# Acceleration and Verification of Virtual High-throughput Multiconformer Docking

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Sally Rose Ellingson
May 2014

Copyright © 2014 by Sally Rose Ellingson
All rights reserved.

# DEDICATION

I dedicate all of my hard work and effort to my beautiful daughter Kyla Anin. You give me motivation to succeed!

# ACKNOWLEDGEMENTS

I would like to acknowledge and thank my advisor, Jerome Baudry, for all of his support, guidance, and confidence in me during my dissertation work. It was a pleasure working with you. I would also like to thank all the members of my committee both for agreeing to serve on my committee and for additional roles they played in my graduate school experience. Jeremy C. Smith, thank you for taking the time and interest in my work and abilities and acting more as a co-advisor than a committee member. Daniel Roberts, thank you for immersing me in biology when I joined the Genome Science and Technology program. The opportunities you provided were instrumental to my success in an interdisciplinary career. Loren Hauser, thank you for your guidance during a very interesting and fruitful lab rotation, which provided me with additional skills and qualifications. Gregory Peterson, thank you for providing challenging and interesting computer science course work. The logic was a nice break from the uncertainties in biology!

I am also eternally grateful to Harry Richards who was instrumental in bringing me to the University of Tennessee in the first place through the SCALE-IT fellowship and his continued guidance and support throughout graduate school!

# ABSTRACT

The work in this dissertation explores the use of massive computational power available through modern supercomputers as a virtual laboratory to aid drug discovery. As of November 2013, Tianhe-2, the fastest supercomputer in the world, has a theoretical performance peak of 54,902 TFlop/s or nearly 55 thousand trillion calculations per second. The Titan supercomputer located at Oak Ridge National Laboratory has 560,640 computing cores that can work in parallel to solve scientific problems. In order to harness this computational power to assist in drug discovery, tools are developed to aid in the preparation and analysis of high-throughput virtual docking screens, a tool to predict how and how well small molecules bind to disease associated proteins and potentially serve as a novel drug candidate. Methods and software for performing large screens are developed that run on high-performance computer systems. The future potential and benefits of using these tools to study polypharmacology and revolutionizing the pharmaceutical industry are also discussed.

# TABLE OF CONTENTS

INTRODUCTION .....1
CHAPTER I Accelerating Virtual High-Throughput Ligand Docking: current technology and case study on a petascale supercomputer .....2
Abstract .....3
Introduction .....3
Virtual Screening: Existing Tools And Techniques .....5
Parallelization approaches of docking of large chemical databases .....5
Current GPU Development .....6
Virtual Screening Using Task-Parallel MPI Autodock4 .....7
Improvements in Efficiency .....7
Improvements in Accuracy .....12
Case Study .....12
Lessons Learned .....13
Additional Improvements .....13
Tutorial .....14
Possible Future Directions .....14
Conclusions .....15
Acknowledgments .....15
Appendix .....16
CHAPTER II VinaMPI: Facilitating Multiple Receptor High-Throughput Virtual Docking on High Performance Computers .....19
Abstract .....20
Introduction .....20
Methods .....23
Code Implementation .....23
Benchmarking Procedures .....27
Results and Discussions .....28
VinaMPI on Large Core Counts (EDC Scheme) .....28
Number of Tasks vs Number of Workers .....29
Comparison to Serial Calculations .....30
Conclusions and Future Work .....30
Acknowledgements .....31
Appendix .....32
CHAPTER III Polypharmacology and supercomputer-based docking: opportunities and challenges .....40
Abstract .....41
Introduction .....41
Beneficial Consequences of Polypharmacology .....42
Detrimental Consequences of Polypharmacology .....42
Repurposing .....43
Towards a Systems Biology (Network-Based) Approach to Drug Discovery .....43
Computational Docking to study Polypharmacology .....44

Chemogenomic Level Understanding of Polypharmacology .....46
Limitations of Computational Docking for Chemogenomic Level Polypharmacology.....47
Conclusions.....48
Acknowledgements .....49
Appendix .....50
CHAPTER IV Assessing Methods and Usefulness of High-Throughput MultiConformer Docking .....52
Abstract .....53
Introduction .....53
Binding Theory for Small Molecules and Proteins.....53
Studies of Active Site Flexibility.....54
Virtual Docking in Drug Discovery .....55
Existing Ways to Model Receptor Flexibility.....55
Conclusions and Future Work.....56
Appendix .....58
CONCLUSION.....59
REFERENCES .....60
VITA.....71

# LIST OF TABLES

Table 1. Changes in predocking procedure .....16
Table 2. Changes in postdocking procedure .....16
Table 3. Degrees of freedom for each library .....32
Table 4. Time-to-completion for each scheme .....32
Table 5. Load balance based on the size of the input library .....32

# LIST OF FIGURES

Figure 1. Virtual Screening as a tool to create “enriched” libraries of potential novel pharmaceuticals ....17
Figure 2. Workflow for PreDocking procedure. ....17
Figure 3. Workflow for PostDocking procedure. ....18
Figure 4. Actual time to screen one million compounds ....18
Figure 5. Even Distribution of Tasks scheme. ....33
Figure 6. Even Distribution of Complexity scheme. ....34
Figure 7. Even Distribution of Complexity worker queue. ....35
Figure 8. Input files needed for task distribution. ....35
Figure 9. Characteristic complexity for libraries of various sizes. ....36
Figure 10. DUD benchmark comparing the EDT and EDC schemes of VinaMPI. ....36
Figure 11. Results of scaling full benchmark on large core counts. ....37
Figure 12. Load balance based on the size of the input library. ....38
Figure 13. Enrichment curves for estrogen receptor agonist screen. ....39
Figure 14. Docking capabilities achieved to date. ....50
Figure 15. src enrichment. ....51
Figure 16. Hurdle for multiconformer screens. ....58

# INTRODUCTION

The pharmaceutical industry suffers from a discovery and development paradigm in which new potential drugs often fail late in the process after much time and effort has already been invested. These failed investments must be recovered in the cost of the few drugs that do make it to market. A recent analysis by Forbes predicts that the cost invested in a single drug to put it on the market is $350 million and the total cost to be recuperated for one successful drug on the market due to other drugs simultaneously failing in the discovery pipeline is $5 billion dollars (Desmond-Hellmann, 2013). Finding new ways to make drug research more cost effective and accurate would make new treatments available on the market at more affordable prices. This work focuses on virtual molecular docking, a tool commonly used in pharmaceutical research. Virtual docking predicts both the bound conformation and the strength of a drug-disease specific protein complex. The use of supercomputers is explored here in order to perform a massive number of these calculations, facilitating both the screening of large chemical databases and ensembles/libraries of proteins.

In Chapter 1, workflows and tutorials are developed to facilitate the job preparation and analysis of large virtual screens. A case study of performing a million compound virtual screen is reported and future directions for developing a screening tool that effectively handles multiple proteins are established. Chapter 2 reports on the development of new screening software that handles multiple proteins and the scaling is reported on some of the largest supercomputers available at the time. In Chapter 3, hopeful future directions of these technologies as a tool to explore polypharmacology are discussed along with the benefits this would offer the pharmaceutical community and systems-biology knowledgebase. Finally, the role of protein dynamics in ligand binding and the use of this software for an assessment of the usefulness of ensemble screening, docking into many protein conformations, is explored in Chapter 4.

All chapters represent information that has either already been published, submitted for publication, or being prepared for submission.

# CHAPTER I
# ACCELERATING VIRTUAL HIGH-THROUGHPUT LIGAND DOCKING: CURRENT TECHNOLOGY AND CASE STUDY ON A PETASCALE SUPERCOMPUTER

A version of this chapter was originally published by Sally R. Ellingson, Sivanesan Dakshanamurthy, Milton Brown, Jeremy C. Smith, and Jerome Baudry:

Ellingson, S. R., Dakshanamurthy, S., Brown, M., Smith, J. C. and Baudry, J. (2013), Accelerating virtual high-throughput ligand docking: current technology and case study on a petascale supercomputer. Concurrency Computat.: Pract. Exper.. doi: 10.1002/cpe.3070

The work and writing presented in this paper was done by Sally Ellingson.

## Abstract

In this chapter we give the current state of high-throughput virtual screening. We describe a case study of using a task-parallel MPI (Message Passing Interface) version of Autodock4 to run a virtual high-throughput screen of one-million compounds on the Jaguar Cray XK6 Supercomputer at Oak Ridge National Laboratory. We include a description of scripts developed to increase the efficiency of the predocking file preparation and postdocking analysis. A detailed tutorial, scripts, and source code for this MPI version of Autodock4 are available online at http://www.bio.utk.edu/baudrylab/autodockmpi.htm.

## Introduction

Many pharmaceuticals act by selectively binding to a specific protein and thus inhibiting a specific process relevant to a disease or illness. Because of this, the early stage of drug discovery consists of identifying potential compounds that bind to a protein of interest with a high affinity and specificity. Experimentally testing a very large number of these compounds is both costly and time consuming. Virtual high-throughput screening is an equivalent computational process that can reduce the time and cost of discovering new drugs (Shoichet, 2004; Werner, Morris, Dastmalchi, & Church, 2012).

A virtual screen is a task-parallel application in which each ligand from a large library of drug-like compounds is “docked” into a protein of interest. The docking method produces the structure of the ligand docked in the presumed active-site of the protein and a calculated affinity estimating how well the ligand binds to the protein.

The scoring functions in typical docking applications use severe approximations in order to efficiently produce results. Therefore, the calculated binding affinities rarely reproduce accurate experimental affinities. However, docking applications can be very successful at reproducing the correct “docked” conformation as well

as scoring active compounds better than inactive compounds (Gilson & Zhou, 2007). Therefore, virtual screenings are powerful tools to create "enriched" libraries (see Figure 1[1] for depiction).

Given a large library of diverse drug-like compounds, virtual screening applications can be used to relatively quickly screen the entire library and assign scores to each compound. Compounds with a high affinity (i.e. active) for the protein of interest are more likely to score higher than non-active compounds. Once all of the compounds are scored and ranked, the library can be partitioned by scores thus creating an "enriched" library that is a subset of the original and presumably contains a much higher percent of active compounds. The smaller number of compounds scoring well in the virtual screen can then be tested experimentally with a higher success rate. This approach is commonly used in the drug discovery process.

Since the true power of docking tools lies in the production of enriched libraries, the capability of distributing a large number of docking tasks and collecting and analyzing the data is extremely important. In this study, a task-parallel MPI implementation of Autodock4 is used that was developed to be used on high-performance computing (HPC) systems. A similar implementation uses the MapReduce paradigm of Hadoop in order to distribute the docking tasks on Hadoop clusters or cloud computing systems (S.R. Ellingson & Baudry, 2011). Other methods have been developed to run virtual screens on small clusters and grid platforms. However, some HPC resources require an MPI implementation in order to utilize tens of thousands of high-end processors at once, which is the aim for Autodock4.lga.MPI. These applications require a fair amount of computational expertise to set-up and run. This makes proper documentation and sharing of code and potential pitfalls important in order for these tools to aid researchers and improve the efficiency of drug development.

The main goals of this chapter is to illuminate the reader on the current technological state of high-throughput virtual screening and to share the experience of running a very large virtual screen of just over one million compounds and direct the readers' attention to the publicized code and tutorial to bring this powerful tool to a wider community of researchers. In Section 2 we give a review of the current published literature regarding high-throughput virtual screening. In section 3 we describe improvements we have made in the high-throughput virtual screening process (for the predocking and postdocking steps)

---

[1] All tables and figures are located in the chapter appendix

[2] The files outlined in red are input files for Autodock4.lga.MPI. The entire workflow is managed by custom scripts available in the online tutorial.

