the electrostatic energy change and solvation free energy change. The larger the total charge of the compound, the larger the penalty cost is for solvation free energy. However, high penalty for large total charge of compound has been paid by the large favorable electrostatic energy changes. Although the electrostatic energy change of Compound L1C4 is less than that of Compound NTP, Compound L1C4 needs less compensation for the solvation free energy. Thus, Compound L1C4 is a better binder than Compound NTP.

Hydrogen bonding analysis determines the numbers of hydrogen bonds to antithrombin that are persistent at >20% of the time. Compound NTP has 40 hydrogen bonds to antithrombin, while L1C4 has 25 hydrogen bonds. For Compounds L1C1, L1C2, L1C3, L1C5, and L1C6, that number of hydrogen bonds are 5, 10, 12, 12, and 12, respectively. Taking the molecular weight into account and using a similar approach as Reynolds' ligand efficiency method [25], Compound L1C4 has the highest ligand efficiency.

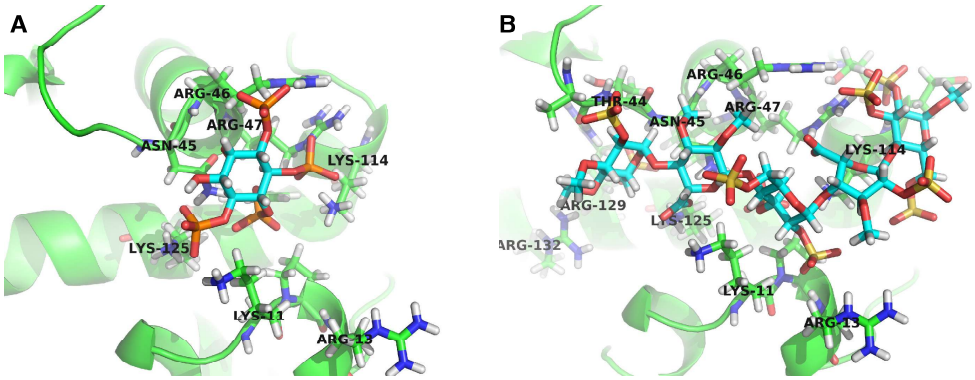


Fig. 3. Initial structures of Compounds L1C4 (A) and NTP (B) complexed with antithrombin

Compound L1C4 forms double hydrogen bonds with Arg47 (Figure 3A). One hydrogen bond (O6-HH21-NH2) has 94.81% persistence, and the other one (O6-HE-NE) has 89.55%. The average hydrogen bond distances between the heavy atoms are 2.74 Å and 2.70 Å, respectively. Compound L1C4 has strong hydrogen bonds with Arg47, and one of the four phosphate groups from Compound L1C4 is locked to the Arg47. According to the hydrogen bonding analysis, Compound L1C4 is also hydrogen bonded to Arg46, Arg13, Lys114, Lys11, Lys125, and Asn45, which are key

residues to the binding process. We find that the binding of Compound L1C4 to antithrombin is non-specific. Except for the phosphate group locked to Arg47, the other three phosphate groups of Compound L1C4 can rotate so that key residues can form hydrogen bonds to different oxygen atoms of phosphate at different times during the MD trajectory. Notably, Arg13 starts far away from Compound L1C4 in the initial conformation. After 8 ns of MD simulation, Arg13 begins to make hydrogen bonds with the phosphate group of Compound L1C4, which suggests that long-time MD simulations are essential to obtaining accurate binding affinities. As shown in Figure 3B, Compound NTP makes hydrogen bonds to antithrombin mainly via its negatively charged sulfate groups. Compound NTP forms high persistent hydrogen bonds with Arg13, Arg129, Arg47, and Asn45 (70~88%) and forms medium persistent hydrogen bonds with Arg132, Lys125, and Thr44 (43~66%). Only relatively weak hydrogen bonds are observed with Arg46, LYS114 and LYS11.

Navarro-Fernandez and colleagues have also predicted the interacting residues by using FlexScreen scoring function [17]. For compound L1C4, they identified Lys11, Asn45, Arg46, Arg47, Lys114, and Lys125 as key residues, but Arg13 is missing from their list. For the Compound L1C4 docking calculation, they used the receptor structure from the X-ray crystal structure of antithrombin complexed with Compound NTP. Thus, the initial receptor structure for Compound L1C4 docking is biased. Docking calculations usually hold the receptor protein rigid. A few docking programs are able to have set side-chains of key residues in receptor as flexible. However, most docking programs cannot sample the larger configuration space for the whole ligand-receptor complex. From our MD simulations, we find that Arg13 is quite flexible and can be adjusted to accommodate both large (e.g. Compound NTP) and small (e.g. Compound L1C4) compounds, which shows the advantages of using MD simulations over simple docking calculations.

Judging from the hydrogen bond analysis on Compound L1C4 and NTP, Arg47, Arg13, and Asn45 play crucial roles in the antithrombin binding process. Antithrombin provides multiple sulfate/phosphate binding sites, consisting of mostly positively charged residues (arginine, lysine) and neutral charged residues that can provide rich hydrogen bond donors/acceptors (asparagine). All four phosphate groups of Compound L1C4 form hydrogen bonds with antithrombin while not all the sulfate groups of Compound NTP can form hydrogen bonds with antithrombin. As pointed out above, introducing positively charged group in the ligand will result in a penalty for solvation free energy. If adding a positively charged group cannot form favorable interactions (e.g. hydrogen bonding), ligand efficiency will be reduced, explaining why Compound L1C4 has higher ligand efficiency than Compound NTP.

Comparing the results from single-structure and ensemble-average MM/GBSA rescoring, the latter yields more accurate results. The ensemble-average MM/GBSA rescoring ranks the binding affinities of antithrombin ligands in the order that agrees with the experimental results. The advantage of ensemble-average MM/GBSA rescoring is that the binding affinity is averaged from an ensemble of structures extracted from long-time MD simulations. Long-time MD simulations can explore more configuration space and find energetically favorable configurations, which could offset the bias of initial structures. This can be verified in the MD trajectory of Compound

L1C4. Arg13 is observed to form hydrogen bonds with the phosphate group of Compound L1C4 after 8 ns of MD simulation.

The ensemble-average MM/GBSA rescoring method is relatively accurate compared to single-structure MM/GBSA. However, the method to obtain an ensemble of structures, long-time MD simulations, is computationally intensive. Since the drug-like virtual libraries often contain millions of compounds, high-throughput virtual screening could be very costly, if using a more accurate but expensive method at the very beginning of the screening. To bridge the gap, our virtual screening pipeline uses a down-select scheme to screen large virtual compound libraries. A standard procedure to run the pipeline is to down-select compounds after they pass each screening method as implemented in the pipeline. The first screening method in the pipeline is VinaLC docking, which can dock one million compounds in 1.4 h on about 15K CPUs [2]. Top ranked poses of down-selected compounds after docking are rescored using a single-structure MM/GBSA rescoring method. Finally, the most expensive ensemble-average MM/GBSA rescoring method in the pipeline can be applied to an amenable number of compounds down-selected after single-structure MM/GBSA rescoring, providing the accuracy needed for a fewer number of compounds.

4 Conclusion

In this article, we introduce a new addition, ensemble-average MM/GBSA rescoring, to our virtual screening pipeline. As a proof of concept, we calculated the binding affinities of seven antithrombin ligands by employing the previous single-structure MM/GBSA rescoring method and newly developed ensemble-average MM/GBSA rescoring method. The correlation coefficient of calculated and experimental binding affinities is improved from 0.36 to 0.69 when using ensemble-average MM/GBSA rescoring. The rank order of calculated binding free energies using ensemble-average MM/GBSA rescoring exactly matches the experimentally derived free energies. We demonstrate that long-time MD trajectory can explore more configuration space and find energetically favorable configurations so that it can offset the bias of initial structures and improve the accuracy of binding affinity prediction. The electrostatic interactions in both solute and solvent contribute favorably to the binding free energy. Adding more negatively charged groups to the ligand provides more favorable electrostatic energy change. However, it creates a higher penalty for the solvation free energy simultaneously. The penalty can be compensated for by forming more hydrogen bonds as more negatively charged groups are added into the ligand. The negatively charge groups added to the ligand must actively interact with receptor by forming hydrogen bonds to achieve high ligand efficiency. Compound L1C4 has higher ligand efficiency because it uses all its phosphate groups to form hydrogen bonds with antithrombin while Compound NTP does not.

Acknowledgements. The authors thank Livermore Computing for the computer time and Laboratory Directed Research and Development for funding (12-SI-004). We also thank Livermore Computing Grand Challenge for extensive computing resources.

This work was performed under the auspices of the United States Department of Energy by the Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Release Number LLNL-JRNL-666361. This work was partially supported by the Fundación Séneca la Región de Murcia under Project 18946/JLI/13. This work has been funded by the Nils Coordinated Mobility under grant 012-ABEL-CM-2014A, in part financed by the European Regional Development Fund (ERDF).

References

1. Zhang, X., Wong, S.E., Lightstone, F.C.: Toward Fully Automated High Performance Computing Drug Discovery: A Massively Parallel Virtual Screening Pipeline for Docking and Molecular Mechanics/Generalized Born Surface Area Rescoring to Improve Enrichment. J. Chem. Inf. Model. 54, 324–337 (2014)
2. Zhang, X., Wong, S.E., Lightstone, F.C.: Message passing interface and multithreading hybrid for parallel molecular docking of large databases on petascale high performance computing machines. J. Comput. Chem. 34, 915–927 (2013)
3. Beveridge, D.L., DiCapua, F.M.: Free Energy Via Molecular Simulation: Applications to Chemical and Biomolecular Systems. Annu. Rev. Biophys. Biophys. Chem. 18, 431–492 (1989)
4. Kollman, P.A., Massova, I., Reyes, C., Kuhn, B., Huo, S., Chong, L., Lee, M., Lee, T., Duan, Y., Wang, W., Donini, O., Cieplak, P., Srinivasan, J., Case, D.A., Cheatham, T.E.: Calculating Structures and Free Energies of Complex Molecules: Combining Molecular Mechanics and Continuum Models. Acc. Chem. Res. 33, 889–897 (2000)
5. Hou, T.J., Wang, J.M., Li, Y.Y., Wang, W.: Assessing the Performance of the MM/PBSA and MM/GBSA Methods. 1. The Accuracy of Binding Free Energy Calculations Based on Molecular Dynamics Simulations. J. Chem. Inf. Model. 51, 69–82 (2011)
6. Pérez-Sánchez, H., Cano, G., García-Rodríguez, J.: Improving drug discovery using hybrid softcomputing methods. Applied Soft Computing 20, 119–126 (2014)
7. Sanchez-Linares, I., Perez-Sanchez, H., Cecilia, J., Garcia, J.: High-Throughput parallel blind Virtual Screening using BINDSURF. BMC Bioinformatics 13, S13 (2012)
8. Rastelli, G., Rio, A.D., Degliesposti, G., Sgobba, M.: Fast and accurate predictions of binding free energies using MM-PBSA and MM-GBSA. J. Comput. Chem. 31, 797–810 (2010)
9. Greenidge, P.A., Kramer, C., Mozziconacci, J.C., Wolf, R.M.: MM/GBSA Binding Energy Prediction on the PDBbind Data Set: Successes, Failures, and Directions for Further Improvement. J. Chem. Inf. Model. 53, 201–209 (2013)
10. Lafont, V., Armstrong, A.A., Ohtaka, H., Kiso, Y., Mario Amzel, L., Freire, E.: Compensating Enthalpic and Entropic Changes Hinder Binding Affinity Optimization. Chemical Biology & Drug Design 69, 413–422 (2007)
11. Turko, I.V., Fan, B., Gettins, P.G.W.: Carbohydrate isoforms of antithrombin variant N135Q with different heparin affinities. FEBS Lett. 335, 9–12
12. Jin, L., Abrahams, J.P., Skinner, R., Petitou, M., Pike, R.N., Carrell, R.W.: The anticoagulant activation of antithrombin by heparin. Proc. Natl. Acad. Sci. U. S. A. 94, 14683–14688 (1997)
13. Thunberg, L., Bäckström, G., Lindahl, U.: Further characterization of the antithrombin-binding sequence in heparin. Carbohydr. Res. 100, 393–410 (1982)