[3] Files outlined in red are the output files from Autodock4.lga.MPI. Custom scripts available in the tutorial manage the

when using Autodock4.lga.MPI. In section 4 we describe the case study of running a million compound virtual screening in under 24 hours and highlight some areas for future improvements. The chemical library and target protein used in this case study are part of a drug discovery collaboration and experimental validation is not the focus of the work reported here. In Section 5 we give a description of some future directions for virtual screenings.

## Virtual Screening: Existing Tools And Techniques

The increasing computational power of supercomputers allows for potentially very large chemical databases to be screened against a variety of protein targets. The software technology that is needed to leverage such computing power is the subject of much effort. A recent overview of some virtual screening tools aimed at making the task easier (typically through a graphical user interface - GUI) on mostly smaller computing architectures or individual machines, can be found in (Jacob, Anderson, & McDougal, 2012).

### *Parallelization approaches of docking of large chemical databases*

Two large docking initiatives on the EGEE Grid infrastructure have been reported (Jacq, Breton, Chen, & Ho, 2007), the first targeting malaria and the second to target Influenza (Lee et al., 2006). The study referenced in (Lee et al., 2006) screened about 300,000 ZINC (Irwin, Sterling, Mysinger, Bolstad, & Coleman, 2012) compounds against eight variants of neuraminidase from homology models on more than 2000 CPUs, generating about 600 Gigabytes of data. This study used two different Grid tools: an enhanced version of WISDOM (Lee et al., 2006) on 2000 worker nodes for a 6 week period, and a lightweight framework called DIANE. Due to Grid scheduling overhead and problems with the Grid resource Broker, WISDOM had a distribution frequency of only 38%. In addition, about 30% of the jobs failed during Grid scheduling and had to be resubmitted. DIANE had a similar failure rate but could automate the resubmission of tasks and had a much higher distribution efficiency of 80%. However, DIANE is not very scalable due to communication needs between the DIANE master and DIANE workers (which are limited to a few hundred).

Closer to the MPI development reported in the present chapter is the multi-level parallelization of Autodock4.2 (mpAD4) (Norgan, Coffman, Kocher, Katzmann, & Sosa, 2011) which uses MPI to parallelize the distribution of individual docking jobs and OpenMP to multi-thread the Lamarkian Genetic Algorithm (conformational search mechanism) in Autodock. This implementation was only tested on up to 16,384 CPU cores.

Approaches to develop better ways to connect to and utilize multiple computational resources such as High-Performance Computing (HPC) and High-Throughput Computing (HTC) systems have been reported (Riedel & Memon, 2011; Riedel et al., 2008). This work is applicable to the WISDOM (Jacq et al.,

2007) project which uses HTC resources for virtual screens and HPC resources to run Molecular Dynamics simulations to more accurately rescore the top hits resulting from the initial screen.

Computing tasks are often classified as either high-throughput computing (HTC) or high-performance computing (HPC). An emerging classification is many-task computing (MTC) and it differs from traditional HTC tasks typically done on a grid in that their metric of interest is the time to completion of the job and can take advantage of more traditionally HPC systems. Virtual screening is recognized as an important MTC application (Raicu, 2008), and there is continued interest on the best paradigms for MTC applications to run optimally on HPC architectures (Katz, Armstrong, Zhang, Wilde, & Wozniak, 2012). In (A. Peters, Lundberg, Lang, & Sosa, 2008), a virtual screening tool was developed using DOCK ("The Official UCSF DOCK Web-site: DOCK6," n.d.), that incorporated HTC features available on the Blue Gene/L supercomputer. The development included single processor optimization with compiler flag optimization and optimized libraries and optimization of the parallel implementation by increasing load balance by presorting the jobs by complexity and decreasing I/O by storing temporary files in memory. Instead of a using a parent-child (master-slave) scheme to distribute the individual docking jobs, a HTC mode available on Blue Gene/L was used in which a work dispatcher runs on the front end of the supercomputer. While the traditional MPI parent-child implementation had a major drop in performance between 8,192 (which had near linear performance) and 16,384 processors, the HTC version maintained near linear performance on 16,384 processors. Another large-scale implementation of DOCK was done on Falcon (Raicu, Zhang, Wilde, & Foster, 2008) which is an execution framework that allows for loosely coupled programs to run on petascale systems and can scale up to 160,000 cores. In a case study, after making some changes to I/O patterns, DOCK scaled to 116,000 cores while being executed through Falcon.

### *Current GPU Development*

Graphical processing units (GPUs) which have been developed for efficient rendering of high-quality graphics can be used in order to accelerate some scientific codes and are currently being incorporated into many large computing systems because of their increased available floating point operations to power ratio. In a recent review of GPU development in the computational molecular sciences (Harvey & Fabritiis, 2012), it is noted that GPU development in docking has not been as popular as other areas such as Molecular Dynamics because the important metric in docking is the time to completion of an entire screening, not the individual docking task and large GPU clusters are still rare. However, there is a trend for some of the largest supercomputer centers in the world to incorporate GPUs in their systems and utilizing them to speed-up individual docking tasks can reduce the time taken for large scale screenings if proper load balancing is ensured. In (Sánchez-Linares, Perez-Sanchez, Guerrero, Cecilia, &

Garcia, 2011), the authors use GPUs to accelerate the precalculation of potential grids. With the docking engine FlexScreen, this calculation dominates 80% of the runtime for an individual docking. In a typical screen where a large library of compounds is docked into the same structure, this information can be reused. However, when using multiple receptor files, it becomes increasingly more important to accelerate this portion of the calculation. They were able to accelerate the grid calculations by a factor of 60. Also, as mentioned above in (Guerrero, Perez-Sanchez, Cecilia, & Garcia, 2012), the authors here achieved a speed-up of 213x when accelerating the calculation of the non-bonded electrostatic interactions.

## Virtual Screening Using Task-Parallel MPI Autodock4

We report here on recent development of Autodock4.lga.MPI (Collignon, Schulz, Smith, & Baudry, 2011) and on its application to screen quickly a very large database of compounds. Autodock4.lga.MPI is a task-parallel version of Autodock4 that allows for independent and simultaneous docking of a large number of ligands into a single specified protein on up to thousands of processors at one time. The original serial version has many different input files: 1) a parameter file for each ligand, 2) 3-D coordinate files for both the ligand and protein, and 3) pre-calculated affinity maps of the protein (one for each atom type in the ligand library). The I/O is reduced in the parallel version by creating only two parameter files for the entire screening, instead of one file for each of the ligands in the screening. One parameter file contains all of the parameters specific to the docking job; these are the same for all of the ligands. The other file contains a list of the ligand specific parameters and is used for distributing the tasks. The I/O is also reduced by having a single binary file for the precalculated affinity maps using HDF5 ("The HDF Group. Hierarchical data format version 5, 2000-2010," n.d.).

### *Improvements in Efficiency*

Autodock4.lga.MPI only distributes the actual docking tasks. While this is the computationally expensive aspect of a virtual screening, when a virtual screening is scaled-up to handle millions of compounds on thousands of processors, file preparation and result analysis become the bottlenecks in decreasing the overall screening time (the time it takes to deliver the results).

#### *PreDocking*

The process in Figure 1 of (Collignon et al., 2011) shows that the standard procedure is done for predocking plus two additional steps. The standard procedure includes 1) preparing the ligand and receptor PDBQT (similar to a Protein Data Bank (PDB) file with Autodock charge (Q) and atom type (T) information) files, 2) creating the affinity maps (grids), and 3) creating the Docking Parameter Files (DPF). The PDBQT files are the coordinate files that Autodock4 takes as input for the ligands and receptor. They include charge and

atom type information needed by Autodock4 and are generated using AutoDockTools (ADT) ("AutoDockTools (ADT)," n.d.). The affinity maps are created using Autogrid (program packaged with Autodock) and one file is needed for each atom type in the ligand library. The DPF files are parameter files used by Autodock4. One file is used for each individual docking of a ligand into a receptor. These files are also created by ADT. ADT includes scripts in order to automate this process as well.

The additional predocking steps required by Autodock4.lga.MPI include running 1) make_binary.exe and 2) make_dpf_input.exe. (1) creates the single binary file used by Autodock4.lga.MPI from the ASCII affinity maps created in the standard procedure. (2) creates the two input files needed by Autodock4.lga.MPI from the DPF files previously required by Autodock4. Both of these programs were developed along with Autodock4.lga.MPI.

C shell scripts were developed along with Autodock4.lga.MPI to automate the predocking process including the standard procedure. One of the major pitfalls of this method is that it was developed to run sequentially either locally or from a login node on a large machine, such as a supercomputer. If the predocking process is done on a local machine then a large amount of data will have to be transferred to the supercomputer between the predocking and docking procedures. Since several users could be using the same login node on a supercomputer, predocking from a login node can create severe contention. Therefore, we have developed python scripts that automate the predocking process including a submission script so that the predocking can be done from a compute node. This allows the input data to be generated at the location it needs to be for the docking process without creating extra contention on the login nodes.

Another pitfall of the previous process is that it only ran sequentially. Since the same process must be done for a large amount of ligands, the predocking process is quite straightforward to parallelize. The modified process can use multiple processors to partition the ligand library and process the input in parallel. The benefit of this method is two-fold because not only is the data being prepared in parallel, but the precursor files are created in a subdirectory for each processor. This reduces the load on the file system when millions of files need to be written to and read from. The final files needed by Autodock4.lga.MPI are moved to the parent directory once they no longer need to be read from. This is where they will need to be for the actual docking procedure and they will not need to be moved again. A partial ligand specific parameter file is made during this predocking process per processor. Every ligand processed by a particular processor is included in the partial parameter file associated with that processor. These partial files are already ordered correctly (ligands with the most torsional degrees of freedom listed first – in order to distribute the tasks with the longest

compute times first and balance the workload). We include a script to combine these files which runs very quickly since the individual files are already correctly ordered.

The previous make_dpf_input.exe process used to create the ligand specific parameter file made extensive use of AWK scripts (an interpreted language common on most Unix-like operating systems). While AWK scripts can be quite powerful, they complicate this file preparation procedure. AWK is not very reusable as it is difficult to trace what the script is doing, and it also necessitates temporary files to be made during the process when the amount of data needed in this step is small enough to hold in memory until the file can be written in its final form. In our current predocking procedure, the necessary information is parsed out of the DPF files and stored in a python dictionary. This makes it very easy to sort the data and write it only once in the final form after all of the DPF files have been parsed.

Previously, the C shell predocking scripts assumed that there were already PDB (Protein Data Bank) coordinate files for every ligand in the library. This would mean that, even if the files are being prepared on the supercomputer, a large number of files will have to be collected and transferred before the predocking stage even begins. In the present updated procedure, the input is a concatenated MOL2 file which contains the entire ligand library in one file. This is the only file that needs to be transferred to the supercomputer in order to start the predocking process. Also, a MOL2 file is commonly used to store the information on a library of ligands and would more likely be the initial input when starting a virtual screening. A script is used in order to partition the library into subdirectories for the parallel predocking process as described above. To illustrate the improvements here, the million compound library that is screened in this study has a concatenated MOL2 file size of 2.3 GB and a compressed file size (gzip) of 406 MB. The directory containing the PDBQT input files of the library is 4.0 GB and the compressed and archived file size (tar) is 531 MB. While the file size of the compressed data is close, the limiting factor here is the time it takes to compress and archive the files. It takes on the order of minutes to compress the concatenated MOL2 file and hours to compress and archive the PDBQT files. In addition to that, uncompressing the archived PDBQT files on a login node of a busy supercomputer could take longer than a day.

Another key difference in the new procedure is completely separating the receptor preparation from the ligand library preparation. The previous predocking method prepared the receptor files and ligand files from the same script. However, the receptor preparation is much simpler as there is only one receptor per screening and so it is much more straightforward to prepare these files manually then to make changes in a script to adapt it to a different project. Also,

ligand libraries may be prepared separately and used in different screenings with different receptors.