14. Verli, H., Guimarães, J.A.: Insights into the induced fit mechanism in antithrombin–heparin interaction using molecular dynamics simulations. J. Mol. Graph. Model. 24, 203–212 (2005)
15. Pol-Fachin, L., Franco Becker, C., Almeida Guimarães, J., Verli, H.: Effects of glycosylation on heparin binding and antithrombin activation by heparin. Proteins: Structure, Function, and Bioinformatics 79, 2735–2745 (2011)
16. Bitomsky, W., Wade, R.C.: Docking of Glycosaminoglycans to Heparin-Binding Proteins: Validation for aFGF, bFGF, and Antithrombin and Application to IL-8. J. Am. Chem. Soc. 121, 3004–3013 (1999)
17. Merlitz, H., Wenzel, W.: Comparison of stochastic optimization methods for receptor–ligand docking. Chem. Phys. Lett. 362, 271–277 (2002)
18. Navarro-Fernández, J., Pérez-Sánchez, H., Martínez-Martínez, I., Meliciani, I., Guerrero, J.A., Vicente, V., Corral, J., Wenzel, W.: In Silico Discovery of a Compound with Nanomolar Affinity to Antithrombin Causing Partial Activation and Increased Heparin Affinity. J. Med. Chem. 55, 6403–6412 (2012)
19. Ponder, J.W., Case, D.A.: Force fields for protein simulations. Protein Simulations 66, 27–85 (2003)
20. Wang, J.M., Wolf, R.M., Caldwell, J.W., Kollman, P.A., Case, D.A.: Development and testing of a general amber force field. J. Comput. Chem. 25, 1157–1174 (2004)
21. Wang, J., Wang, W., Kollman, P.A., Case, D.A.: Automatic atom type and bond type perception in molecular mechanical calculations. J. Mol. Graph. Model. 25, 247–260 (2006)
22. Jakalian, A., Bush, B.L., Jack, D.B., Bayly, C.I.: Fast, efficient generation of high-quality atomic Charges. AM1-BCC model: I. Method. J. Comput. Chem. 21, 132–146 (2000)
23. Case, D.A., Cheatham III, T.E., Darden, T., Gohlke, H., Luo, R., Merz Jr., K.M., Onufriev, A., Simmerling, C., Wang, B., Woods, R.J.: The Amber biomolecular simulation programs. J. Comput. Chem. 26, 1668–1688 (2005)
24. Onufriev, A., Bashford, D., Case, D.A.: Exploring protein native states and large-scale conformational changes with a modified generalized born model. Proteins-Struct. Funct. Bioinf. 55, 383–394 (2004)
25. Reynolds, C.H., Tounge, B.A., Bembenek, S.D.: Ligand binding efficiency: Trends, physical basis, and implications. J. Med. Chem. 51, 2432–2438 (2008)

Predicting Atherosclerotic Plaque Location in an Iliac Bifurcation Using a Hybrid CFD/Biomechanical Approach

Mona Alimohammadi[1,*], Cesar Pichardo-Almarza[1,2], Giulia Di Tomaso[1], Stavroula Balabani[1], Obiekezie Agu[3], and Vanessa Diaz-Zuccarini[1]

[1]UCL Mechanical Engineering, Multiscale Cardiovascular Engineering Group (MUSE), Torrington Place, WC1E 7JE. London, UK
[2]Xenologiq Ltd, Canterbury, CT2 7FG, UK
[3]University College Hospital, Vascular Unit, NW1 2BU, London, UK
mona.alimohammadi.10@ucl.ac.uk

Abstract. Experimental evidence indicates that haemodynamic stimuli influence some properties of the arterial endothelium, such as cell geometry and permeability, leading to possible accumulation of blood-borne macromolecules and initiation of atherosclerosis. Patient-specific computational models are able to capture complex haemodynamic characteristics to explore and analyse the development of these diseases *in silico*. Patient-specific models are particularly beneficial in the case of aortic dissection (AD), a condition in which the aortic wall is split in two, creating a true and a false lumen. In this condition, the proportion of blood through the main vessel and the main aortic branches is substantially modified and malperfusion (lack of blood supply) of the downstream vessels is often observed. Furthermore, AD alters the haemodynamics downstream of the lesion, potentially leading to the formation of atherosclerotic plaques at the iliac bifurcation. In order to correctly approximate the haemodynamic changes and analyse the role they play in the development of atherosclerosis formations in AD patients, a combined multiscale methodology is required. In this study, both, blood flow through an iliac bifurcation of a patient suffering from type-B aortic dissection and endothelium behavior are analysed, in order to investigate atherosclerosis formation.

1 Introduction

Cardiovascular diseases are the leading cause of death in developed countries [1]. Thoracic aortic diseases occur between 16 to 10 per 100000 per year for both men and women in Europe [2], among which, aortic dissection (AD), and its associated vascular diseases and further complications, are relatively poorly understood. One of these complication is calcification of the aortic wall, which carries with it significant risks and increased mortality rates [3, 4]. AD is initiated by the formation of a cleavage in the intimal layer of the aortic vessel wall, allowing blood to enter the vessel wall,

* Corresponding author.

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 594–606, 2015.

forming a pathway between the layers and creating a so called false lumen (FL), as opposed to the true lumen (TL). AD is highly prevalent in patients with hereditary connective tissue disorders such as Ehler-Danlos and Marfan syndrome [1, 5, 6], which would potentially affect the aortic wall distensibility, stiffness and tissue fragility [7, 8]. It has been reported that 31% of patients suffering from AD have a history of plaque formation [9] and atherosclerosis has been shown to be a major post-operative factor in the mortality rate for dissections involving the descending part of the aorta (Stanford type-B) [10] (as opposed to the ascending aorta: Stanford type-A). Calcification is present in both the TL and FL, as well as downstream of the dissected region [11, 12]. The dissection causes a pathological haemodynamic environment, prone to calcification of the vessel wall and atherosclerotic plaque formation downstream of the dissected thoracic aorta at the iliac bifurcation.

The process of atherosclerotic plaque formation is complex, multifactorial and systemic. Experimental evidence indicates that haemodynamic stimuli influence some properties of the arterial endothelium, such as cell geometry and permeability, leading to possible accumulation of blood-borne macromolecules (e.g. Low Density Lipoproteins – LDL) and initiation of atherosclerosis. In the early stages of the disease, an accumulation of lipid-laden macrophages (foam cells) is observed in the subendothelium. As the disease develops, smooth muscles cells and fibrous tissue accumulate. Formation of lesions is promoted by plasma proteins carrying elevated levels of cholesterol and triglycerides. Clinical manifestations of atherosclerosis, including coronary artery disease, cerebrovascular disease, and peripheral arterial disease, occur in 2 of 3 men and 1 in 2 women over the age of 40 [13].

The management of these combined, complex conditions proves to be challenging for clinicians. Patient-specific computational models are able to capture complex haemodynamic characteristics to explore and analyse the development of these diseases *in silico*. Patient-specific models are particularly beneficial in the case of AD, as the proportion of blood through the aorta and the aortic branches is substantially modified and malperfusion (lack of blood supply) of the downstream vessels is often observed [5, 14]. AD also alters the haemodynamics downstream of the lesion, potentially leading to the formation of atherosclerotic plaques at the iliac bifurcation. To correctly approximate the haemodynamic changes and analyse the role they play in the development of atherosclerosis formation in AD patients, a combined multiscale methodology is required. Several studies have been carried out to analyse haemodynamic parameters such as pressure, flow and wall shear stress (WSS) through the iliac arteries [15, 16]. Kim et al. [17] used a lagrangian method for the boundary conditions (BCs) for example, while others have used electrical analogues to represent the characteristics of the downstream and upstream vasculature in dissected aortae.

In this study, simulation of the blood flow through an iliac bifurcation of a patient suffering from type-B aortic dissection is used to estimate locations of atherogenesis using a mathematical model of atherosclerosis which includes the modelling of LDL transport from the lumen into the arterial wall (through the endothelium) using a three-pore modelling approach [18] and based in previous work [19, 20].

In order to capture detailed haemodynamic values that will affect the formation of plaque in individual patients, realistic fluid simulations are required and this requires

appropriate model definition. In this work, the pressure and flow rate at each boundary change during the cardiac cycle, a feature that is captured by the simulation by using dynamic boundary conditions. Coupling the three-dimensional (3D) domain with Windkessel models is an approach that can account for the interdependent time varying flow and pressure values along the patient's cardiac cycle [21, 22]. The challenge in applying this approach is the estimation of the Windkessel parameters (resistance or compliance) and there is no agreed methodology on how to select these [23].

In this paper, a numerical simulation to investigate the possibility of plaque formation in an individual patient is produced. The blood flow through a patient-specific iliac bifurcation is simulated by coupling the 3D domain to the three element Windkessel models at each of the (3D) domain's outlets. The Windkessel parameters are tuned using the invasive pressure measurements acquired on the same patient prior to the simulation. These results are then coupled to an endothelial permeability model, as explained in Sections 2 and 3 in order to predict locations of atherosclerosis, using a 'virtual follow-up' approach.

2 Methods

2.1 Geometry

An iliac bifurcation from a 54 year-old female patient suffering from type-B dissection and atherosclerosis was segmented from MSCT images (Ethics Reference number 13/EM/0143), obtained at University College Hospital (UCH). The 3D geometry was reconstructed from approximately 200 CT slices using ScanIP (Simpleware Ltd. UK). In order to segment the region of interest (the fluid domain), multiple thresholds and flood fills were applied to select the iliac arteries (IAs). Dilatation was used for ensure all pixels within the domain were selected. In order to miminise pixelisation artefacts, the final mask was smoothed using recursive Gaussian and median filters. Existing atherosclerotic formations observed in the images (identified by particularly bright regions at the vessel walls) were virtually removed to obtain a patent lumen. This geometry was then used to predict locations of atherosclerosis in the patient, using a 'virtual follow-up' approach. The final geometry can be seen in Figure 1a, which shows the flow inlet at the distal abdominal aorta (DA) and four outlets of left and right, internal and external IAs (R_{ext}, R_{int}, L_{ext}, L_{int}). Figure 1b shows the atherosclerotic plaque visible in the lumen (upper panel) and segmented using ScanIP. The blue and yellow mask (lower panel), representing the lumen and plaque were added together to create a single lumen (virtually removing the plaque regions). All boundaries were cropped to provide flat surfaces.

2.2 Computational Fluid Dynamics

Blood was considered to be an incompressible fluid with a density of 1056 kg m^{-3}. To model the non-Newtonian properties of the blood flow, the Carreau-Yasuda model was used with parameters defined by Gijsen et al. [24]. The flow was considered to be laminar. The pressure wave at the DA from a previous study on the same patient [22]

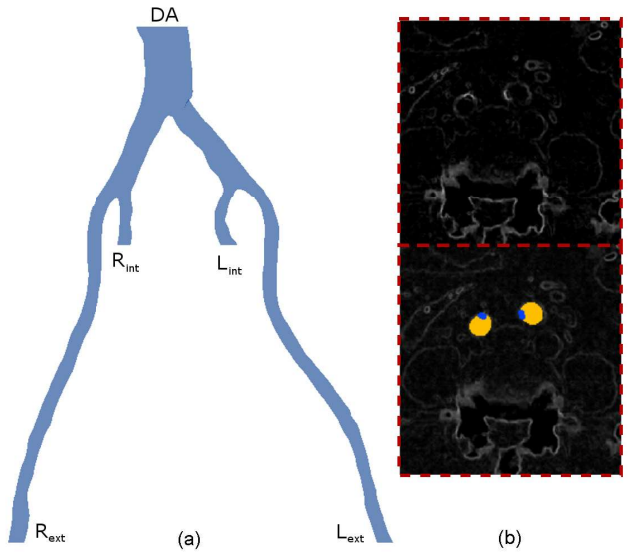


Fig. 1. (a) Geometry of the iliac bifurcation. (b) CT scan slices – upper panel: raw, lower panel: showing lumina (yellow) and atherosclerotic plaques (blue).

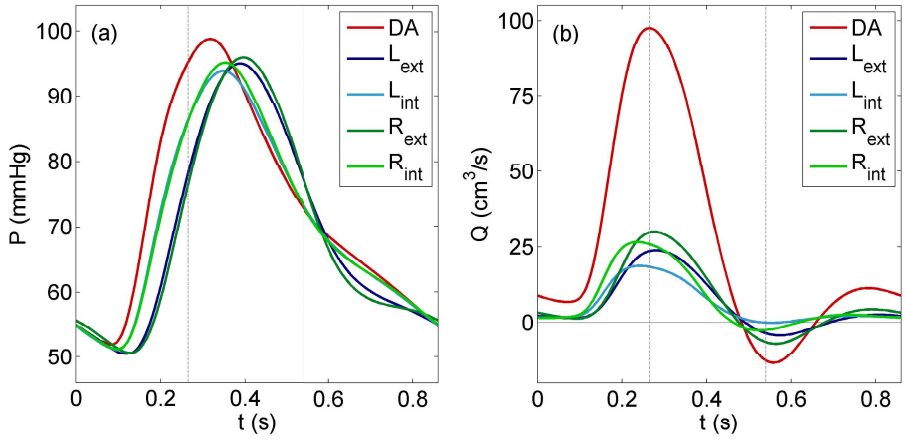


Fig. 2. (a) Pressure waves at each of the 3D domain outlets. (b) Flow waves at each of the 3D domain outlets. Vertical grey lines show time points considered in following figures.

(Fig. 2a) was used as inlet boundary condition. A three-element Windkessel model was coupled to each of the outlets. The zero-dimensional Windkessel model uses a circuit analogy, comparing pressure and flow to voltage and current, respectively. Therefore, solving the equation for a circuit of a parallel resistor (R_2) and capacitor (C) in series with a resistor (R_1) yields:

$$P = (R_1 + R_2)Q - \frac{R_2 C dp}{dt} + \frac{R_1 R_2 dQ}{dt} \quad (1)$$

The derivative terms are calculated using the backward Euler method. The vessel wall was assumed to be rigid with no slip condition and a time step of 5 *ms* was used for the simulation.