The complete workflow of the new predocking procedure is given in Figure 2 and a summary of changes is given in Table 1. A concatenated file containing the 3D coordinates for all of the ligands in the screening is split into individual files for each worker processor. Each processor parses out the individual molecules in their workset and uses ADT scripts in order to create the PDBQT input files and the DPF files needed to create the list_par.dpf input file.

In addition to increasing the efficiency of the predocking process, the new procedure also aims at simplifying it. The previous C shell scripts needed many changes in order to adapt them to a new project, which required lots of code deciphering. In contrast, the new scripts require minimal changes, such as changing path names. There is a thorough tutorial that documents all necessary changes for a first time user.

### *PostDocking*

Similar to the previous predocking procedure, C shell scripts were developed along with Autodock4.lga.MPI in order to analyze the results and create a ranked list of all the ligands in the library. The calculated affinity score for each docked ligand is extracted in order to prioritize the compounds for further evaluation. Again, these scripts make extensive use of AWK and are very hard to decipher and reuse, even though modifications will always need to be made to adapt them to a new project. These scripts require many temporary files while manipulating the data to extract the interesting results. The scripts are also developed to either run locally or from a login node, and this requires either a large amount of data to be transferred that there is no need to save or extra contention on a login node.

The new postdocking procedure uses python scripts that can run locally or be submitted to a compute node via a submission script. Initially, we developed the script to parse all of the result files and keep the needed information in a python dictionary in order to quickly sort and write the ranked list once after all the results have been parsed. However, due to wall-clock time limits (the maximum amount of time that a job can execute before being terminated), we developed a second way that splits the parsing and sorting. One temporary file is kept with the needed information to maintain it if the job is killed. Once all of the results are collected, the entire temporary file is read into memory, sorted, and written in the correct order very quickly.

The current scripts utilize the python glob module which finds all pathnames matching the given pattern. The glob() function creates a list of file names that can be used to iterate through all of the files. This makes restarting the job very

easy (if it was killed due to wall-clock time restrictions, etc) because the file list can be truncated to start at the index past the last file parsed in the previous job. The postdocking procedure returns the file names for any result files with incomplete results. Another script ensures that all the result files have been created. Any missing or incomplete docking tasks can then be restarted to ensure a complete screening.

The improvements described above increased the efficiency of postdocking by an order of magnitude, so the postdocking procedure has not been parallelized. However, as with the predocking, it would be very easy to parallelize this step as well. The result files could be partitioned, then parsed and sorted on individual processors. If enough processors are used to guarantee that the postdocking would finish within a systems wall-clock time limit, then a temporary file would not be necessary. Each processor could hold the results in memory and write the partial ranked list once. Then partial sorted lists could be combined very quickly on one processor.

In addition to increasing the efficiency of the postdocking process, the new procedure aims at simplifying it and making it more thorough. Just as with the predocking, a detailed tutorial outlines all the changes that need to be made in order to adapt the scripts for a new project. The new scripts also have more features and options.

Autodock4 results cluster the individual trials for one docking between one ligand and the receptor. Trials with similar ligand conformations are placed in the same cluster and the coordinates of the conformation with the lowest energy are reported for each cluster. A ranked list can be created in two different ways: 1) using the lowest energy cluster or 2) using the largest cluster. If the docking parameters are set-up correctly, the largest cluster should be the converged result. Whereas, the previous scripts only ranked the results using the largest cluster, the new scripts create sorted lists both ways: however, the largest cluster is typically used. A docking calculation in Autodock 4 consists of many attempts of docking the ligand in various conformations inside of the specified docking box. Autodock 4 clusters these results based on the similarities of the docked poses (RMSD based) in order to evaluate the convergence of the results. The idea is that if a large enough number of evaluations are done, there will be a small number of best results.

The Autodock4.lga.MPI scripts collected only the structural information for the top specified percentage of the ranked library. However, the new procedure allows for the structure of a range of scored ligands to be collected. If a known active ligand is included in the input library, then the ranked ligands with scores similar to the known active ligand may be collected.

The previous scripts did not include any ligands in the ranked list that had a positive binding free energy. This is because Autodock4 cannot calculate the binding constant for any ligand with a positive free energy and the inhibition constant is also reported in the list and a secondary sort key in the ranking. The new method gets around this by assigning a null value of 0 for the inhibition constant for any ligand with a positive binding free energy and not attempting to parse the information from the file. This allows all of the ligands to be included in the ranked list.

Finally, the new method includes scripts and detailed instructions to create a concatenated SDF (Structure Data File) of the enriched database. SDF files are structural files that can include associated data. This method uses the <UNIQUE_ID> associated data tag to include the rank of each ligand from the screening. This enables the delivery of the results in an easy and efficient manner.

The complete workflow for the new postdocking procedure is given in Figure 3 and a summary of the changes is given in Table 2. All the result files from Autodock4.lga.MPI are parsed in order to create a ranked list of compounds. The result files corresponding to the highest ranking ligands are parsed in order to generate PDB files containing the 3D coordinates of the final docked ligand poses. Options are given in order to generate a concatenated file of the results.

### *Improvements in Accuracy*

When examining log files, we discovered some random errors of reading the binary affinity maps. These appear to be memory errors as they are not reproducible and occur at different rates on different architectures. The errors occurred while running $10^5$ ligand screenings on the Newton cluster at The University of Tennessee as well as while running a 1 million ligand screening on the Jaguar Cray XT5 at Oak Ridge National Laboratory. However, we did not receive the errors while running a 1 million ligand screening on the new XK6 architecture.

During these errors, a message is written in the result file claiming that the wrong size grid spacing is used. This is due to the binary file not being read correctly. However, using incorrect (null) input, the calculations are still performed and results are reported that appear normal during the postdocking process. If these files are not searched for explicitly, the erroneous results will end up in the analysis and may skew the final rankings. Therefore, we have developed a script to search for the errors. Detailed instructions are included in the tutorial.

## Case Study

We completed a screening of one million ligands. The details and biological results of the screening are not discussed here. In this section, the focus is on

the lessons learned while performing such a large screening and ideas to further improve the virtual screening of very large ligand libraries.

The one million compound screening ran on the Jaguar Cray XT5 Supercomputer at Oak Ridge National Laboratory using 65 thousand processors. The entire one million compound library was screened in just under 24 hours, with half of the library finishing in about 5 hours (see Figure 4). The library consisted of ligands ranging from 0 rotatable bonds to 32 rotatable bonds, with an average of 4.9. About 5% of the library consisted of ligands with at least 10 rotatable bonds (not "drug-like" molecules). Almost the entire library was screened in 10 hours with the large, extremely flexible ligands dominating the screening time, which can be seen by the right hand side vertical tail in Figure 4.

### *Lessons Learned*

#### *Millions of files is a lot!*

Handling millions of compounds is extremely time consuming; even tasks such as transferring files to a new location is very taxing on the file system. When a directory is overloaded, it takes the file system longer to find necessary files. Some of this stress can be removed by creating a directory hierarchy to store files. Also, Autodock4.lga.MPI writes the output files in the same directory as the input files. This creates twice as many files to deal with and necessitates an extra step to separate the files in order to process them more efficiently. Future virtual screening tools that are developed to handle such large jobs should use a file system hierarchy to both reduce the number of input files in one directory and separate the output from the input.

#### *Naming conventions are important*

Autodock4.lga.MPI names the result files by the following naming convention:

dockn_m.dlg

where n is the worker number (processor) and m is the ligand number (from distribution list). There are two problems with this convention. 1) If a screening is killed (due to wall-clock time, etc.) then the result files from the first round must be renamed before the screening is restarted to avoid files being overwritten. There will be at least x files with the same name where x is the number of worker processors. This could potentially lead to a lot of data loss when using thousands of processors. 2) There is no way to link a result file to a particular ligand without reading the results.

### *Additional Improvements*

In order to decrease the amount of necessary I/O, Autodock4.lga.MPI uses binary file creation and reading (with HDF5). This requires additional libraries to be available on the system in which the screen will run and increase the

complexity of the software. A better solution would be to just eliminate the need to pass such a large amount of I/O in the first place.

Autodock Vina (O. Trott & Olson, 2010) is another docking program similar to Autodock4. Using Vina as the docking engine would eliminate the need to pass the binary files. Vina does not use precalculated affinity maps, but calculates the information efficiently during the docking process. Therefore, only the ligand and receptor files need to be passed to the compute node.

### *Tutorial*

In order to make high-throughput virtual screening of large chemical libraries more accessible to researchers, we have provided a tutorial detailing the entire process. It includes the predocking (file preparation), docking (calculations), and postdocking (analysis). The tutorial was written relative to running on the Jaguar supercomputer and Lens analysis cluster (smaller cluster with shared file system with Jaguar). Compilation instructions are included in the tutorial.

http://www.bio.utk.edu/baudrylab/autodockmpi.htm

## Possible Future Directions

Flexible Receptors: Due to the large number of degrees of freedom in a protein, they are usually considered to be rigid or at least mostly rigid in molecular docking. However, proteins are not static structures and can exist in an ensemble of different conformational states, each of which a different chemical could potentially bind. Reference (Durrant & McCammon, 2011) reviews how molecular dynamics simulations (MD) can be included in the drug discovery process to take into account protein flexibility, such as the Relaxed Complex Scheme (RCS) (Cheng et al., 2008). In RCS, multiple representative snapshots are obtained from a MD trajectory and used in docking to efficiently capture the diversity of protein's conformational states. Methods to create "super" enriched lists from multiple ranked lists from different conformational states are an interesting and active area of research.

A novel drug must not only bind well to its target protein but also have limited side effects and toxicity. The earlier these adverse effects are found in the drug discovery pipeline, the more cost efficient the entire process becomes. Therefore, a means to virtually test for toxicity and side-effects before the synthesis and laboratory testing of a new chemical would be of great financial benefit. Previous studies have shown that 83% of the experimentally known toxicity and side effects for a drug target could be predicted by an inverse-docking approach of docking the potential drug into a library of proteins (Chen & Ung, 2001) and are more novel than the traditional one receptor – many ligands virtual screening (Hui-fang, Qing, Jian, & Wei, 2010).

## Conclusions

In this chapter we describe an improved predocking and postdocking procedure that works with Autodock4.lga.MPI and give details on a case study of running a million ligand library screening on the Jaguar supercomputer. Autodock4.lga.MPI, a previously published high-throughput screening application, is to the best of our knowledge the most scalable screening application that does not require a non-standard distribution framework (such as Falcon (Raicu et al., 2008)) but relies on MPI which is standard on most high-performance computers. The work here focuses on further improving the total time to complete a screening, from preparing files to obtaining meaningful results for experimental validation. These steps are often not the focus of screening applications and become the bottleneck of very large screenings.

## Acknowledgments

We thank Barbara Collignon for useful discussion and Kristina Thiagarajan for support. The National Center for Computational Sciences (NCCS) is acknowledged for a computational time grant (project BIP015). This work was financially supported by NIH grant 1KL2RR031974, Georgetown-Howard Universities Center for Clinical and Translational Science.

# Appendix

Table 1. Changes in predocking procedure

| **Previous Procedure** | **Current Workflow** |
|---|---|
| C-shell scripts to manage workflow | Python scripts to manage workflow |
| Workflow only runs serially on one processor | Workflow is parallelized to use multiple workers |
| AWK and temp files used to create list_par.dpf | Python dictionary used to create list_par.dpf |
| Takes PDB files as input | Takes concatenated MOL2 file as input |
| Ligand and receptor preparation in one workflow | Separation of ligand and receptor preparation |
| 100 thousand ligand library processed in ~10 hours | 1 million ligand library processed in < 24 hours |

Table 2. Changes in postdocking procedure

| **Previous Procedure** | **Current Workflow** |
|---|---|
| C-shell scripts to manage workflow | Python scripts to manage workflow |
| AWK and temp files used to extract results | Python dictionary used to extract results |
| Only creates a sorted list of the computed binding energies | Creates a concatenated coordinate file for the top results |
| 100 thousand results processed in ~10 hours | 1 million results processed in < 24 hours |

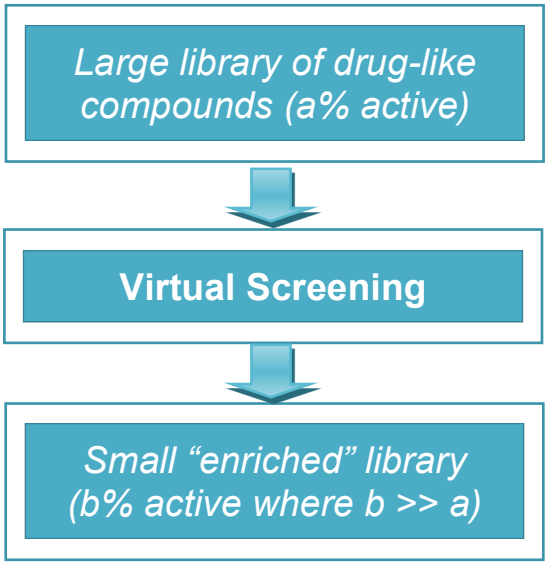


Figure 1. Virtual Screening as a tool to create "enriched" libraries of potential novel pharmaceuticals

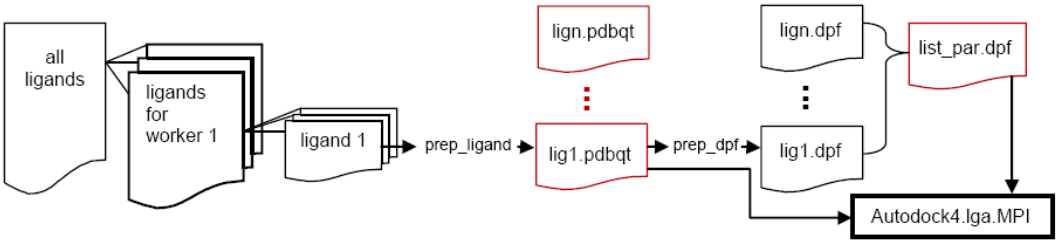


Figure 2. Workflow for PreDocking procedure.[2]

[2] The files outlined in red are input files for Autodock4.lga.MPI. The entire workflow is managed by custom scripts available in the online tutorial.

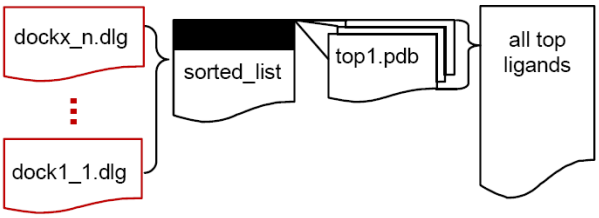


Figure 3. Workflow for PostDocking procedure.[3]

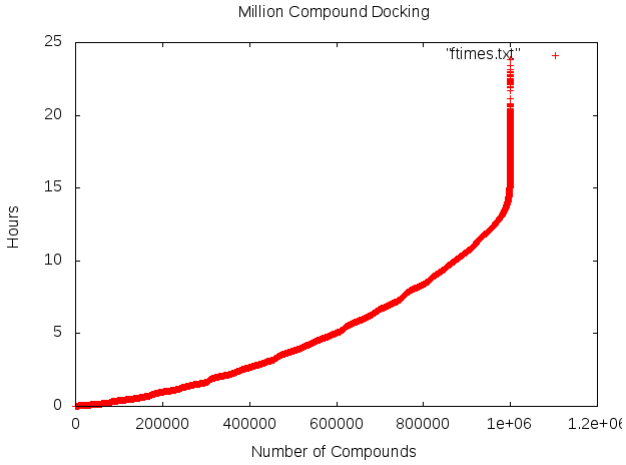


Figure 4. Actual time to screen one million compounds

[3] Files outlined in red are the output files from Autodock4.lga.MPI. Custom scripts available in the tutorial manage the workflow.

# CHAPTER II
# VINAMPI: FACILITATING MULTIPLE RECEPTOR HIGH-THROUGHPUT VIRTUAL DOCKING ON HIGH PERFORMANCE COMPUTERS

A version of this chapter was originally published by Sally R. Ellingson, Jeremy C. Smith, and Jerome Baudry:
S. R. Ellingson, J. C. Smith, J. Baudry. J. Comput. Chem. 2013, 34,2212–2221. DOI: 10.1002/jcc.23367

The work and writing presented in this paper was done by Sally Ellingson.

## Abstract

The program VinaMPI, has been developed to enable massively large virtual drug screens on leadership-class computing resources, using a large number of cores to decrease the time-to-completion of the screen. VinaMPI is a massively parallel Message Passing Interface (MPI) program based on the muti-threaded virtual docking program AutodockVina, and is used to distribute tasks while multi-threading is used to speed-up individual docking tasks. VinaMPI uses a distribution scheme in which tasks are evenly distributed to the workers based on the complexity of each task, as defined by the number of rotatable bonds in each chemical compound investigated. VinaMPI efficiently handles multiple proteins in a ligand screen, allowing for high-throughput inverse docking that presents new opportunities for improving the efficiency of the drug discovery pipeline. VinaMPI successfully ran on 84,672 cores with a continual decrease in job completion time with increasing core count. The ratio of the number of tasks in a screening to the number of workers should be at least around 100 in order to have a good load balance and an optimal job completion time. The code is freely available and downloadable. Instructions for downloading and using the code are provided in the supplementary materials of the published article.

## Introduction

According to a 2010 study, the cost of bringing a new drug on the market is estimated to be $1.8 billion and steadily rising (Paul et al., 2010). Concomitantly, according to a report by Bernstein Research (as cited in [2]) there has been a steady decrease in the number of new drugs on the market per billion US dollars spent on commercial drug research and development, and the term ”Eroom’s Law”, backwards for Moore’s Law that describes the exponential growth in computer chip power over time, has been used to describe the state of the pharmaceutical industry (Scannell, Blanckley, Boldon, & Warrington, 2012). A proposed solution to this problem calls for a change in the traditional approach to discovery in which new drug candidates fail late in the discovery pipeline resulting in great financial and time loss (Paul et al., 2010). To reduce the losses incurred by failed drugs, the industry needs to find a “quick win – fast fail” paradigm. The approach described in this article investigates ways of leveraging Moore’s Law in concert with massive parallelism to establish new R&D paradigms arresting Eroom’s Law by increasing the number of new, well-tested

drugs on the market in a quicker and more cost effective manner thus allowing more diseases to be treated at a lower healthcare cost.

There are many virtual techniques that can be used to study the interaction between a drug candidate and target protein, including extremely computationally intensive approaches of simulating every atom in the protein-ligand complex in solution. For instance, in a recent example the cancer drug dasatinib was found to bind in its experimentally determined binding pocket during an unguided molecular dynamics simulation (Shan et al., 2011). These techniques can be very insightful in determining how a small molecule interacts with its target, but they are too time and computationally intensive to be used in a high-throughput manner to effectively develop a quick win – fast fail discovery pipeline. Cost effective methods are needed that can virtually screen a large number of molecules quickly.

Virtual docking is an efficient computational process that aims at predicting the bound conformation of a protein-ligand complex and how well it binds through a scoring algorithm (Shoichet, 2004; Werner et al., 2012). Various docking programs exist and have been successfully used, such as the freely-available programs DOCK (Estrada, Armen, & Taufer, 2010), Autodock4 (Morris et al., 1998), the MPI version of Autodock4 (Collignon et al., 2011), or the more recent Autodock Vina (O. Trott & Olson, 2010). Docking applications and scoring functions have been compared in reviews (Moitessier, Englebienne, Lee, Lawandi, & Corbeil, 2008; Warren et al., 2006). The scoring functions commonly used in docking applications use significant approximations to rapidly estimate protein-ligand binding affinities and the resulting computational efficiency make these applications useful for virtual high-throughput screens in which millions of molecules can be tested quickly.

Many tools have been developed to facilitate virtual screens using a library of chemical compounds and a target protein; see reference (Jacob et al., 2012) for a recent review. Many of these tools aim to facilitate the use of virtual docking programs by non-computational laboratories by being user friendly. These tools have limited to no scalability to a large number of computer cores, and hence are limited to library sizes manageable by a small amount of computer resources. Screening tools developed to run on high-performance computers (HPC), scaling to very large computers rely on unique distribution schemes and execution frameworks. In one example a virtual screening tool developed using DOCK (Estrada et al., 2010), incorporated high-throughput computing (HTC) features available on the Blue Gene/L supercomputer (A. Peters et al., 2008). Falcon is an execution framework that allows for loosely coupled programs to run on petascale systems and can scale up to 160,000 cores. In a case study, after making some changes to I/O patterns, DOCK scaled to 116,000 cores while being executed through Falcon (Raicu et al., 2008). However, these applications

are limited to the systems that support the special features and frameworks used to distribute tasks.

In contrast to the above work, the program Autodock4.lga.MPI (Collignon et al., 2011), developed in our laboratories, is a screening tool that uses a Message Passing Interface (MPI) based distribution scheme, a programming model typically found on a large number of supercomputers and clusters. This application has been used to screen over a million compounds in less than 24 hours wall clock time using 65,536 processors (Sally R. Ellingson, Dakshanamurthy, Brown, Smith, & Baudry, 2012). Similarly, mpAD4 takes advantage of a multi-level parallelization of Autodock4.2 which uses MPI to parallelize the distribution of individual docking jobs and OpenMP to multi-thread the Lamarkian Genetic Algorithm (conformational search mechanism) in Autodock (Norgan et al., 2011). mpAD4 was tested on up to 16 thousand CPU cores with near linear scaling when using a compound library of 34,841 (and poorer scaling when using smaller libraries). A recent paper (Zhang, Wong, & Lightstone, 2013), uses a similar parent-child distribution scheme as with Autodock4.lga.MPI but with the Autodock Vina docking engine. Using this approach a nearly linear speed-up is achieved on 15 thousand cores. However, from recent experiences using a parent-child distribution scheme with Autodock4.lga.MPI (Collignon et al., 2011) a near linear speed-up was received on 8 thousand CPU cores, with decreasing performance on over 16 thousand CPU cores and no gained performance on over 65 thousand CPU cores (not published). The goal of the current work is to develop software that can take advantage of the largest supercomputers in their entirety.

Most of the screening applications developed to date focus on docking a library of drug-like molecules into one protein target. However, inverse techniques of docking libraries of chemical compounds into a library of proteins are of significant interest (Hui-fang et al., 2010), as these permit the investigation of many conformational states of a single protein thus increasing the chemical diversity of drug candidates (Amaro et al., 2008), and of the effects of a single target compound against a range of different proteins permitting the exploration of toxicity/side-effects of the drug and polypharmacology capabilities. A previous study showed that 83% of the experimentally known toxicity and side effects (off-target protein interactions) for a drug target could be predicted by an inverse-docking approach of docking the potential drug into a library of proteins (Chen & Ung, 2001). These screening techniques can be complemented by structural investigations of protein flexibility such as molecular dynamics simulations (MD), as described in (Durrant & McCammon, 2011).

If a particular chemical is able to bind to multiple proteins, the proteins are considered as interacting in chemical space. This concept of potential “target-hopping” can be very useful in drug development, and in particular for

repurposing, in which existing pharmaceuticals can be used in new clinical conditions to target proteins different than the one for which they were originally designed. A better understanding of pharmacological networks can lead to a better understanding of compound promiscuity and provide insights in the rational design of promiscuous agents (Morphy & Rankovic, 2005; Paolini, Shapland, Hoorn, Mason, & Hopkins, 2006).