In order to define the parameters of the Windkessel models, a tuning methodology was used to match invasive pressure measurements, which were available 3 *cm* upstream and 20 *cm* downstream, of the iliac bifurcation. Detailed analysis of this methodology was reported in a previous study[22].

2.3 Atherogenesis

A mathematical model describing the processes related to the early stages of atherosclerosis was implemented. Here, low density lipoprotein (LDL) is transported from the artery lumen into the arterial wall, taking into account the effects of local wall shear stress (WSS) on the endothelial cell layer and its pathways of volume and solute flux (see [19] and [20] for more details). The endothelium is described with a three-pore modelling approach, taking into consideration the contributions of the vesicular pathway, normal junctions, and leaky junctions. The fraction of leaky junctions is calculated as a function of the local WSS based on previous models and published experimental data [18, 19] and is used in conjunction with pore theory to determine the transport properties of this pathway.

It is assumed that the endothelium is influenced by the mechanical stimuli exerted by the blood flow. Experimental findings show that in areas of altered hemodynamics, endothelial cells do not have a typical cobblestone shape, but a more circular shape instead and exhibit increased permeability. The model uses a relationship between endothelial permeability and local WSS based on the Endothelial Cell Shape Index (ECSI) taken after experimental findings [25]. ECSI is related to the cellular shape and takes values from zero to one, i.e. a circle has an ECSI of one whilst a straight line has an ECSI of zero. The relationship between WSS and ECSI was modelled as shown in equation (2):

$$ECSI = 0.380e^{-0.3537\,WSS} + 0.225e^{-0.3537\,WSS} \tag{2}$$

To model the LDL transport through the endothelium, a modified version of the Kedem Ketchalsky's equations for membrane transport was used [19, 20]:

$$J_v = L_p(\Delta p_{\text{end}} - \sigma_d \Delta \Pi) \tag{3}$$

$$J_s = P_i\left(c_{lum} - c_{w,\text{end}}\right)\frac{Pe}{e^{Pe} - 1} + J_v(1-\sigma)\bar{c} \tag{4}$$

$$Pe = \frac{J_v(1-\sigma)}{P_i} \tag{5}$$

where J_v is the volumetric flux through the endothelium, L_p is the hydraulic conductivity, Δp_{end} is the pressure difference through the endothelium, σ_{d} is the osmotic

reflection coefficient and $\Delta\Pi$ is the osmotic pressure. The solute flux, J_s, can be divided into a convective component and a diffusive component (Equation 5). P_i is the diffusive permeability, Pe the modified Peclet number, $c_{w,\text{end}}$ the LDL concentration at the sub-endothelial layer and σ the solvent drag coefficient.

As previously mentioned, three main penetration pathways were considered: leaky junctions, normal junctions and vesicular pathways [19]. The model considers that the bulk volume flux through the endothelial membrane is given by:

$$J_v = J_{v,lj} + J_{v,nj} \tag{6}$$

where $J_{v,lj}$ is the flux through leaky junctions and $J_{v,nj}$ is the flux through normal junctions. Following the 'three-pore theory' [18], solute flux only occurs through endothelial leaky cell junctions and vesicles:

$$J_s = J_{s,lj} + J_{s,v} \tag{7}$$

where the solute flux through the vesicular pathway ($J_{s,v}$) is calculated as 10% of the solute flux through the leaky junction pathway ($J_{s,,lj}$) [18].

Leaky cells have high permeability to LDL, which can be linked to the magnitude of WSS acting on the endothelium. Experimental findings show that in areas of low WSS and high ECSI, the number of mitotic cells (MC) is increased, leading to [18]:

$$MC = 0.003739e^{14.75 \cdot ECSI} \tag{8}$$

Within the endothelium, it has been shown that the quantity of leaky mitotic cells is approximately 80.5% and since these represent approximately 45.3% of the total number of leaky cells (LC) in that area [26], the number of LC was calculated as:

$$LC = 0.307 + 0.805MC \tag{9}$$

The ratio of endothelium (ϕ) covered by LCs is calculated as:

$$\phi = \frac{LC\ \pi R_{cell}^2}{unit\ area} \tag{10}$$

where R_{cell} is the radius. Using these values, the transport properties of the endothelium can be determined. The total hydraulic conductivity of the endothelial leaky junctions ($L_{p,lj}$) is defined as:

$$L_{p,lj} = \phi \cdot L_{p,slj} \tag{11}$$

where $L_{p,slj}$ is the hydraulic conductivity of a single leaky junction, w and l_{lj} are the half-width (20 nm) and the length (2 μm) of the LJ [27, 28].

$$L_{p,sj} = \frac{w^2}{3\mu_p l_{lj}} \tag{12}$$

3 Results

Fig. 2 shows the pressure and flow waves at each of the domain boundaries for one cardiac cycle. Due to the dynamic boundary conditions, the pressure waves are out of phase, with the internal IAs lagging behind the DA, and the external IAs delayed further. Correspondingly, the flow rates are out of phase with one another, with the external IAs lagging behind the internal IAs. Note that the flow maxima occur prior to the pressure maxima. The maximum flow rate entering the domain at the DA is approximately in phase with the external IAs.

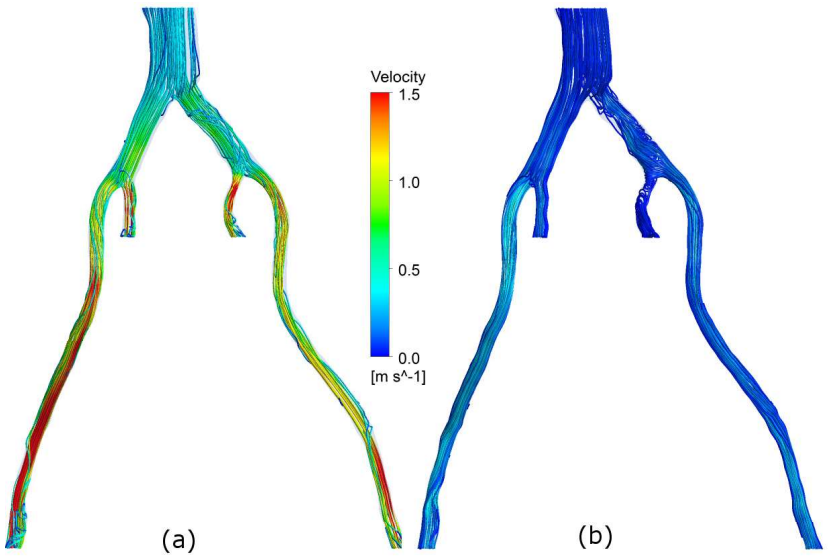


Fig. 3. Streamlines at (a) Peak systolic flow and (b) maximum negative flow

Fig. 3 shows the streamlines at two time points in the cardiac cycle; peak systolic flow and the point of minimum flow, as indicated by the dashed vertical lines in Fig. 2. At peak systole (Fig. 3a), the velocity is elevated after the first bifurcation and further increased after the second bifurcation through both L_{int} and R_{int} branches. A small region of disturbed flow can be seen at the curvature of the L_{int} branch (Fig. 3a). The velocity through the R_{ext} branch is marginally higher than the L_{ext} branch, and overall fairly uniform streamlines can be observed throughout the domain at peak systole. At the negative flow time-point (where the negative flow can be observed in Fig. 2b), the velocity is low throughout the domain, but is relatively uniform, in contrast to streamlines in the dissected thoracic aorta at a similar time point [22].

Fig. 4 shows pressure contours along the bifurcation at different time points. At peak systole (Fig. 4a), the pressure drop across the domain is ~ 25 mmHg. The pressure drops continuously throughout the domain. Small elevations in pressure at the wall can be observed at the stagnation points in each of the internal/external bifurcations. At the point of maximum negative flow (Fig. 4b), the pressure throughout the domain is fairly uniform.

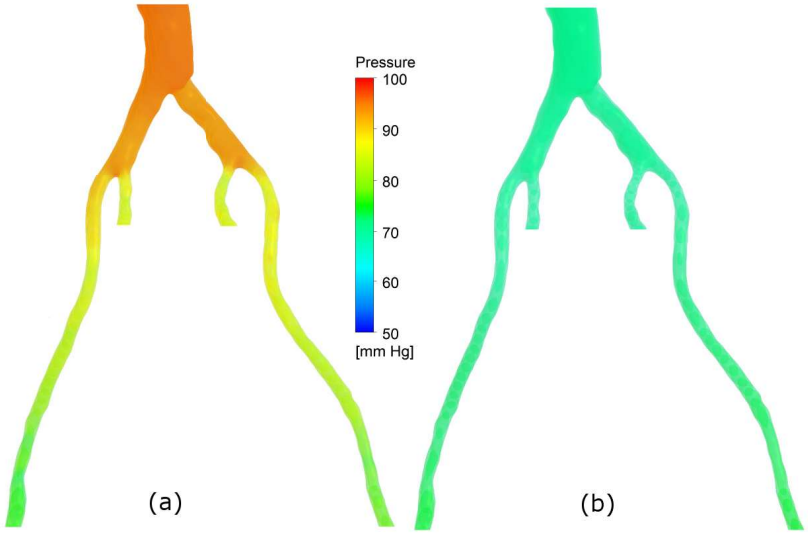


Fig. 4. Pressure contours at (a) Peak systolic flow and (b) maximum negative flow

Wall shear stress (WSS) is an important factor to investigate. It is considered to be a key haemodynamic parameter in both AD initiation and plaque formation [25, 29, 30]. A number of WSS indices are commonly used in the analysis of transient physiological flow simulations, in order to describe the WSS characteristics as a single spatial distribution.

The time-averaged wall shear stress (TAWSS) is defined by the following expression:

$$TAWSS = \frac{1}{T}\int_0^T |\tau(t)|\, dt \tag{12}$$

Where $|\tau(t)|$ is the magnitude of WSS vector at time *t*. This equation is applied at each location on the vessel wall to give the TAWSS distribution. An alternative index that provides insight into the nature of the oscillatory forces acting on the endothelium is the oscillatory shear index (OSI) [31]:

$$OSI = \frac{1}{2}\left(1 - \frac{\left|\frac{1}{T}\int_0^T \tau(t)dt\right|}{TAWSS}\right) \tag{13}$$

The TAWSS is shown in Fig. 5 in both left-posterior and right-anterior views. A small region of high TAWSS can be observed at the iliac bifurcation in Fig. 5a. Otherwise, the TAWSS is relatively low in the DA and left and right iliac arteries prior to the secondary bifurcations. Around the external/internal bifurcations, there are scattered regions of very high TAWSS, which persist along the external iliac arteries.

Fig. 6 shows the OSI in both views. In the descending aorta, the OSI is relatively high, at around 0.3, and the same is true of the right IA. In the left IA, there is a region of very low OSI. The OSI in the L_{int} artery is also low, and elsewhere the OSI is scattered.

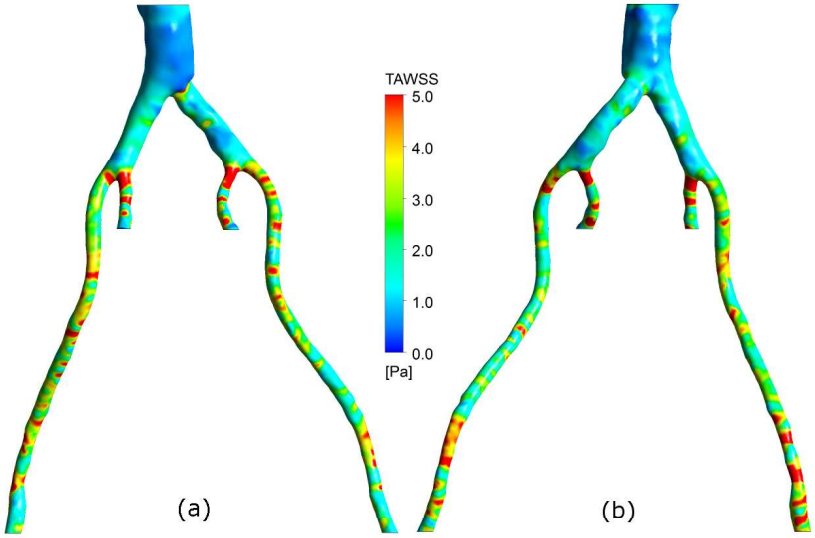


Fig. 5. TAWSS (a) Left posterior view (b) Right anterior view

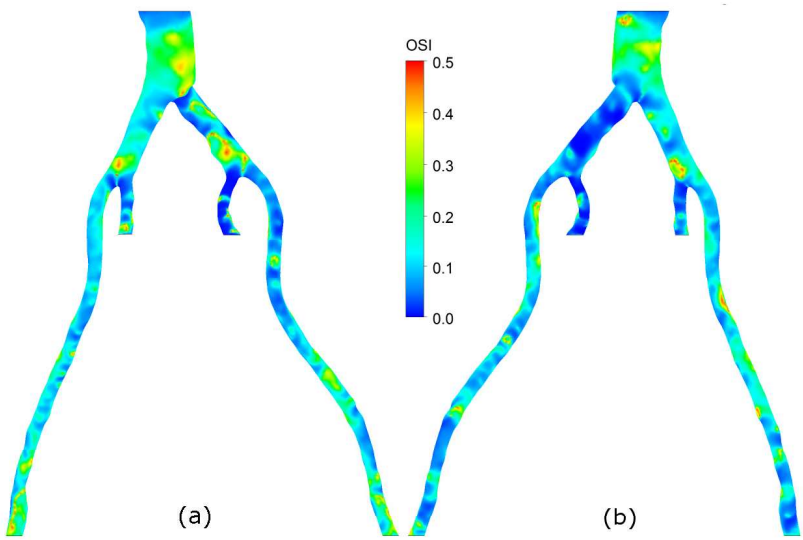


Fig. 6. OSI (a) Left posterior view (b) Right anterior view

Figures 7 and 8 show the endothelial cell shape index (ECSI) and the hydraulic conductivity ($L_{p,lj}$, units $mPa^{-1}s^{-1}$) of the leaky junctions in the arterial wall. ECSI values (Fig. 7) are between 0.1 and 0.6, which is consistent with the assumptions for modelling leaky junctions proposed by Olgac et al. [18] and Di Tomaso et al. [19]

$L_{p,lj}$ values (Fig. 8) are between 3.9×10^{-12} and 8.3×10^{-10} $mPa^{-1}s^{-1}$) with values larger than 1.19e-11 $mPa^{-1}s^{-1}$ (as reported by Tedgui and Lever [26] for normal junctions) in regions with lower WSS values.