The above approaches demand considerable computational power. While such power is now beginning to be available with petascale supercomputers, the corresponding software needs to be developed, implemented and tested to take advantage of the entire capability of a supercomputer, rather than just a fraction of it. One of the goals of this work is to develop a screening application specifically designed to scale on a large number of cores on supercomputing architectures in order to decrease the time-to-completion of the entire workflow of very large virtual drug screens. In addition, this work aims at making such very large scaling as 'universal' as possible by using freely available docking code and parallelization techniques.

## Methods

### *Code Implementation*

The motivation behind the development of VinaMPI was to create a platform to facilitate multiple receptor screens with a large number of compounds utilizing the largest number of processors in order to reduce the time-to-completion. Previous work from our laboratories led to Autodock4.lga.MPI (Collignon et al., 2011), aimed at distributing screens of a large number of compounds to as many processors as possible in order to decrease the overall job time. However, this program has shortcomings when applied to multiple receptor screens. First of all, Autodock4.lga.MPI jobs consist of only one protein receptor and a library of ligands. Therefore, a multiple receptor screen would require a separate job execution for each receptor in the screen. Additionally, Autodock4.lga.MPI uses Autodock 4 (Morris et al., 1998) as its docking engine which requires precalculated grids for the protein receptor to be generated. These precalculated grids are passed to the processing units (i.e. the computer cores performing the computation) efficiently in Autodock4.lga.MPI by creating concatenated binary files of all the affinity maps to be passed to the processing units and read into memory once. However, this makes Autodock4 not ideal for greatly increasing the number of receptors in a screen because a large amount of data would need to be preprocessed and additional information would need to be passed to the processing units during the virtual screen. Another limiting factor of Autodock4.lga.MPI is that it uses a parent-child scheme to distribute docking tasks in which one MPI process (the parent) distributes the tasks to n-1 MPI processes (the children) where n is the total number of computer cores used by

the job. A parent-child distribution scheme emits finely tuned load balancing since a new task can be started on each child every time the child finishes its current task. However, this scheme is limited to the number of children the parent process can communicate with before the I/O associated with the communication scheme saturates any additional performance gained by adding more children. While this scheme is efficient for a moderate number of cores, the increased efficiency obtained by using additional cores in a large screen levels out at around 65 thousand cores, which is significantly less than the capabilities of current supercomputers. Due to the added difficulties of screening multiple receptors because of the limitations of Autodock4, Autodock Vina (O. Trott & Olson, 2010), developed in the same laboratory as Autodock (referred to as Vina below) was selected as the search engine for this new application. Vina has advantages over Autodock4 such as calculating grid maps efficiently during docking and not storing them on disk, which decreases the amount of necessary preprocessing of grid files and the amount of I/O needed to start tasks on a processing unit. Clustering and ranking details are hidden which decreases the amount and frequency of program output. Vina also has fewer limitations, e.g. there is no maximum number of rotatable bonds for an input ligand, and since it is multi-threaded, each docking can potentially be more efficient. Since these features were already implemented in Vina, it was an ideal docking engine to use. In addition to the added technical benefits, Vina yields better Receiver Operator Characteristics Enrichment than the Autodock program for (Kukol, 2011). To scale Vina on supercomputing architectures, an all worker scheme was chosen to overcome the communication bottleneck of the parent-child distribution scheme. VinaMPI is a compiled C program that can be submitted as one job on leadership-class computing resources, obtaining a large number of processors in order to reduce the time-to-completion of very large screens. VinaMPI utilizes MPI since it is the *de facto* standard used on modern supercomputers (Sridhar & Panda, 2009).

### *VinaMPI: Even Distribution of Tasks (EDT)*

The Vina code was at first unmodified and an MPI wrapper was built to distribute every combination of docking tasks from a list of receptors and a list of ligands. In this VinaMPI program, every worker calculates its own set of tasks (receptor-ligand pairs for docking) based on its MPI rank, a unique identifier for each MPI process. The workset is calculated based on the start_task (first task in the list of total tasks) for each worker and the number_tasks (number of tasks for that worker). For every task_id in the workers' workset, the receptor_id and ligand_id (numbers corresponding to a receptor and ligand in the receptor library and ligand library) are calculated in the get_pair method and the protein-ligand pair is docked by the worker. This is illustrated in Figure 5. The scalability of large screens with Vina using this worker-only EDT scheme was tested.

### *VinaMPI: Even Distribution of Complexity (EDC)*

Since a worker-only distribution scheme requires all tasks to be divided at the start of a job, and different tasks have varying run times, the load balance in a worker-only scheme is not optimal (i.e. there are highly varying finishing times of each core). To see how large an improvement can easily be gained relative to naively distributing each task as in EDT, the Even Distribution of Complexity (EDC) scheme distributes jobs based on the complexity of each worker's workload. A factor that is often used to sort docking jobs by their computational complexity is the number of degrees of freedom, or rotatable bonds in the ligand to be docked. To distribute the tasks to the workers so that the workload of each worker is closer in computational time (better load balanced), here in EDC each worker receives a workload consisting of tasks with as close to the average total degrees of freedom that each worker should have if the total number of degrees of freedom in the entire screen were evenly distributed to each worker, as illustrated in Figure 6. Every task in the screen has an associated complexity (i.e. number of rotatable bonds in the ligand) and therefore the complexity of the entire high-throughput screen can be given by the summation of the complexity for each task. The combined complexity is divided by the number of workers to give an average complexity to be handled by each worker. A list of available workers is maintained and the cumulative number of degrees of freedom for each ligand in its current workset is maintained. Tasks are sorted by their complexity and then distributed to the workers one-by-one so that the most complex tasks run first on each worker. The workers are iterated over and assigned new tasks. When the current worker being iterated over still has a lower cumulative degrees of freedom than the average number of degrees of freedom plus the number of degrees of freedom for the ligand currently being assigned, that task is assigned to that worker. Otherwise, the worker list is iterated over until a worker with sufficient space for the new task is found. If there is insufficient space on any worker, the task is assigned to the worker with the lowest cumulative degrees of freedom. Once a worker reaches the average number of degrees of freedom of the current screen, the worker is deleted from the list of available workers and will not be iterated over for the remainder of the task distribution procedure. Figure 7 depicts the worker queue. The original Vina code was modified to reduce the amount of log file writing in order to reduce the overhead on the file system and make Vina more efficient to run in parallel on HPC architectures.

### *Input Files and Job Execution*

A PDBQT file is the input file used by Vina, and is similar to a PDB (Protein Data Bank) file. A PDBQT file contains the 3-D atomic coordinates of every atom of the molecule the file represents, the partial charge (Q) and the Autodock atom type (T) information. To set up a virtual screen using VinaMPI all the PDBQT files for the ligands and receptors must be generated. Scripts included with AutodockTools (ADT) are used to generate these files ("AutoDockTools (ADT)," n.d.). Python scripts were developed that facilitate the predocking procedure for

multiple ligands and receptors. The ADT scripts needed to create the ligand PDBQT files were modified in order to output the basename of the ligand PDBQT file, the number of rotatable bonds for that ligand and the number of atoms in the ligand PDBQT file in one concatenated file. From this information another script is able to generate the input file needed by VinaMPI as shown in Figure 8. The file ligand_sort.txt is a list of the basenames for all the ligand input files with a space and the number of rotatable bonds for that particular ligand. The list is sorted by highest number of degrees of freedom to lowest. The number of rotatable bonds is used to divide the tasks by complexity. The file receptor.txt (as seen in Figure 8) is a list of the receptors in the screen. The first line contains the name of each Vina parameter included for each receptor. The parameters with unique values are listed first and any parameter with a default value for the entire screen is listed at the end. If a parameter is followed by '=value' that value is used for every receptor. Each subsequent line consists of the values for each non-unique parameter value. Only the basename of the receptor file is used. This file must be manually generated or have a custom script written. This is because the center of the docking site must be specified and we currently do not use an automated tool for this. Typically, a central atom in the co-crystallized ligand from the corresponding PDB file for the receptor is used, or the geometric center of the possible binding site investigated. As already available in Vina, the 'cpu' parameter can be changed to allow each docking task to be multi-threaded.

Once all of the input files are generated, VinaMPI is called using the MPI command available on the system being used (such as aprun on many supercomputers). An example job execution is

```
aprun -n 84672 VinaMPI receptors.txt ligands_sort.txt 4 98164 receptors ligands
```

where –n is the MPI flag indicating the number of MPI tasks to be used, receptors.txt and ligands_sort.txt are paths to the input files described above, 4 and 98164 are the numbers of receptors and ligands, respectively, for this example, and receptors and ligands are paths to the directories containing the receptor and ligand PDBQT files, respectively. The output files are automatically named by the basename of the receptor file joined by an underscore with the basename of the ligand file and given a PDBQT extension. This was done to alleviate problems with the naming convention of Autodock4.lga.MPI which used a concatenation of the MPI process ID and the task ID. This was problematic both because the output could not be associated with the input files without reading the file and because if a job is restarted in the same directory, files will be overwritten with different output data. The output files are written to a directory named `out' created in the working directory of the VinaMPI job. This was also done to prevent output files from being written in the same directory as the input files, requiring extensive wall clock time to move millions of output files when screening a large database.

### *Benchmarking Procedures*

The performance of VinaMPI was benchmarked using the Directory of Useful Decoys (DUD) (Huang, Shoichet, & Irwin, 2006) database. All tests were carried out on the Kraken supercomputer at the National Institute for Computational Science (NICS). In initial tests, two DUD proteins (PDB codes: 2SRC and 1P44) and a subset of the DUD ligands (1,012 2SRC ligands) were used, totaling 2,024 docking tasks. The EDT and EDC codes were both run on 768 cores with this set to compare the load balance of the different schemes. Tests were carried out using both 'cpu=1' and 'cpu=6' in order to test both the non-threaded and the multi-threaded versions.

Following these initial tests, the same benchmark set used in Ref. (Collignon et al., 2011) were used to test VinaMPI on a larger number of cores. This set comprised the ACE (angiotensin-converting enzyme), ER_AGONIST (estrogen receptor agonist), VEGFR2 (vascular endothelial growth factor receptor kinase), and PARP (poly(ADP-ribose) polymerase) (PDB codes: 1O86, 1L2I, 1VR2, and 1EFY, respectively). These four proteins were docked with the 98,164 nonredundant known ligands and decoys given in DUD, constituting a total of 392,656 docking tasks (98,164 chemicals in four protein structures). Center points for the docking box were obtained from examining the co-crystalized ligands from the corresponding PDB files. The (x,y,z) coordinates (in Å) for the center of the docking site were : ACE: (41.315 35.065 47.682); ER_AGONIST: (6.085 -0.422 -5.791); VEGFR2: (29.884 30.491 17.465); and PARP: (38.653 22.426 20.947), relative to the coordinate systems in the corresponding PDB files. The side of the cubic box was 20 Å for all four protein targets. Tests were carried out using both 'cpu=1' and 'cpu=6' i.e. with both the non-threaded and threaded versions. The 'seed=4567' setting was kept in all docking calculations to ensure consistent results and all other Vina parameters were left at their default values. Both the non-threaded and threaded versions were run on ¼, ½, and ¾ of the Kraken machine, i.e. 28,224; 56,448; and 84,672 cores, respectively.