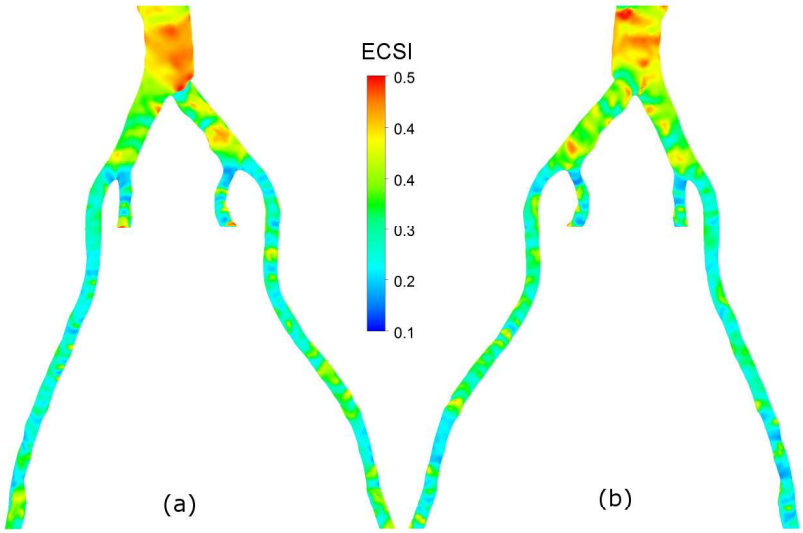


Fig. 7. Endothelial cell shape index (ECSI) for (a) Left posterior view (b) Right anterior view

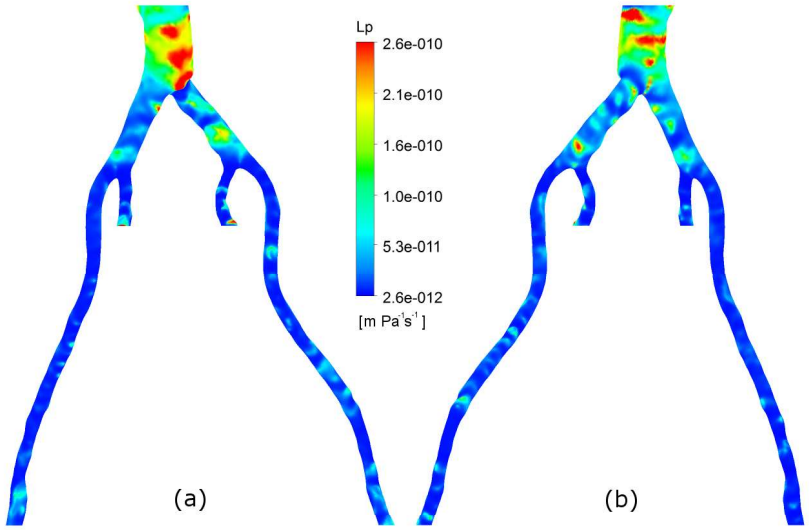


Fig. 8. Hydraulic conductivity of leaky junctions in the endothelium ($L_{p,lj}$). (a) Left posterior view (b) Right anterior view

4 Discussion

The acquired pulse pressure, as well as the maxima and minima of the pressure waves at each outlet, were close to those measured invasively, giving confidence that the boundary conditions were appropriately prescribed. The simulation results were processed to analyse the WSS, and other haemodynamic parameters were derived.

As expected, regions with low WSS show higher values for ECSI and L_p (having a similar pattern as OSI). Higher values of ECSI imply a higher number of endothelial cells producing leaky junctions, which in turn, would increase the permeability of the endothelium in those regions. L_p is inversely related to the resistance to mass flow provided by the endothelium, so higher values of L_p are associated with higher values of endothelial permeability. Thus regions where L_p is higher are expected to be more prone to develop atherosclerosis. Normal junctions in the endothelium will decrease in regions where the LDL flux through leaky junctions increases. This is normally most pronounced around the flow separation and reattachment points, where WSS is low. The increase in LDL flux through leaky junctions is the result of the increased number of leaky junctions themselves, combined with a decreased flow resistance through the entire leaky junction pathway. Because normal junctions and leaky junctions are parallel pathways, the decreased flow resistance of the leaky junction pathway leads to a decrease in the LDL flux through the normal junction pathway.

The areas that the model identified as prone to atherosclerosis formation were compared to the plaque distribution observed in the patient, prior to the virtual removal. The technique was able to predict the key locations of plaque formation, providing validation for the approach.

5 Conclusions

Given the difficulties in preventive diagnosis and clinical management of the disease, patient-specific computational models of atherosclerosis can offer much needed help by allowing the study of plaque formation *in silico* for individual patients. Understanding the effects that different haemodynamic modelling approaches and haemodynamic parameters have on the accuracy of detection of the atherosclerotic areas can lead to an improvement of *in silico* modelling techniques for atherosclerosis. These scenarios are currently being tested in the vascular services unit at UCH.

Acknowledgments. This work was supported by the EPSRC grant ''Personalised Medicine Through Learning in the Model Space'' (grant number EP/L000296/1).

References

1. Erbel, R., Alfonso, F., Boileau, C., Dirsch, O., Eber, B., Haverich, A., Rakowski, H., Struyven, J., Radegran, K., Sechtem, U., Taylor, J., Zollikofer, C., Klein, W.W., Mulder, B., Providencia, L.A.: Task Force on Aortic Dissection, European Society of Cardiology: Diagnosis and management of aortic dissection. Eur. Heart J. 22(18), 1642–1681 (2001)
2. Bastien, M., Dagenais, F., Dumont, É., Vadeboncoeur, N., Dion, B., Royer, M., Gaudet-Savard, T., Poirier, P.: Assessment of management of cardiovascular risk factors in patients with thoracic aortic disease. Blood Pressure Monitoring 17, 235–242 (2012)
3. Itani, Y., Watanabe, S., Masuda, Y.: Aortic calcification detected in a mass chest screening program using a mobile helical computed tomography unit. Relationship to risk factors and coronary artery disease. Circ. J. 68, 538–541 (2004)

4. Iribarren, C.: Patients with vascular calcifications are at increased risk of cardiovascular events: implications for risk factor management and further research. J. Intern. Med. 261, 235–237 (2007)
5. Tsai, T.T., Trimarchi, S., Nienaber, C.A.: Acute aortic dissection: perspectives from the International Registry of Acute Aortic Dissection (IRAD). Eur. J. Vasc. Endovasc. Surg. 37, 149–159 (2009)
6. Dietz, H.C., Cutting, G.R., Pyeritz, R.E., Maslen, C.L., Sakai, L.Y., Corson, G.M., Puffenberger, E.G., Hamosh, A., Nanthakumar, E.J., Curristin, S.M.: Marfan syndrome caused by a recurrent de novo missense mutation in the fibrillin gene. Nature 352, 337–339 (1991)
7. Pyeritz, R.E.: The Marfan Syndrome 51, 481–510 (2000)
8. Braverman, A.C.: Aortic dissection: Prompt diagnosis and emergency treatment are critical. Cleveland Clinic Journal of Medicine 78, 685–696 (2011)
9. Coady, M.A., Rizzo, J.A., Elefteriades, J.A.: Pathologic variants of thoracic aortic dissections. Penetrating atherosclerotic ulcers and intramural hematomas. Cardiol. Clin. 17, 637–657 (1999)
10. Tsai, T.T., Fattori, R., Trimarchi, S., Isselbacher, E., Myrmel, T., Evangelista, A., Hutchison, S., Sechtem, U., Cooper, J.V., Smith, D.E., Pape, L., Froehlich, J., Raghupathy, A., Januzzi, J.L., Eagle, K.A., Nienaber, C.A.: International Registry of Acute Aortic Dissection: Long-term survival in patients presenting with type B acute aortic dissection: insights from the International Registry of Acute Aortic Dissection. Circulation 114, 2226–2231 (2006)
11. LePage, M.A., Quint, L.E., Sonnad, S.S.: Aortic dissection: CT features that distinguish true lumen from false lumen. Am. J. Roentgenology 177, 207–211 (2001)
12. Willoteaux, S., Lions, C., Gaxotte, V., Negaiwi, Z., Beregi, J.P.: Imaging of aortic dissection by helical computed tomography (CT). European Radiology 14, 1999–2008 (2004)
13. Robinson, J.G., Fox, K.M., Bullano, M.F., Grandy, S.: The SHIELD Study Group: Atherosclerosis profile and incidence of cardiovascular events: A population-based survey. BMC Cardiovasc. Disord. 9, 46 (2009)
14. Svensson, L.G., Kouchoukos, N.T., Miller, D.C., Bavaria, J.E., Coselli, J.S., Curi, M.A., Eggebrecht, H., Elefteriades, J.A., Erbel, R., Gleason, T.G., Lytle, B.W., Mitchell, R.S., Nienaber, C.A., Roselli, E.E., Safi, H.J., Shemin, R.J., Sicard, G.A., Sundt III, T.M., Szeto, W.Y., III Wheatley, G.H.: Expert Consensus Document on the Treatment of Descending Thoracic Aortic Disease Using Endovascular Stent-Grafts. Ann. Thorac. Surg. 85, S1–S41 (2008)
15. O'Rourke, M.J., McCullough, J.P.: An investigation of the flow field within patient-specific models of an abdominal aortic aneurysm under steady inflow conditions. Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine 224, 971–988 (2010)
16. Alishahi, M., Alishahi, M.M., Emdad, H.: Numerical simulation of blood flow in a flexible stenosed abdominal real aorta. Scientia Iranica 18, 1297–1305 (2011)
17. Kim, H.J., Figueroa, C.A., Hughes, T.J.R., Jansen, K.E., Taylor, C.A.: Augmented Lagrangian method for constraining the shape of velocity profiles at outlet boundaries for three-dimensional finite element simulations of blood flow. Comp. Meth. Appl. Mech. Eng. 198, 3551–3566 (2009)
18. Olgac, U., Kurtcuoglu, V., Poulikakos, D.: Computational modeling of coupled blood-wall mass transport of LDL: effects of local wall shear stress. American Journal of Physiology-Heart and Circulatory Physiology 294, H909–H919 (2008)
19. Di Tomaso, G., Díaz-Zuccarini, V., Pichardo-Almarza, C.: A multiscale model of atherosclerotic plaque formation at its early stage. IEEE Trans. Biomed. Eng. 58, 3460–3463 (2011)