In order to determine the optimal ratio between the number of tasks and number of workers and to thus have a near to ideal load balance while still benefiting from a faster time to job completion by utilizing a larger core count, tests were also performed on the same four DUD proteins (PDB codes: 1O86, 1L2I, 1VR2, and 1EFY) with a varying number of ligands in the input library on 516 cores. Starting with the entire 98,164 compound library each iteration split the library in half, i.e. with ligand library sizes of 98,164; 49,082; 24,541; 12,271; 6,136; 3,068; 1,534; and 767 chemicals. The library was split by extracting every $2^{nd}$, $4^{th}$, $8^{th}$, $16^{th}$, $32^{nd}$, $64^{th}$, and $128^{th}$ ligand respectively from the original library (which was sorted by complexity), thus ensuring that each new library contained a set of

ligands with varying complexity. The number of ligands in each library with a given number of rotatable bonds is shown in Figure 9. The average total degrees of freedom (sum of rotatable bonds for each ligand in the library divided by the size of the library) are all similar (Table 3). This was done to ensure each library had a similar range of complexity and that only the size of the library was varying. Calculations were performed using 'cpu=1' (the non-threaded version in which each core is one worker).

Finally, to ensure that the database enrichments are identical to the results obtained with running the original Vina docking engine serially, the estrogen receptor agonist screen was also performed with the unmodified Vina code giving it the same seed value used in the VinaMPI screen.

## Results and Discussions

### *Load Balance (EDT vs EDC Schemes)*

In order to test whether or not distributing tasks based on the number of degrees of freedom in the ligands results in a better load balanced job than evenly splitting the tasks between the workers, the benchmark consisting of 2,024 tasks was run on 768 cores using both the EDT and EDC schemes. Figure 10 shows that the slope of the EDC non-threaded scheme is lower than that of the EDT non-threaded scheme, indicating that the load balance is better with the EDC scheme. The end point, i.e. the finishing time for the last worker, happens sooner for EDC indicating a faster overall time-to-completion.

In the case of multi-threaded calculations, also shown in Figure 10, the slope of the EDC scheme is even lower than that of the EDT scheme than it was for the non-threaded version. This shows that the load balance is improved when using fewer workers to which the tasks are distributed. In addition, the end point for the EDC calculations is again lower than it is for EDT. However, the end time for the EDC non-threaded and threaded versions are about the same (exact times given in Table 4).

### *VinaMPI on Large Core Counts (EDC Scheme)*

The scaling of VinaMPI on very large core counts was evaluated using 28,224; 56,448; and 84,672 cores (shown in Figures 11 (A), (B), and (C) respectively). Times are shown for the non-threaded version in which every core is a worker and for the threaded version in which each worker utilizes 6 cores. Figure 11(D) shows the job completion times for the benchmarks of Figs. 11(A), 11(B), and 11(C). Figure 11(D) shows that VinaMPI continues scaling on large core counts. The threaded lines exhibit a smaller slope than the non-threaded lines, indicating more even load balance. However, in Figs. 11(A) and 11(B) the time-to-completion of the entire workload is greater with the threaded version. Using 84,672 cores the threaded version finishes before the non-threaded version. The non-threaded calculations have a smaller number of tasks to number of workers

ratio, which affects the load balance and decreases the overall performance. The following section explores the optimal ratio.

***Number of Tasks vs Number of Workers***

In order to determine if there is an optimal ratio of the number of tasks to the number of workers, a set of jobs with varying sizes of input ligand libraries (i.e. varying number of chemicals) was executed on 516 cores. Figure 12(A) shows the normalized finish time for each worker. The normalization was performed by dividing the finish time of each worker by that of the first worker, such that the finish time for each subsequent worker is the ratio of time it takes to finish its tasks relative to the quickest worker. As can be seen in Fig. 12(A), the time for the last worker (also the time-to-completion for the entire workload) for the job using the entire 98,164 ligand library is only slightly more than 20% higher than the first worker. The jobs consisting of ligand library sizes of 98,164; 49,082; 24,541; and 12,271 chemicals, all have an overall job completion time of less than 40% greater than the first finished worker.

Figure 12 (B) shows how many workers are within one standard deviation of both the mean and median finishing times for each job. There is no correlation between the library sizes and the number of workers within a standard deviation of the average time. However, there appears to be a correlation between the number of workers within one standard deviation of the median finishing time and the library size. All the library sizes that finished within 40% of the time of the quickest running worker (namely, jobs consisting of ligand library sizes of 98,164; 49,082; 24,541; and 12,271 chemicals) have 75% of their workers finishing within one standard deviation of the median work time.

Figure 12 (C) shows the percent of CPU time spent idle after workers finish their workload and wait for the last worker to finish. This is calculated by subtracting the sum of all of the individual workers' time from the total CPU time used for the job (i.e. job completion time * number of cores). Figure 12 (D) shows the average wall time per task (time-to-completion divided by the number of tasks). Both of these plots show three different sections, in which the slope of the line changes (i.e. 98,164-12,271; 12,271-3,068; and 3,068-767). This shows that the corresponding calculations with ligand library sizes above 12,271 all have similar load balance (i.e. there is not a drastic increase in the amount of idle CPU time or the average wall clock time per task).

Figure 12 (E) shows the ratio of the number of tasks to the number of workers for each job. Using the library of 12,271 ligands this ratio is about 95 tasks to one worker (12,271 ligands * 4 receptors / 516 workers). Therefore, jobs consisting of about 95 or more tasks per worker gave good load balance with the benchmarks presented here. These values are summarized in Table 5.

### *Comparison to Serial Calculations*

A comparison of the enrichment curves obtained from serially running the unmodified version and from running VinaMPI is given in Figure 13. The enrichments curves are identical in both the (serial) original Vina and the VinaMPI codes described here. This shows that the vastly improved scaling obtained with VinaMPI is not obtained at the expense of a decreased enrichment performance compared to the original Vina code.

## Conclusions and Future Work

With more than 70 million chemical substances referenced in ACS Chemical Abstracts Service database ("CAS: A Division of the American Chemical Society," n.d.), up to an estimated novemdecillion ($10^{60}$) small molecules awaiting discovery (Reymond & Awale, 2012), an estimated 1,500 human drug targets (intersection of the druggable genome and disease altering genes) (Hopkins & Groom, 2002) each with potentially many druggable conformational states, and an estimated greater than 10,000 ligand binding domains (Bailey, Zanders, & Dean, 2001), the massive drug discovery space is too large to thoroughly explore experimentally. An alternative approach is to harness extensive computer power (such as cloud computing or supercomputers) in order to virtually explore this space.

The present results indicate that methods alternative to the parent-child scheme can be used to distribute a large number of Vina virtual docking tasks to a large number of cores on supercomputers. In particular, the Even Distribution of Complexity Scheme (EDC) can be used to distribute tasks and circumvent the I/O bottleneck experienced by the parent-child scheme on a large number of cores. Distributing the tasks by their complexity, as opposed to evenly distributing them, allows for better load balance of the workload and decreases the time-to-completion for the entire job. The results here suggest that for jobs with varying complexities similar to those in the present benchmarks, a ratio of around 100 tasks per worker will give good load balance. When there are more cores available for a screening job to have a 100:1 task:worker ratio, the threaded version can be used in order to improve the load balance and decrease the overall time-to-completion.

While VinaMPI is capable of scaling to large numbers of cores, there is room to further improve the load balance to reduce the total time-to-completion. Therefore, future implementations beyond the scope of the present work may include work-sharing features that allow busy workers to hand tasks off to finished workers. This would allow for an increase in performance when more compute resources are available than those that provide for a well load-balanced job. Further work can also be done to increase the efficiency of individual docking tasks through Graphics Processing Units (GPUs). GPU acceleration is becoming

increasingly important since many of the largest supercomputers are utilizing them in their architecture.

With a task:worker of about 760:1, the CPU time per docking is ~103 seconds (time-to-completion * number of cores / number of dockings or 78,294.286 seconds * 516 cores / (98,164 ligands * 4 proteins)). The Titan supercomputer located at Oak Ridge National Laboratory has 299,008 cores and therefore has nearly 26 billion available seconds of CPU time in a 24-hour period ("Introducing Titan: Advancing the Era of Accelerated Computing," n.d.). This computer power could be used to run over 250 million virtual binding experiments of similar complexity to the ones presented here in a 24-hour period.

## Acknowledgements

The National Institute for Computational Science (NICS) is acknowledged for a computational time grant (project TG-MCA08X032). This work was financially supported by NIH grant 1KL2RR031974, Georgetown-Howard Universities Center for Clinical and Translational Science.

# Appendix

Table 3. Degrees of freedom for each library

| Size of Library (number of ligands) | Total Number of Degrees of Freedom | Average Number of Degrees of Freedom |
|---|---|---|
| 98164 | 536527 | 5.466 |
| 49082 | 268270 | 5.466 |
| 24541 | 134140 | 5.466 |
| 12271 | 67076 | 5.466 |
| 6136 | 33544 | 5.467 |
| 3068 | 16775 | 5.468 |
| 1534 | 8392 | 5.471 |
| 767 | 4201 | 5.477 |

Table 4. Time-to-completion for each scheme

| Scheme | Time-to-completion (seconds) |
|---|---|
| EDT non-threaded | 555.899 |
| EDT threaded | 633.551 |
| EDC non-threaded | 408.517 |
| EDC threaded | 409.354 |

Table 5. Load balance based on the size of the input library

| Ligand Library Size | Time-to-completion (seconds) | First:Last worker finish times | % of workers finishing within stdev | | % CPU time spent idle | Wall time per task (seconds) | Task:Worker |
|---|---|---|---|---|---|---|---|
| | | | of mean | of median | | | |
| 98164 | 78294.286 | 1.239 | 75 | 75 | 13 | 0.199 | 760.961 |
| 49082 | 39420.886 | 1.258 | 72 | 75 | 14 | 0.201 | 380.481 |
| 24541 | 19983.669 | 1.292 | 69 | 75 | 15 | 0.204 | 190.240 |
| 12271 | 10130.346 | 1.332 | 68 | 75 | 16 | 0.206 | 95.124 |
| 6136 | 5281.937 | 1.442 | 70 | 73 | 19 | 0.215 | 47.566 |
| 3068 | 2705.015 | 1.517 | 68 | 72 | 21 | 0.220 | 23.783 |
| 1534 | 1555.327 | 1.999 | 72 | 71 | 31 | 0.253 | 11.891 |
| 767 | 878.065 | 2.620 | 69 | 70 | 39 | 0.286 | 5.946 |

```
get_work_set(rank)
{
  minimum_number_tasks = floor(total_tasks/number_of_workers)
  maximum_number_tasks = minimum_number_tasks + 1
  number_maximum_workers = total_tasks mod number_of_workers
  number_minimum_workers = number_of_workers – number_maximum_workers

  if rank < number_maximum_workers:
    number_tasks = maximum_number_tasks
    start_task = rank * maxium_number_tasks
  else
    number_tasks = minimum_number_tasks
    start_task = number_maximum_tasks * number_maximum_workers + ((rank –
                        number_maximum_workers) * minimum_number_tasks)
}

get_pair(task_id)
{
  receptor_id = task_id / number_ligands
  ligand_id = task_id – (receptor_id * number_ligands)
}

main()
{
  get_work_set(rank)

  for task_id in range(start_id,start_id+number_tasks)
    get_pair(task_id)
    dock(receptor_id, ligand_id)
}
```

Figure 5. Even Distribution of Tasks scheme.[4]

[4] Every worker calculates its own workset based on its unique MPI rank.

```
ligands_n = library of ligands (n is the number of ligands in the library)
receptors_m = library of receptors (m is the number of receptors in the library)
task = ligand-receptor pair
dof_n = list of degrees of freedom for every ligand in the library

get_work_set(ligands, receptors, dof)
{
        workers_j = list of worker numbers (j is the number of workers)
        dof_count_j = updated list of current total degrees of freedom for each worker
        adof = ceiling((Σdof) * m / j)

        for all ligands:
                for all receptors:
                        current_dof = dof_this_ligand
                        current_worker = workers_next
                        current_tdof = dof_count_current_worker

                        while current_dof + current_tdof > adof and unchecked workers:
                                current_worker=workers_next
                                current_tdof = dof_count_current_worker
                                if current_tdof < lowest_tdof:
                                        lowest_tdof = current_tdof
                        if all workers checked:
                                current_worker=workers_next until worker with lowest_tdof
                        current_worker.assign(task)
                        dof_count_current_worker += current_dof

                        if dof_count_current_worker >= adof:
                                workers.delete(current_worker)

}
```

Figure 6. Even Distribution of Complexity scheme.[5]

[5] Every worker calculates its own workset based on the cumulative number of degrees of freedom in its workset.