20. Diaz-Zuccarini, V., Di Tomaso, G., Agu, O., Pichardo-Almarza, C.: Towards personalised management of atherosclerosis via computational models in vascular clinics: technology based on patient-specific simulation approach. Healthcare Technology Letters, pp. 1–6 (2014)
21. Brown, A.G., Shi, Y., Marzo, A., Staicu, C., Valverde, I., Beerbaum, P., Lawford, P.V., Hose, D.R.: Accuracy vs. computational time: translating aortic simulations to the clinic. J. Biomech. 45, 516–523 (2012)
22. Alimohammadi, M., Agu, O., Balabani, S., Díaz-Zuccarini, V.: Development of a patient-specific simulation tool to analyse aortic dissections: Assessment of mixed patient-specific flow and pressure boundary conditions. Med. Eng. Phys. 36, 275–284 (2014)
23. Shi, Y., Lawford, P., Hose, R.: Review of Zero-D and 1-D Models of Blood Flow in the Cardiovascular System. BioMed. Eng. OnLine 10, 33 (2011)
24. Gijsen, F., Van de Vosse, F.N., Janssen, J.D.: The influence of the non-Newtonian properties of blood on the flow in large arteries: Steady flow in a carotid bifurcation model. J. Biomech. 32, 601–608 (1999)
25. Levesque, M.J., Liepsch, D., Moravec, S., Nerem, R.M.: Correlation of endothelial cell shape and wall shear stress in a stenosed dog aorta. Arteriosclerosis 6, 220–229 (1986)
26. Tedgui, A., Lever, M.J.: Filtration through damaged and undamaged rabbit thoracic aorta. Am. J. Phys. 247, H784–H791 (1984)
27. Bird, R.B., Stewart, W.E., Lightfoot, E.N.: Transport Phenomena. John Wiley and Sons (2007)
28. Sun, N., Wood, N.B., Hughes, A.D., Thom, S.A.M., Xu, X.Y.: Influence of pulsatile flow on LDL transport in the arterial wall. Ann. Biomed. Eng. 35, 1782–1790 (2007)
29. Gao, F., Guo, Z., Sakamoto, M., Matsuzawa, T.: Fluid-structure Interaction within a Layered Aortic Arch Model. J. Biol. Phys. 32, 435–454 (2006)
30. Gerdes, A., Joubert-Hübner, E., Esders, K., Sievers, H.H.: Hydrodynamics of aortic arch vessels during perfusion through the right subclavian artery. Ann. Thorac. Surg. 69, 1425–1430 (2000)
31. Ku, D.N., Giddens, D.P., Zarins, C.K., Glagov, S.: Pulsatile flow and atherosclerosis in the human carotid bifurcation. Positive correlation between plaque location and low oscillating shear stress. Arteriosclerosis 5, 293–302 (1985)

Identification of Biologically Significant Elements Using Correlation Networks in High Performance Computing Environments

Kathryn Dempsey Cooper, Sachin Pawaskar, and Hesham H. Ali

College of Information Science and Technology,
University of Nebraska at Omaha, NE 68182 USA
{kdempsey,spawaskar,hali}@unomaha.edu

Abstract. Network modeling of high throughput biological data has emerged as a popular tool for analysis in the past decade. Among the many types of networks available, the correlation network model is typically used to represent gene expression data generated via microarray or RNAseq, and many of the structures found within the correlation network have been found to correspond to biological function. The recently described gateway node is a gene that is found structurally to be co-regulated with distinct groups of genes at different conditions or treatments; the resulting structure is typically two clusters connected by one or a few nodes within a multi-state network. As network size and dimensionality grows, however, the methods proposed to identify these gateway nodes require parallelization to remain efficient and computationally feasible. In this research we present our method for identifying gateway nodes in three datasets using a high performance computing environment: quiescence in *Saccharomyces cerevisiae,* brain aging in *Mus Musculus,* and the effects of creatine on aging in *Mus musculus.* We find that our parallel method improves runtime and performs equally as well as sequential approach.

Keywords: high performance computing, parallel algorithms, correlation networks, gateway nodes.

1 Introduction

As the popularity of network modeling for big biological data grows, the need for algorithms and methods that can analyze these data grows with it. Network modeling in biological data came of age in 2001 with the finding of small world property in complex system [12]; protein-protein interaction networks were one of the models analyzed. Then came the structure-function correspondence: in PPI's, hub nodes are speculated to be linked with essential genes or proteins [3, 11, 12]; nodes in a clique tend to correspond to proteins in complex [3,7,10,16] , and the disassortativity of hubs could suggest that hub proteins are ancestral in nature [17]. The correlation network, where genes are represented as nodes, finds some measure of correlation between gene expression patterns to determine a relationship [13]. For example, linear relationships can be captured by the Pearson Correlation coefficient; networks built using this

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 607–619, 2015.

measure have been found to tend toward assortativity [17], to have a lower hub lethality rate [5], and to contain clusters whose manipulation suggests that the expression system is robust to minor changes [6,7].

The goal of the identification of gateway nodes is to identify the key genes in mechanistic changes between states. Gene expression experiments, particularly where sample size is large, provide an ideal experimental setup where comparison of states (treated, untreated or different time points) can occur while other key parameters are held consistent (tissue type, organism type and strain, etc). As such, in this research, we identify three datasets and the gateway nodes between the states found within them. Then, we take this gateway node analysis approach and parallelize it.

The recent integration of high performance computing approaches and bioinformatics or biomedical informatics methods approaches have allowed for massive strides in systems biology, or the identification of the mechanistic dynamics of a system as a whole. Previous work in this area, for example, has improved sequence assembly via Energy Aware parallelization, which minimizes energy and computational resources while improving runtime [21]. This marriage of computing and biological expertise is critical in the advancement of technologies designed to diagnose and prevent diseases, and as such, continued research in this area is critical.

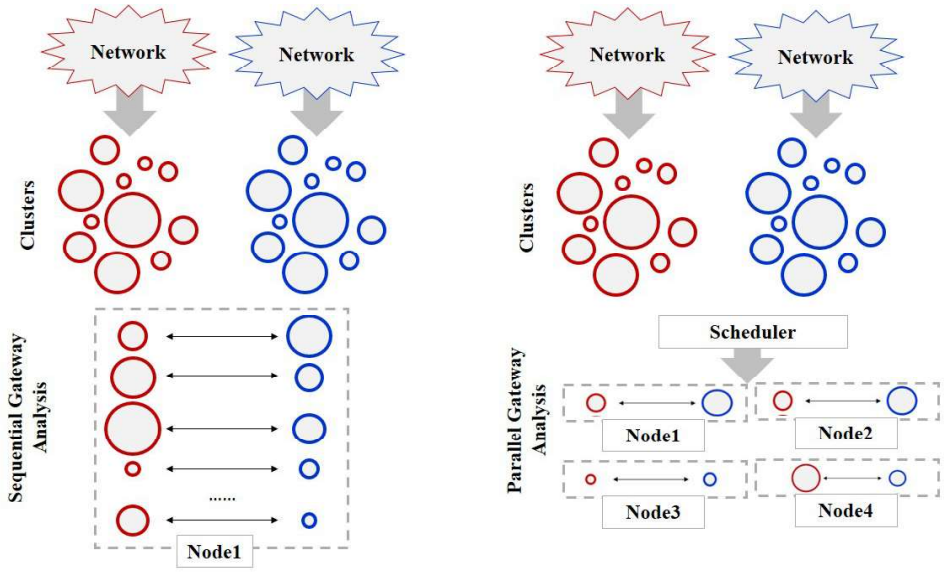


Fig. 1. The sequential versus naively parallel gateway nodes analysis. On the left, we have two networks, which after clustering, need to have a sequential pairwise comparison of clusters. In the parallel approach, a scheduler (the master node) takes the number of jobs and distributes them evenly among nodes (worker nodes).

2 Experimental Suite

In this research, three datasets are presented to highlight the computational and biological power of the parallel gateway analysis. Known datasets were drawn from NCBI's Gene

Expression Omnibus [9]. The first uses a model organism, *Saccharomyces cerevisiae*; this dataset is chosen for the vast array of knowledge available about the yeast organism, which allows for a more confident biological assessment of the gateway node functionality without actually performing any experiments *in vivo*. The second, GSE5078, is one of the datasets used in the original gateway node analysis; this dataset is used largely to determine if the same gateway nodes are identified sequentially versus in parallel. The final dataset is chosen for its large network size (more specifically, larger number of clusters) to highlight the scalability of the parallel method.

- *GSE5078*: Generated by Verbitsky *et al.* 2004 [14]; this dataset includes expression data from Bl/6 mice hippocampus separated into two groups: Young (YNG), at 2 months, and Middle-Aged (MID) at 15 months. Both sets have 9 samples.
- *GSE8542*: Generated by Aragon *et al.* 2008 [18]; this dataset includes expression data from BY4742 yeast separated into two groups: quiescent (QUI) or non-quiescent (NON).

RandomClique: Six sets of "clusters" made by random generation of 100 cliques between the sizes of 5 and 100 nodes. The clusters were grouped into six sets, R1, R2, R3, R4, R5, and R6, all consisting of 100 cliques each. Comparisons of the faux networks were performed in the following matchups: R1 vs. R2 (R1-R2), R3 vs. R4 (R3-R4), and R5 vs. R6 (R5-R6).

2.1 Network Creation and Manipulation

Networks were created by pairwise calculation of the Pearson Correlation coefficient (as described in [8]) with a correlation (ρ) threshold of 0.85 to 1.00; hypothesis testing was performed using the Student's t-test and values with p-value > 0.0005 were thrown out. The resulting network uses gene probes as nodes and correlated expression patterns as edges. As a creation quality check, the networks were checked for duplicate and self-edges; none were found.

Clustering of the networks was performed with AllegroMCODE v.1.0 [16]. Nodes with degree less than 15 were not used in cluster finding, and a scoring cutoff of 0.2 (the default) was used. Clusters with a minimum K-core of 10 were found using a maximum search depth of 10.

2.2 Gateway Node Identification

Per each dataset, gateway nodes are calculated as described in Dempsey 2014 and briefly here: Networks are first clustered to identify the dense groups within the network, and then the clusters are compared to determine if any nodes are shared between them. If nodes are shared, the number of edges between them and the clusters they connect are determined to calculate a gatewayness score. This gatewayness score is calculated as:

$$gatewayness_{nodeA} = \frac{\deg ree_{nodeA}}{\deg ree_{all\,gatewaynodes}} \qquad \text{(Equation 1)}$$

In this equation the gateway node A being studied is defined as any node shared between two clusters of a different state and the total degree of all gateway nodes is the sum of the degree of any node shared between two clusters of a different state. If node A is the only gateway node between two clusters 1 and 2 and has a degree of 50, the gatewayness score will be 50/50 = 1.00, or 100%. If there are two gateway nodes A and B, where the degree of A is 45 and the degree of B is 55, the gatewayness of A will be 45/(45+55) = 0.45 or 45%, and the gatewayness of B will be 55/(45+55) = 0.55 or 55%. Thus, gatewayness is a measure of the responsibility of a node's connectivity between two clusters of a different state.

One way to reduce the runtime of the gateway nodes analysis in large networks is by only allowing clusters of a certain density to be analyzed; for example, if a network has 100 clusters, a density filter can be imposed (say, where the edge density of the cluster is used to remove clusters); in previous studies, using a cluster density filter of 65% can remove up to 60% of the clusters analyzed. However, it is most beneficial to compare all possible clusters instead of imposing further restriction (and thus possibly removing more biological information), which our parallel algorithm approach allows for.

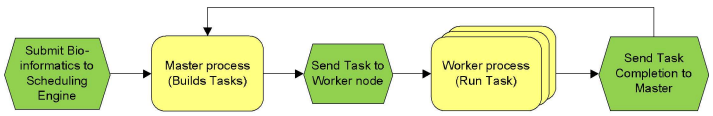


Fig. 2. Parallel implementation process flow diagram

2.3 High Performance Computing Environment

The gateway node analysis is an easily parallelizable problem – the algorithm takes a pair of clusters, compares the nodes between them, and when nodes overlap between clusters, calculates the edge intersection between the two clusters. The runtime for this analysis increases in linear time increases when the size, density, or number of clusters increases. However, the problem can be scheduled to different processors by simply determining how many comparisons need to be made and then delegating them to respective worker nodes from one master.

As shown in Figure 1, the sequential approach and parallel approach differ only in the determination of gateway nodes. First, networks are created or downloaded (the networks are assumed to originate from the same set of probes – genes, gene products, proteins, etc., or such that nodes can be paired together according to some mapping function). Next, networks are clustered – using any type of clustering function desired – and the resulting clusters are forwarded to the gateway analysis. In this study, we use a specific clustering approach known for its identification of small, dense clusters (MCODE), but any type of clustering can be made. Since this approach borrows from previous studies, the same clustering method used in Dempsey *et al.* 2013 was

```
Int main(int argc, char **argv)
{
   int rank;
   MPI_Init(argc, argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   if (rank == 0) {
      Init();
      exec_master(); // Builds tasks & sends to worker
   } else {
      exec_worker();
   }
   MPI_Finalize();
}
```

```
static void exec_worker(void) {
   char rundate[16], runtime[16], cmd[256];
   Work work;
   MPI_Status status;
   while (true) { // Receive a message from the master
      MPI_Recv(&work,1,Worktype,MASTER,MPI_ANY_TAG,MPI_COMM_WORLD,&status);
      if (status.MPI_TAG == EXITTAG) { // Check tag of the received message.
         return;
      }
      pBIProg->BuildCommandString(&work, cmd);
      ExecuteTask(cmd);

      // Send the result back to the master task
      strcpy(work.sNodeName, sProcessorName);
      work.iNode = RANK;
      strcpy(work.sRunDate, rundate);
      strcpy(work.sRunTime, runtime);
      work.iET = sw.ElapsedTime();
      MPI_Send(&work,1,Worktype,MASTER,WORKTAG,MPI_COMM_WORLD);
   }
}
```

Fig. 3. Pseudo-code of parallel implementation

used for comparison. Finally, the gateway analysis approach uses anywhere from 1-64 nodes to identify gateway nodes using code written in Perl.