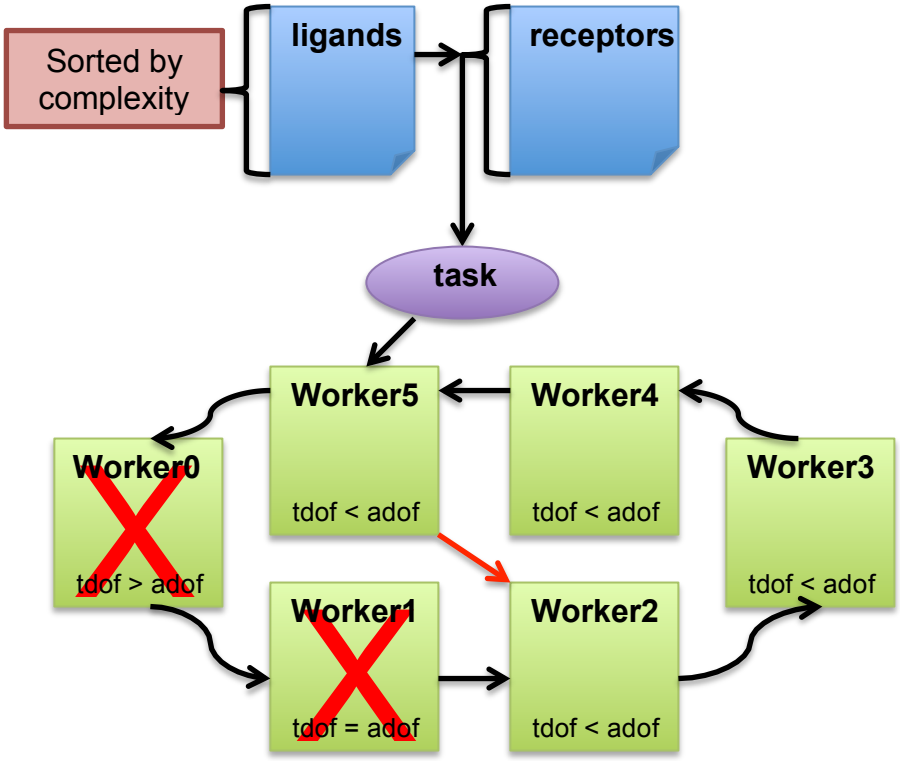


Figure 7. Even Distribution of Complexity worker queue.[6]

```
ligand1 6
ligand2 5
ligand3 2
ligand4 1
ligand5 1
   :
   :
```

ligands_sort.txt

```
receptor center_x center_y center_z size_x=20 size_y=20 size_z=20 cpu=1
receptor1 38.653 22.426 20.947
receptor2 29.884 30.491 17.465
receptor3 6.085 -0.422 -5.791
receptor4 41.315 35.065 47.682
            :
            :
```

receptors.txt

Figure 8. Input files needed for task distribution.

[6] When a worker is deleted from the list of available workers, a new iteration link is created between neighboring remaining workers to distribute tasks quickly without checking already filled workers.

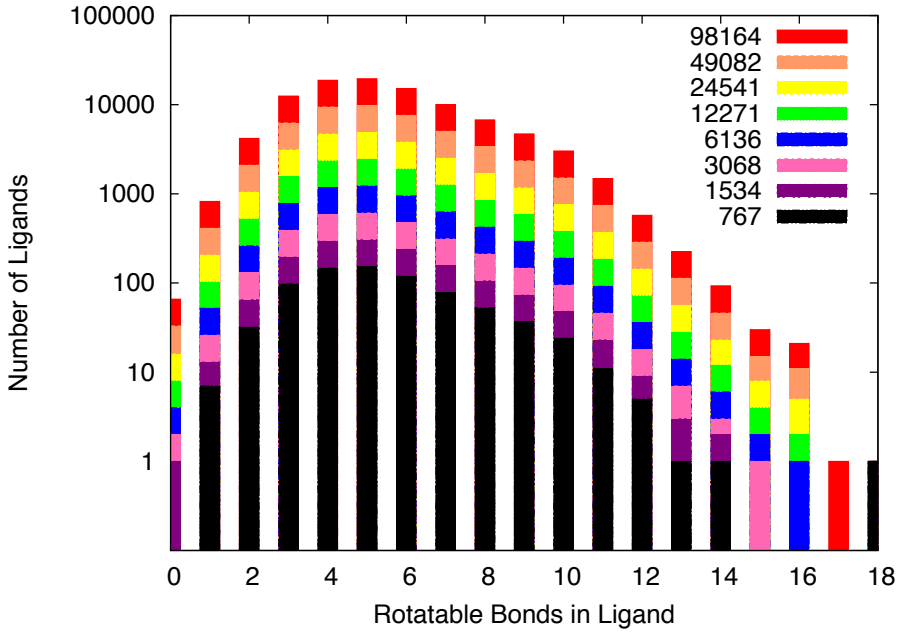


Figure 9. Characteristic complexity for libraries of various sizes.[7]

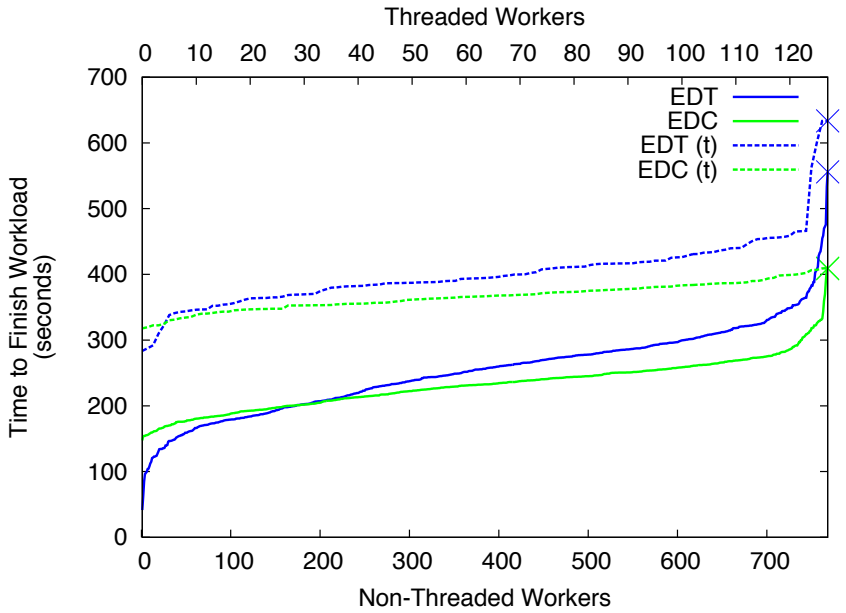


Figure 10. DUD benchmark comparing the EDT and EDC schemes of VinaMPI.[8]

[7] Number of ligands (y-axis) with a given number of rotatable bonds (x-axis) for each library of varying size (size of benchmarking libraries indicated by the coloring).

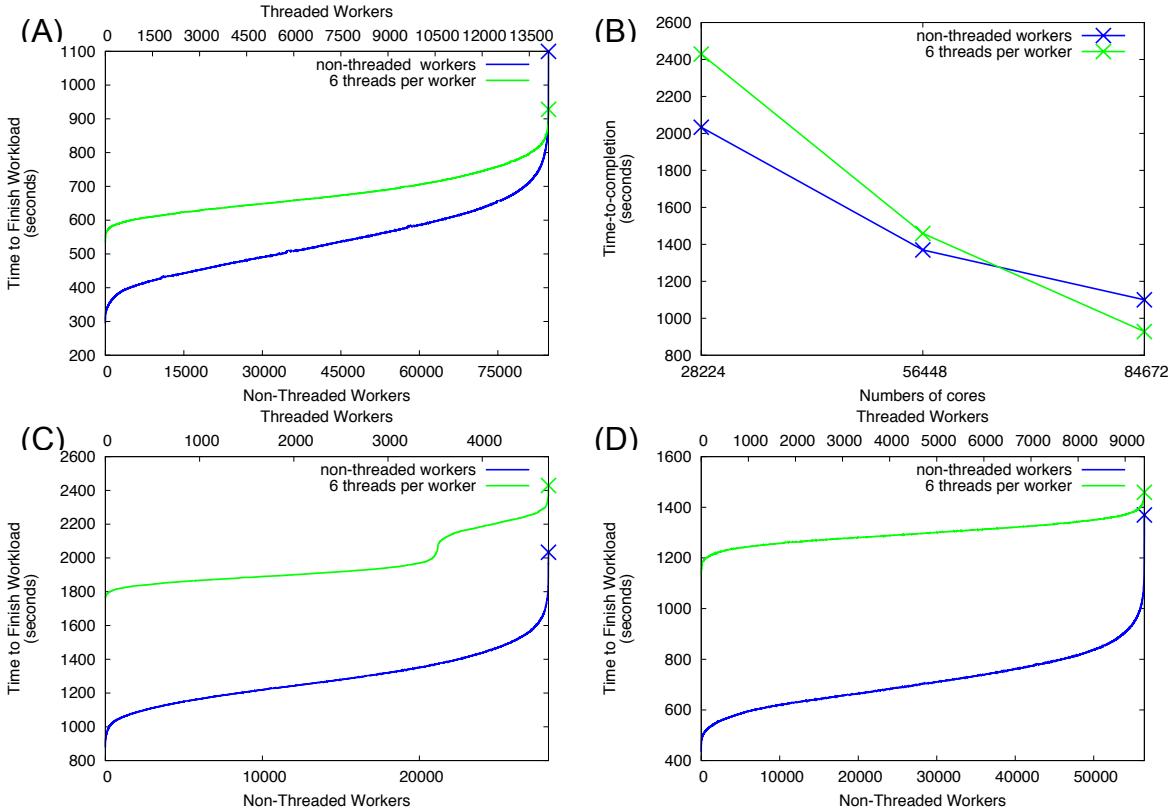


Figure 11. Results of scaling full benchmark on large core counts.[9]

[8] X-axis: workers performing the docking calculations, ordered by the time it took to complete their set of tasks. Y-axis: the time at which each worker finished its calculations. All calculations are performed using 768 cores. Solid lines correspond to non-threaded workers (i.e. using 768 workers) and dashed lines correspond to threaded workers using 6 threads each (i.e. using 128 workers). The crosses mark the time-to-completion for each job.

[9] (A), (B), and (C): time to finish the workload for each worker using the benchmark of four proteins and 98,164 ligands on 28,224; 56,448; and 84,672 cores respectively (1/4, 1/2, and 3/4 of the Kraken machine). (D): comparison of time-to-completion of the benchmark on different core counts in threaded (6 threads per worker) and non-threaded versions.