2.4 Parallel Implementation

The input dataset, consists of cluster files as mentioned above which are stored in their respective directories. Let us say that Organism1 cluster files are in Dir1 and contains **m** cluster files, and Organism2 cluster files are in Dir2 and contain **n** cluster files. The scheduling engines master process reads creates tasks for gateway analysis by comparing these files against each other. It takes two clusters as input and outputs any gateway nodes and their scores; a wrapper is used sequentially to run the script and deliver all possible combinations of clusters. The Big O of our parallel approach is O (m*n). The master thread sends each task with the 2 files as input to worker processors running gateway analysis algorithm. The master thread manages the execution order of the gateway analysis step. Figure 2 below shows the process flow of our parallel implementation and the pseudo Code of this implementation is shown Figure 3. The code was implemented on the Tusker Cluster described below as well. Tusker is a 40 TF cluster consisting of 106 Dell R815 nodes using AMD 6272 2.1GHz processors, connected via Mellanox QDR Infiniband and backed by approximately 350 TB of Terascala Lustre-based parallel filesystem. All experiments were run on this cluster.

Table 1. Top ten gateway nodes for mouse and yeast networks

Network	ID	Dataset	Nodes	Edges	Density	Clusters	Clustering Runtime
Young	YNG	GSE5078	12368	72967	0.095%	35	31.434 seconds
Middle-Aged	MID	GSE5078	12340	79176	0.104%	36	20.298 seconds
Non-quiescent	NON	GSE8542	1541	2515	0.212%	11	1.793 seconds
Quiescent	QUI	GSE8542	2543	5363	0.166%	62	2.671 seconds

3 Results

The results of our naively parallel gateway node analysis study are below. Table 1 describes the network sizes, edge density, number of clusters, clustering parameters, and clustering runtime. While the numbers of nodes and edges differ greatly due to difference in genome sizes, the density of the networks are relatively similar, and all networks are sparse. Using the same parameters to identify clusters in each network reveals a similar number of clusters in the mouse network (35 in the YNG and 36 in the MID) compared to the yeast network which has a more varied number (11 in the NON and 62 in the QUI). Clustering runtime appears to have no relationship with density, but rather seems to be linked to overall network size via edge count.

Table 2. Top ten gateway nodes for mouse and yeast networks

MOUSE - 0% Density		MOUSE - 65% Density		YEAST – 0% Density	
Gene ID	Gatewayness Score:	Gene ID	Gatewayness Score:	Gene ID	Gatewayness Score:
Map3k2	100.00%	Sla	100.00%	MCM21	33.33%
Pira1	100.00%	Matn3	100.00%	CPR5	33.33%
Ace	100.00%	Dio1	100.00%	TIM11	33.33%
Cts7	100.00%	Fbp1	100.00%	YGR164W	33.33%
Six3	100.00%	Ceacam12	100.00%	CBP4	33.33%
Immp1l	100.00%	Ptprb	100.00%	RPL1B	33.33%
Ythdf2	100.00%	Plin4	100.00%	GTR2	25.00%
Krt25	100.00%	Cldn1	100.00%	HGH1	25.00%
Tsks	100.00%	Akr1c21	100.00%	CRH1	25.00%
Vil1	100.00%	Ltc4s	100.00%	CLC1	25.00%

3.1 Model Organism – *S. cerevisiae* Gateway Nodes

There were 97 gateway nodes identified in the sequential and all parallel runs of the yeast network dataset; there were no gateway nodes with a score of 1.00. The gateway

nodes identified in each respective run did not change with processor number. The density threshold used for yeast was 0%, meaning that any clusters that overlapped with one another were considered. While the yeast networks are relatively small, in larger networks, this all to all comparison with no density filter is desired. A density filter is typically used to reduce the amount of clusters to compare to improve runtime of gateway identification, but via naïve parallelization of the approach, all clusters can be compared. Further, this can be used to determine the distribution of gateway nodes and their relative functional impact according to cluster density, if such a relationship exists.

Gene list analysis of the gateway nodes [15] was performed using PantherDB's tool (version 8.1) [19, 20]. Gateway nodes were functionally classified according to Molecular Function, Biological Process, and Pathway. The results of these classifications are shown in Figures 4 and 5. The classifications of genes in terms of Molecular Function (MF) and Biological Process (BP) are largely standard with the majority of genes involved in metabolic processes and catalytic activity (the profile of BP and MF classification in the mouse dataset is very similar – see Figure 4). However, in the pathway classification set, telling evidence of gateway biological impact emerges. The EGF receptor signaling pathway has been implicated as an upstream regulator in astroglial cells in the transition from quiescence to reactivity [1]. The PDGF signaling pathway plays a similar role; stimulation of cell growth and proliferation; quiescence stems out of the metabolism by activation of certain elements [2]. Glycolysis, the third most pathway identified via the gateway node classification, plays a major role in the shift from non-quiescence to quiescence. Glucose levels available in media can be used to stimulate the shift from non-quiescence to quiescence; [2] suggests that this is due to the inherent changes caused in glucose metabolism when glucose is lacking or present in media.

3.2 Known Dataset – GSE5078 Gateway Nodes

There were 172 gateway nodes identified in the sequential and all parallel runs of the mouse network dataset at 0% density threshold in mice; for the 65% density threshold, 25 gateway nodes were identified. In parallel and sequential runs for both parameterizations, all gateway nodes matched. Functional classification of gateway nodes at 0% density threshold and 65% density thresholds are shown in Figures 5 and 6. The gateway nodes identified at 65% match up with those identified in Dempsey *et al.* The gateway nodes identified within this dataset have been found to be related to aging. One example of this is Klotho and Ins2 (not listed in the top 10 gateway nodes, shown in Table 2), which are involved in the insulin signaling pathway, which has long been known to be involved in biological aging.

Of the top twenty gateway nodes identified in the mouse datasets for 0% and 65% densities, nine (45%) are protein binding molecules (*Map3k2, Ace, Six3, Kr25, Vil1,Sla, Fbp1,Ptprb,* and *Cldn1*), meaning that their gene products they bind with other proteins; nearly all of the gateway genes identified are pleiotropic, or having a number of roles in the cell. This follows with the concepts proposed in Dempsey *et al.*, that gateway nodes are tied to the mechanistic changes in expression that occur to restore homeostasis in changing environments within the cell.

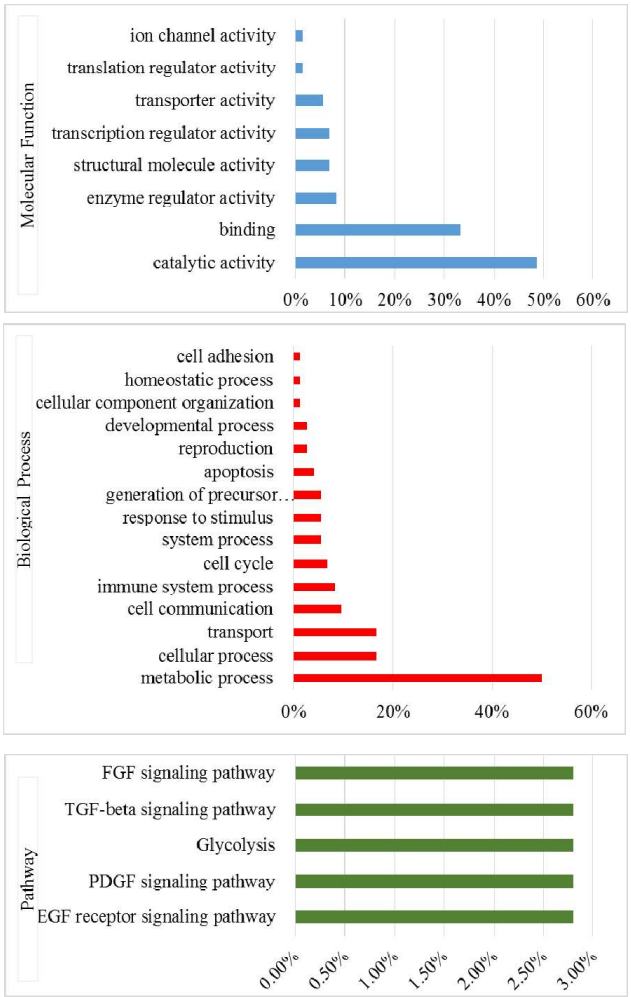


Fig. 4. The functional classifications of the yeast gateway nodes at 0% density. Blue – Biological process, Red – Molecular Function, and Green – Pathway. The axis is the percentage of genes in the gateway node list with that annotation compared to the background (mouse genome).

3.3 Scalability

The smaller model network analyses (mouse and yeast) both ran in minimal time sequentially – 144 seconds for yeast, 305 seconds for mouse at 0%, and 309 seconds at 65%. While this time requirement hardly calls for parallelization, extending the gateway node analysis into larger and more dimensional studies will require analysis of much larger networks and datasets at many more states. Systems biology approaches nearly guarantee that the data available will continue. Regardless, parallelization of the gateway node analysis in these models shows good scalability, as shown in Figure 7.

The random networks are designed to represent the scalability of these larger networks, and on this larger view, the scalability of this naively parallel approach

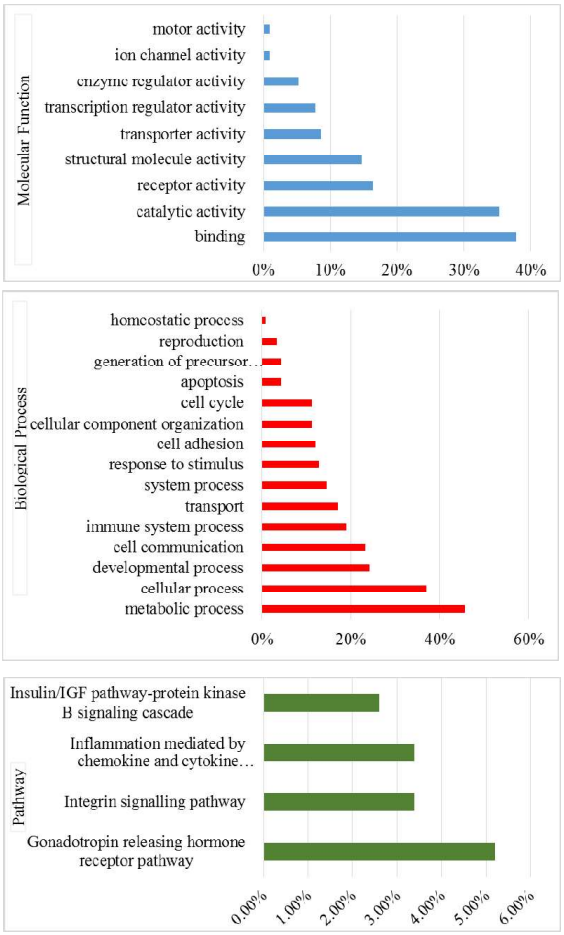


Fig. 5. Left: The functional classifications of the mouse gateway nodes at 0% density. Blue – Biological process, Red – Molecular Function, and Green – Pathway. The axis is the percentage of genes in the gateway node list with that annotation compared to the background (mouse genome).

does not disappoint. For the R1-R2 analysis, the runtime takes 68 minutes using 1 processor, and 1 minute and 25 seconds using 64 processors, a speedup of 48.6. The runtime and speedup for the random runs are shown in Figures 8 and 9. The naively parallel approach described reduces runtime, particularly as networks get larger.

4 Discussion

In recent years, modeling of high throughput biological data via network or graph theoretic modeling has emerged as a popular tool for analysis. The correlation network model, used to represent gene expression data, is one of many different types of models that rely on correlation of expression patterns to form internal graph structures. One

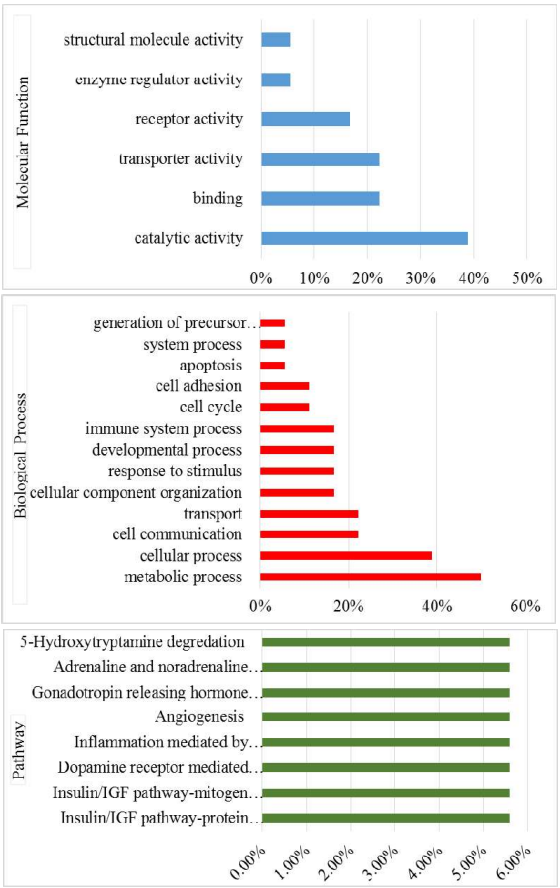


Fig. 6. Left: The functional classifications of the mouse gateway nodes at 65% density. Blue – Biological process, Red – Molecular Function, and Green – Pathway. The axis is the percentage of genes in the gateway node list with that annotation compared to the background (mouse genome).

of these structures, the gateway node, has been found to represent co-regulation with distinct groups of genes at different conditions or treatments. The structure that results typically represents 1-10% of the original network, making them a desirable target for deciphering the mechanistic changes between states or environments. As network size and dimensionality grows, however, the methods proposed to identify these gateway nodes require parallelization to remain efficient and computationally feasible. In this research we have presented our method for identifying gateway nodes in three datasets using a high performance computing environment: quiescence in *Saccharomyces cerevisiae,* brain aging in *Mus Musculus,* and the effects of creatine on aging in *Mus musculus.* The results show that our parallel method improves runtime and performs equally as well as sequential approach, meaning that as network dimensionality and size increases, we will have the tools required to analyze the entire system.

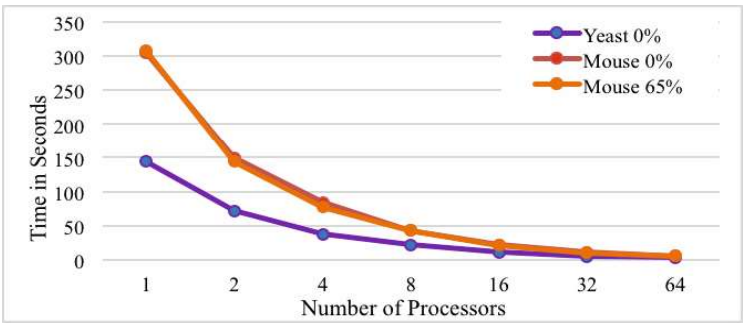


Fig. 7. Scalability for the Yeast and Mouse networks. The x-axis represents the number of processors used and the y-axis represents the time in seconds.

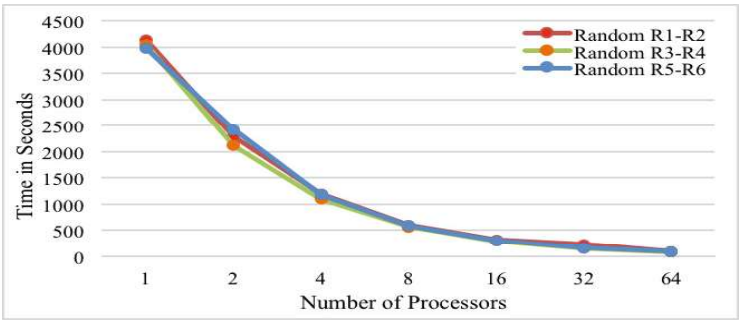


Fig. 8. Scalability for the Random networks. The x-axis represents the number of processors used and the y-axis represents the time in seconds.

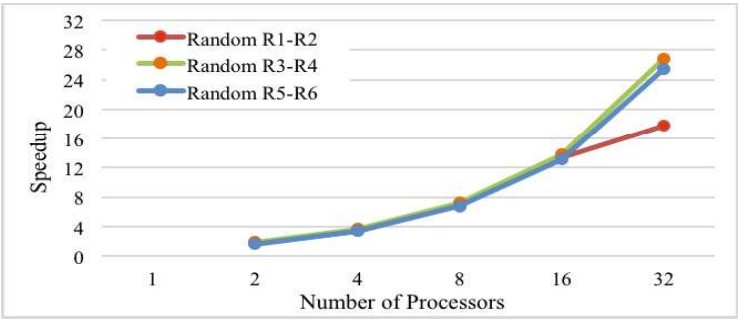


Fig. 9. Speedup for the Random Networks. The x-axis represents the number of processors

References

1. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)
2. May, P., Ehrlich, H.-C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)

3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
6. National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`
7. Liu, Chen, Johns, Neufeld: Epidermal growth factor receptor activation: an upstream signal for transition of quiescent astrocytes into reactive astrocytes after neural injury. J. Neurosci. 26(28), 7532–7540 (2006)
8. Laporte, D., Lebaudy, A., Sahin, A., Pinson, B., Ceschin, J., Daignan-Fornier, B., Sagot, I.: Metabolic status rather than cell cycle signals control quiescence entry and exit. J. Cell Biol. 192(6), 949–957 (2011), doi:10.1083/jcb.201009028
9. Barabasi, A.L., Oltvai, Z.N.: Network biology: Understanding the cell's functional organization. Nature Reviews. Genetics 5(2), 101–113 (2004)
10. Bult, C.J., Eppig, J.T., Kadin, J.A., Richardson, J.E., Blake, J.A., and the members of the Mouse Genome Database Group.: The Mouse Genome Database (MGD): mouse biology and model systems. Nucleic Acids Res. 36(database issue), D724–D728 (2008)
11. Dempsey, K., Ali, H.: On the discovery of Cellular subsystems in correlation networks using centrality measures. Current Bioinformatics 7(4) (2014)
12. Duraisamy, K., Dempsey, K., Ali, H.: S. Bhowmick.: A noise reducing sampling approach for uncovering critical properties in large scale biological networks. In: High Performance Computing and Simulation 2011 International Conference (HPCS), Istanbul, Turkey, July 4-8 (2011)
13. Dong, J., Horvath, S.: Understanding network concepts in modules. BMC Systems Biology 1, 24 (2007)
14. Ewens, W.J., Grant, G.R.: Statistical methods in bioinformatics, 2nd edn. Springer, New York (2005)
15. Edgar, R., Domrachev, M., Lash, A.E.: Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nuc. Acid Res. 30(1), 207–210 (2002)
16. Enright, A.J., Van Dongen, S., Ouzounis, C.A.: An efficient algorithm for large-scale detection of protein families. Nucleic Acids Research 30(7), 1575–1584 (2002)
17. Hao, D., Li, C.: The dichotomy in degree correlation of biological networks. PloS One 6, e28322 (2011), doi: 10.1371/journal.pone.0028322
18. Jeong, H., Mason, S.P., Barabasi, A.L., Oltvai, Z.N.: Lethality and centrality in protein networks. Nature 411(6833), 41–42 (2001)
19. Opgen-Rhein, R., Strimmer, K.: From correlation to causation networks: A simple approximate learning algorithm and its application to high-dimensional plant gene expression data. BMC Systems Biology 1, 37 (2007)
20. Verbitsky, M., Yonan, A.L., Malleret, G., Kandel, E.R., Gilliam, T.C., Pavlidis, P.: Altered hippocampal transcript profile accompanies an age-related spatial memory deficit in mice. Learning & Memory (Cold Spring Harbor, N.Y.) 11(3), 253–260 (2004)
21. Subramanian, A., Tamayo, P., Mootha, V., Mukherjee, S., Ebert, B., Gilette, M., Paulovich, A., Pomeroy, S., Golub, T., Lander, E., Mesirov, J.P.: Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wise expression profiles. Proc. Natl. Acad. Sci. 102(43), 15545–15550 (2005)

22. Yoon, J.S., Jung, W.H.: A GPU-accelerated bioinformatics application for large-scale protein interaction networks. APBC poster presentation (2011)
23. Newman, M.: Assortative Mixing in Networks. Phys. Rev. Lett. 89(20), 208701 (2002)
24. Aragon, A.D., Werner-Washburne, M.: Characterization of differentiated quiescent and non-quiescent cells in yeast stationary-phase cultures. Mol. Biol. Cell 19(3), 1271–1280 (2008)
25. Miu, H., Muruganujan, A., Thomas, P.: PANTHER in 2013: Modeling the evolution of gene function, and other gene attrbutes, in the context of phylogenetic trees. Nucl. Acids Res. 41(database issue), D377–D386 (2012)
26. Thomas, P., Kejariwal, A., Guo, N., Mi, H., Campbell, M.J., Muruganujan, A., Lazareva-Ulitsky, B.: Applications for protein sequence-function evolution data: mRNA/protein expression analysis and coding SNP tools. Nuc. Acids Res. 34(suppl. 2), W645-W650
27. Pawaskar, S., Warnke, J., Ali, H.: An energy-aware bioinformatics application for assembling short-reaads. In: High Performance Computing Systems, HPCS 2013, pp. 154–160. IEEE (2012)

Enhancing the Parallelization of Non-bonded Interactions Kernel for Virtual Screening on GPUs

Baldomero Imbernón, Antonio Llanes, Jorge Peña-García, José L. Abellán, Horacio Péréz-Sánchez, and José M. Cecilia

Bioinformatics and High Performance Computing Research Group (BIO-HPC),
Computer Science Department,
Universidad Católica San Antonio de Murcia (UCAM), Spain
{bimbernon,allanes,jpena,jlabellan,hperez,jmcecilia}@ucam.edu

Abstract. Virtual Screening (VS) methods can considerably aid clinical research, predicting how ligands interact with drug targets. Most VS methods suppose a unique binding site for the target, usually derived from the interpretation of the protein crystal structure. But it has been demonstrated that in many cases, diverse ligands interact with unrelated parts of the target and many VS methods do not take into account this relevant fact.However, this fact increases the computationally complexity exponentially. In this work we enhance the parallelization of non-bonded interactions kernel for VS methods on Nvidia GPU architectures. We show several parallelization strategies that lead to a speed up factor of 15x compared to previous GPU implementations.

Keywords: Virtual Screening, GPUs, HPC.

1 Introduction

The discovery of new drugs can enormously benefit from the use of Virtual Screening (VS) methods [4]. The different approaches used in VS methods differ mainly by the way they model the interacting molecules but all of them have in common that they screen databases of chemical compounds containing up to millions of ligands [3]. Larger databases increase the chances of generating hits or leads, but the computational time needed for the calculations increases not only with the size of the database but also with the accuracy of the chosen VS method. Fast docking methods with atomic resolution require a few minutes per ligand [15], while more accurate molecular dynamics-based approaches still require hundreds or thousands of hours per ligand [14]. Therefore, the limitations of VS predictions are directly related to a lack of computational resources, a major bottleneck that prevents the application of detailed, high-accuracy models to VS.

In most of the VS methods the biological system is represented in terms of interacting particles. For the calculation of the interaction energies, classical potentials are commonly used, separated into bonded and non-bonded terms. The

F. Ortuño and I. Rojas (Eds.): IWBBIO 2015, Part II, LNCS 9044, pp. 620–626, 2015.

latter describe interactions between all the elements of the system. The relevant non-bonded potentials used in VS calculations are the Coulomb and the Lennard-Jones potentials, since these describe very accurately the most important short and long range interactions between protein and ligand atoms [5].

In VS methods the most intensive computations are spent in the calculation of non-bonded kernels. For example, in Molecular Dynamics it takes up to 80% of the total execution time [7]. Thus this part can be considered as a bottleneck, and it has been shown that its parallelization and optimization [10] permits VS methods to deal with more complex systems, simulate longer time scales or screen larger databases.

High Performance Computing (HPC) solutions like GPUs [6,11] have demonstrated they can increase considerably the performance of the different VS methods, as well as, the quality and quantity of the conclusions we can get from screening [12]. In this work, we enhance the simulation of the Lennard-Jones potentials kernels of virtual screening on Graphics Processing Units. Our starting point is the work described in [12]. Then, we provide two different new alternatives for this kernel. The former evenly distributes the atoms into warps to avoid warp divergences and increase the Kepler's SMx occupancy. The latter is a new design that focuses on a region of interest to calculate the potential instead of calculating the interactions with the whole protein. Although this latter approach compromises the quality of the results, the computationally time for the simulation is decreased drastically.

The rest of the paper is structured as follows: First, we introduce the sequential baselines of the targeted kernel before our CUDA parallel designs are introduced to the reader. Then, a preliminary experimental results are shown to finish with some conclusions and directions for future work.

2 The Lennard-Jones Potential on the Graphics Processing Units

This Section briefly summarizes the Lennard-Jones potential calculation kernel, beginning from the sequential code, and analyzing different CUDA alternative designs. We first briefly review the main characteristics of CUDA [9], for the benefit of readers who are unfamiliar with the programming model. CUDA is based on a hierarchy of abstraction layers; the *thread* is the basic execution unit; threads are grouped into *blocks*, each of which runs on a single multiprocessor, where they can share data on a small but extremely fast memory. A *grid* is composed of blocks, which are equally distributed and scheduled among all multiprocessors. The parallel sections of an application are executed as *kernels* in a SIMD (Single Instruction Multiple Data) fashion, that is, with all threads running the same code. A kernel is therefore executed by a grid of thread blocks, where threads run simultaneously grouped in batches called *warps*, which are the scheduling units.

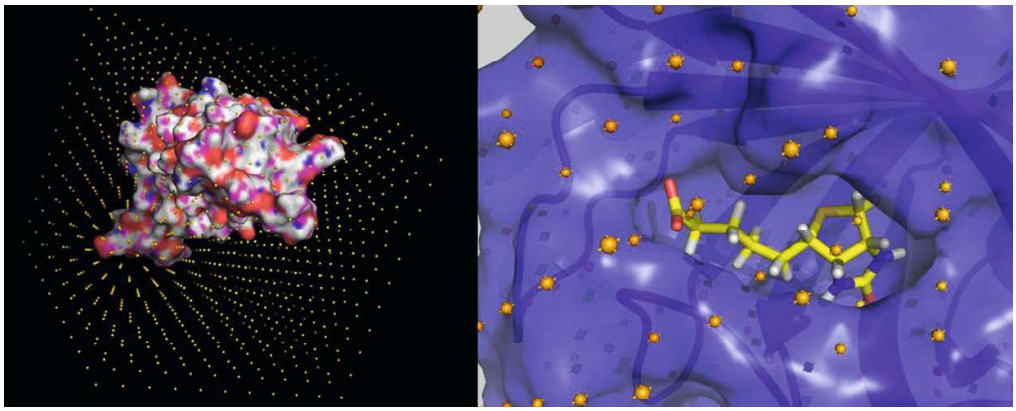


Fig. 1. (A) Representation of the grid for the protein streptavidin. Length of the side of the cube (L) is 50 Å, spacing between grid points d is 5 Å, and the total number of grid points is equal to 11^3. (B) Biotin in the binding pocket of streptavidin.

2.1 The Roadmap for the Optimization

In our previous work, we pointed out the main bottleneck of Virtual Screening methods are related to the computation of full non-bonded interactions Kernels and how GPUs can yield speedups of up to 260 times [2]. Nevertheless, those kernels need to perform up to N^2 interaction calculations (being N = total number of particles in the system) and even using GPUs, the required computation time grows polynomially with N, which definitely limits the successful simulation of large systems.

Taking a look at different algorithmic alternatives, we decided to evaluate grid kernels as an alternative to enhance the computational complexity of our simulations [8]. In [13], we empirically demonstrated that up to 200x speedup factor could be obtained by those grid kernels, even though they were executed in sequential architectures. An additional 30x speedup factor was provided by applying GPUs to these grid kernels.

We briefly summarizes the calculation of non-bonded interactions using grids (see Algorithm 1). The protein is placed inside a cube of minimal volume $Vol = L^3$ that encloses it. A three dimensional grid is created dividing the cube into $(N-1)^3$ smaller cubes of identical volume, each one of side length $d = L/N$, so that the total number of grid points is N^3. A graphical depiction of the grid for streptavidin can be seen in Figure 1 (A) and in more detail for the ligand biotin on its binding pocket in Figure 1(B).

Once the protein grid is loaded into memory, the calculation of the Lennard-Jones potential for the protein-ligand system is performed as follows; for each ligand atom i with charge q_i at point P_i we calculate which are the eight closest protein grid point neighbours. Next, an interpolation procedure is applied to estimate the value of the Lennard-Jones potential due to all protein atoms at P_i. The same procedure is applied to all ligand atoms summing them up.

Algorithm 1. Sequential pseudocode using grids for the calculation of the Lennard-Jones potentials

```
1. for i = 1 to N_simulations do
2.    for j = 1 to nlig do
3.       index = positionToGridCoordinates(VDWGRidInfo, j)
4.       for k = 0 to numNeighbours(VDWGRid[index]) do
5.          vdwTerm+ = vdwEnergy(j, VDWGRid[index][k])
6.       end for
7.       energy[i * nlig + j] =vdwTerm;
8.    end for
9. end for
```

Algorithm 2. GPU pseudocode for the calculation of the Lennard-Jones potentials

```
1. for all nBlocks do
2.    rlig = rotate(clig[myAtom], myQuaternion)
3.    ilig = shift(myShift, rlig)
4.    index = positionToGridCoordinates(VDWGridInfo, ilig)
5.    for k = 0 to numNeighbours(VDWGRid[index]) do
6.       vdwTerm+ = vdwEnergy(ilig, VDWGRid[index][k])
7.    end for
8.    energy_shared[myAtom] = vdwTerm
9.    totalEnergy = parallelReduction(energy_shared)
10.   if threadId == (numThreads%nlig) then
11.      energy[mySimulation]+ = totalEnergy
12.   end if
13. end for
```

2.2 Our Departure GPU Design

The Algorithm 2 shows the pseudocode of the Lennard-Jones potentials, where $VDWGridInfo$ and $VDWGrid$ are the grid description and the grid data, both stored in the GPU global memory. Each thread calculates the energy of a single atom. Each thread applies the rotation and displacement corresponding to the simulation over the ligand model in order to obtain the current atom position (lines 2-3). Then, it calculates the grid position, calculates the lennard-Jones potential using the neighbors stored in the $VDWGrid$ and stores the result in shared memory (lines 4-9) . The parameters needed by the $VDWEnergy$ procedure is previously stored in the GPU constant memory. Finally, threads of the same simulation sum up their results by a parallel reduction (line 9) and one of these threads accumulates the final result in global memory (lines 11-12).

2.3 Improving Memory Usage and Throughput

This section summarizes two alternative GPU designs. The former reduces the memory usage for our simulation. One of the main problem in docking simulation

is the memory usage whenever large proteins are simulated to interact with ligands. Even more so for our docking perspective as it scans the full protein surface using several different ligand conformation.

In our previous design, the full protein is stored in GPU device memory as all the protein's atoms actually interact with every atom of the ligand. However, this interaction can be limited to those atoms that are closer to the ligand as they have more influence in whole kernel calculation. In this way, the neighborhood to a particular ligand's conformation is stored at a given time. The number of atoms in the neighborhood is a degree of freedom which may affect to both: performance and accuracy.

The latter design is based on the computation distribution among all threads that execute a given kernel. In this design, a conformation is identified to a warp not like our previous design which identified a conformation per block. In this way, we can take advantage of the new shuffle instructions that allows inter-thread communication among threads within the same warp.

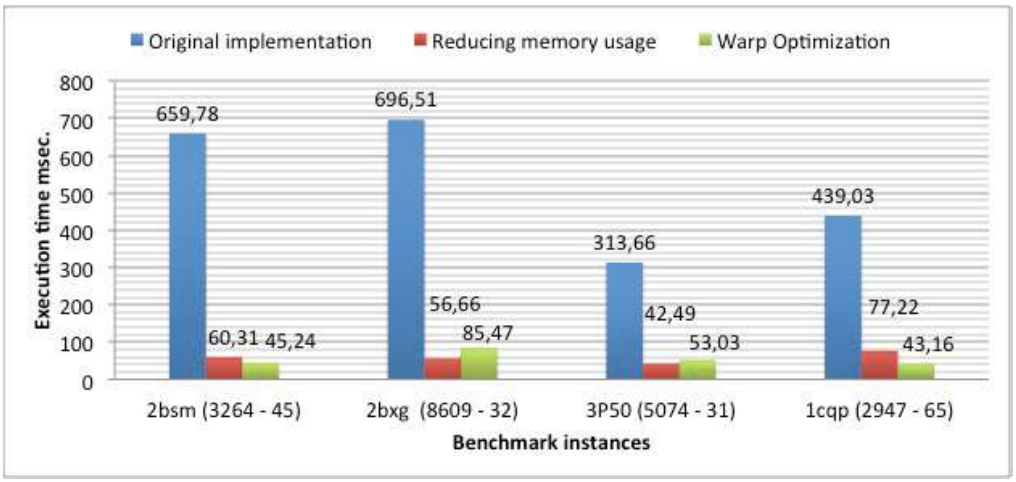
3 Experimental Results

This section briefly shows the experimental results obtained with our implementations. First of all we review our experimental set up.

In this work, we target a Kepler-based architecture Tesla K40c. Kepler is the last generation of Nvidia GPU architecture [1]. Compared to previous designs (Fermi), it extends the number of cores within a multiprocessor from 32 to 192 and the scheduling units from 2 to up to 8 warps at a time. In addition the L2 cache doubles its size. Moreover, it also introduces new capabilities for irregular computation like dynamic parallelism and Hyper Q. Despite the resource additions (resulting in a far higher transistor count per unit area), Kepler GPUs are three times more power-efficient than previous generations. This is mainly achieved keeping the frequency below 1 GHz and using a manufacturing process of 28 nm. It has up to 2880 cores running at 0.88 GHz, giving a raw processing power up to 5068 GFLOPS. The memory speed is 3,0 GHz with a 384-bits memory bus width that provides a bandwidth of 288 GB/sec. The memory size is 12 GB of GDDR5 with ECC capabilities.

On the software side, The CUDA programming model is used to program the Nvidia Tesla K40c. More precisely, CUDA toolkit 6.5 leverages the Nvidia architecture. Moreover, we carried out VS calculations for the direct prediction of binding poses using four different ligands, taken form the Protein Data Bank (PDB), that conveniently represent chemical diversity of large compound databases.

The Figure 3 shows the execution times for our GPU implementations. Our departure point it was able to deal with the whole molecule but it developed to leverage previous generation of Nvidia GPU architectures. Depending of the protein and ligand size our both alternatives behave differently. The former approach is the less accurate solution although it also consume 80% less memory than the others. The latter is however the fastest implementation and it fully reproduce the non-bonded interactions.



4 Conclusions and Outlook

In this work we have presented different GPU implementations to enhance the most expensive part of Virtual Screening methods, i.e. the interactions between all the elements of the system, and particularly, the Lennard-Jones potential. Our experimental results reveal that the optimization on new generation of GPUs can lead to an important speed-up factor compared to previous implementations. Moreover, the realistic simulation protein-ligand interactions can exceed the memory capacity of the GPU and thus clever ideas need to be implemented to place all the information on reduced GPU memories.

In the next steps we want to include these implementations on our docking program called BINDSURF to deal with the simulation of larger systems. Lastly, we are also working on improved scoring functions to include efficiently metals and aromatic interactions, and a GPU based method, already developed in our group for implicit solvation models.

Acknowledgements. This work has been funded by grants from the Fundación Séneca of the Región of Murcia (18946/JLI/13) and by the Nils Coordinated Mobility under grant 012-ABEL-CM-2014A, in part financed by the European Regional Development Fund (ERDF). We also thank Nvidia for the hardware donation under CUDA Teaching Program. Moreover, this work was also partially supported by the computing facilities of Extremadura Research Center for Advanced Technologies (CETACIEMAT), funded by the European Regional Development Fund (ERDF). CETACIEMAT belongs to CIEMAT and the Government of Spain. The authors also thankfully acknowledge the computer resources and the technical support provided by the Plataforma Andaluza de Bioinformática of the University of Málaga.

References

1. AMD. Nvidia corporation. the kepler architecture (2013)
2. Guerrero, G.D., Pérez-Sánchez, H., Wenzel, W., Cecilia, J.M., García, J.M.: Effective Parallelization of Non-bonded Interactions Kernel for Virtual Screening on GPUs. In: Rocha, M.P., Rodríguez, J.M.C., Fdez-Riverola, F., Valencia, A. (eds.) PACBB 2011. AISC, vol. 93, pp. 63–69. Springer, Heidelberg (2011)
3. Irwin, J.J., Shoichet, B.K.: ZINC–a free database of commercially available compounds for virtual screening. Journal of Chemical Information and Modeling 45(1), 177–182 (2005)
4. Jorgensen, W.L.: The Many Roles of Computation in Drug Discovery. Science 303, 1813–1818 (2004)
5. Jorgensen, W.L., Chandrasekhar, J., Madura, J.D., Impey, R.W., Klein, M.L.: Comparison of simple potential functions for simulating liquid water. The Journal of Chemical Physics 79(2), 926–935 (1983)
6. Kadau, K., Germann, T.C., Lomdahl, P.S.: Molecular Dynamics Comes of Age: 320 Billion Atom Simulation on BlueGene/L. International Journal of Modern Physics C 17(12), 1755–1761 (2006)
7. Kuntz, S.K., Murphy, R.C., Niemier, M.T., Izaguirre, J., Kogge, P.M.: Petaflop computing for protein folding. In: In Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing, pp. 12–14
8. Meng, E.C., Shoichet, B.K., Kuntz, I.D.: Automated docking with grid-based energy evaluation. Journal of Computational Chemistry 13(4), 505–524 (1992)
9. NVIDIA Corporation. NVIDIA CUDA C Programming Guide 6.5 (2014)
10. Pérez-Sánchez, H., Wenzel, W.: Optimization methods for virtual screening on novel computational architectures. Curr. Comput. Aided Drug. Des. 7(1), 44–52 (2011)
11. Prakhov, N.D., Chernorudskiy, A.L., Gaiin, M.R.: VSDocker: A tool for parallel high-throughput virtual screening using AutoDock on Windows-based computer clusters. Bioinformatics 26(10), 1374–1375 (2010)
12. Sánchez-Linares, I., Pérez-Sánchez, H., Cecilia, J.M., García, J.M.: High-throughput parallel blind virtual screening using bindsurf. BMC Bioinformatics 13(suppl. 14), S13 (2012)
13. Sánchez-Linares, I., Pérez Sánchez, H.E., García, J.M.: Accelerating grid kernels for virtual screening on graphics processing units. In: PARCO, pp. 413–420 (2011)
14. Wang, J., Deng, Y., Roux, B.: Absolute Binding Free Energy Calculations Using Molecular Dynamics Simulations with Restraining Potentials. Biophys. J. 91(8), 2798–2814 (2006)
15. Zhou, Z., Felts, A.K., Friesner, R.A., Levy, R.M.: Comparative performance of several flexible docking programs and scoring functions: enrichment studies for a diverse set of pharmaceutically relevant targets. Journal of Chemical Information and Modeling 47(4), 1599–1608 (2007)