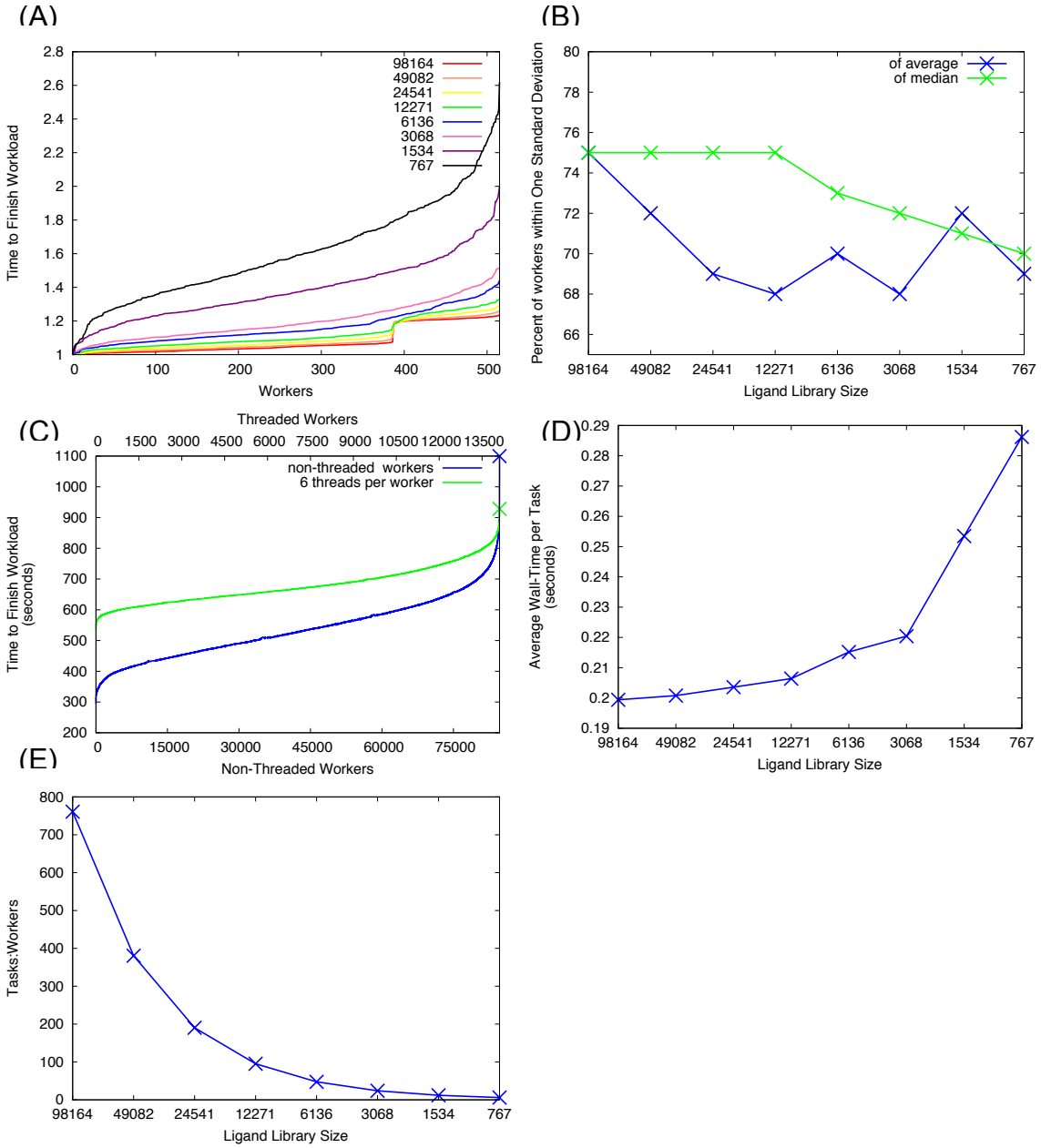


Figure 12. Load balance based on the size of the input library.[10]

[10] Results obtained on a total of 516 non-threaded workers with varying ligand sizes, ranging from the entire 98,164 ligands library, and decreasing in size by one-half each step. (A): finishing time for each worker. The data is normalized by the fastest worker in order to see the relative speeds of each worker for the different library sizes. (B): percent of workers that are within one standard deviation of both the average finishing time and the median finishing time of all the workers for varying library sizes. (C): percent of CPU time spent idle for varying ligand library sizes. (D): average wall-time per task (the average total time for completion per task) for varying ligand library sizes. (E): number of tasks to the number of workers for different ligand library sizes.

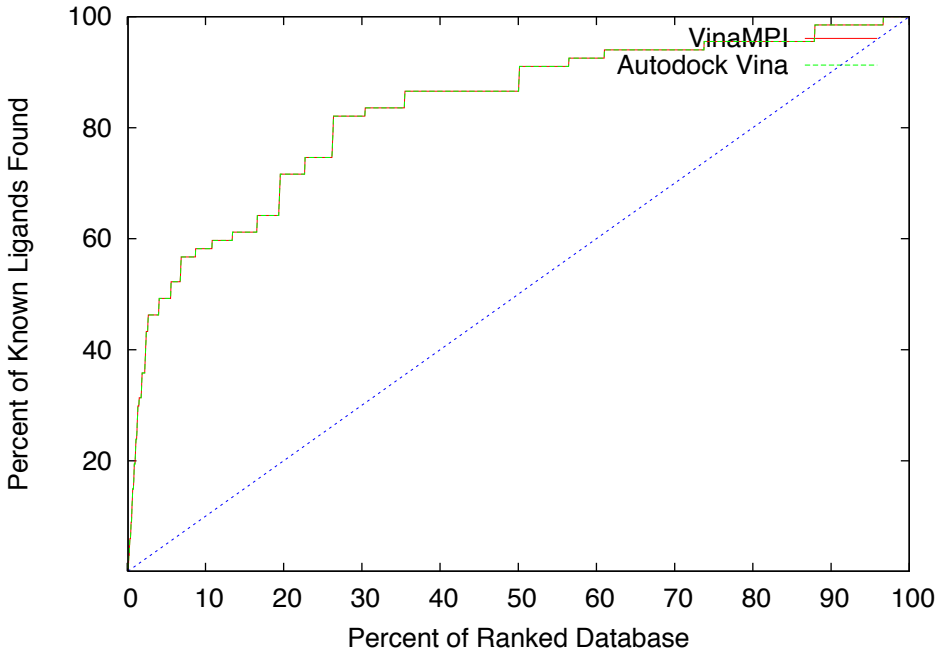


Figure 13. Enrichment curves for estrogen receptor agonist screen.[11]

[11] The red and green lines correspond to the DUD database enrichment obtained with VinaMPI (red) and Vina (green). The enrichments are identical. Blue: random enrichment.

# CHAPTER III
# POLYPHARMACOLOGY AND SUPERCOMPUTER-BASED DOCKING: OPPORTUNITIES AND CHALLENGES

A version of this chapter is accepted for publication by Sally R. Ellingson, Jeremy C. Smith, and Jerome Baudry to Molecular Simulation.

The work and writing presented in this paper was done by Sally Ellingson.

## Abstract

Polypharmacology, the ability of drugs to interact with multiple targets, is a fundamental concept of interest to the pharmaceutical industry in its efforts to solve the current issues of the rise in the cost of drug development and decline in productivity. Polypharmacology has the potential to greatly benefit drug repurposing, bringing existing pharmaceuticals on the market to treat different ailments quicker and more affordably than developing new drugs, and may also facilitate the development of new, potent pharmaceuticals with reduced negative off-target effects and adverse side-effects. Present day computational power, when combined with applications such as supercomputer-based virtual High-Throughput Screening (docking) will enable these advances on a massive chemogenomic level, potentially transforming the pharmaceutical industry. However, while the potential of supercomputing-based drug discovery is unequivocal, the technical and fundamental challenges are considerable.

## Introduction

Most of today's pharmaceuticals, or "drugs", are small organic molecules interacting with proteins in the patient's body. Hence most drug discovery effort aims at identifying, optimizing and clinically validating small molecules that have the needed chemical features to bind strongly and specifically to a protein target relevant to a specific medical condition. Polypharmacology is based on the concept that pharmaceuticals may interact with more than one different protein, and even with proteins without similar sequences and/or structures (Keiser et al., 2007). To illustrate this concept Yildirim et al. have built a drug-target network (a bipartite graph) of interactions for all known FDA-approved drugs and their targets currently on the market using data from the DrugBank database (Knox et al., 2011). Of the 890 approved drugs with known targets used to develop the drug-target network, 89% are linked with verified multiple protein targets. This indicates that the polypharmacological nature of existing pharmaceuticals is more of a rule than an exception, and suggests that future discovered molecules will most likely also possess polypharmacological properties as well.

The polypharmacological, promiscuous nature of pharmaceuticals, can have both beneficial and detrimental consequences. The former of which can be exploited to, for example, improve drug efficacy and prevent drug resistance (J.-U. Peters, 2013). In addition to the ability of chemical compounds to interact with an array of protein targets, many diseases have multiple genetic determinants, individual genetic determinants may be involved in multiple diseases, and protein function and expression are controlled by a regulatory network of other proteins (Boran &

Iyengar, 2010). Understanding of the full network of drug-target interactions and disease and regulatory pathways will permit for repurposing of approved drugs for new applications and, inversely, novel approaches to repurposing already-studied drug targets for new diseases and guidance in discovering new drugs that take advantage of beneficial secondary target interactions while avoiding adverse effects. The fundamental characterization and exportation of polypharmacological networks has the potential to change the pharmaceutical industry and lead to more drugs on the market that target new diseases, at a reduced cost and with a better understanding of their potential side-effects. Doing so will, however, present unique challenges and will necessitate state-of-the art supercomputing capacities to produce the needed data and analyze it efficiently.

## Beneficial Consequences of Polypharmacology

Functional genomic studies have shown that most single gene knockouts have little to no effect on phenotype (Giaever et al., 2002; Winzeler, 1999; Zambrowicz & Sands, 2004). The robustness of phenotypes can be explained by the existence of redundant protein functions and signaling routes (Kitano, 2007). This suggests that a polypharmacological drug may be efficacious because it is modulating multiple components of a disease pathway or multiple pathways relevant to an undesirable phenotype. Additionally, when a pharmaceutical targets multiple points in a pathway, if one point develops mutations that cause drug resistance, there remain multiple mechanisms and pathways in which the drug may still act. For instance, fluoroquinolones are prescribed as broad spectrum antibiotics at concentrations at which its two main targets (bacterial gyrase and topoisomerase IV) are inhibited, even though the inhibition of only one of them is needed to stop bacterial growth, thus preventing antibiotic resistance caused by single mutations of one of the targets (J.-U. Peters, 2013). While there is arguably a case for polypharmacological drug design, the pharmaceutical industry still largely relies on a paradigm in which one drug very selectively interacts with one target because a multitarget approach would be much more complex to design and implement. Hence, new drug design methodologies are needed in order to fully take advantage of the polypharmacological nature of drugs.

## Detrimental Consequences of Polypharmacology

Interactions between a drug and multiple proteins also result in undesirable side-effects and toxicity. Many adverse drug reactions result from drugs interacting with nontherapeutic antitargets (J.-U. Peters, 2013). For example, fenfluramine, an anorexigen, was withdrawn from the market because it led to pulmonary hypertension and heart valve damage due to the unwanted activation of serotonin 5-$HT_{2B}$ (Connolly & Crary, 1997; Hutcheson, Setola, Roth, & Merryman, 2011; Rothman et al., 2000). It has been shown that animal studies during pre-clinical trial may not give good indications of these adverse interactions in humans (Olson et al., 2000) and such adverse effects are

generally not discovered until a drug has reached clinical trial or is already on the market. With the number of different proteins in humans and the genetic variations observable in the population, a full understanding of all possible interactions through experiments and clinical testing alone is infeasible, making computational investigations particularly useful and relevant.

## Repurposing

One way to make use of some of the resources that have been lost to failed drugs is to find ways to utilize previous investments in research for new discoveries. Drug repurposing (also called repositioning or therapeutic switching) allows for drugs that have already been tested and approved as safe to be marketed and used to treat diseases that the drug was not initially developed to treat. This is possible if the intended drug targets are pleiotropic and involved in multiple disease pathologies or if the drug's off-target interactions are relevant in an alternative disease pathway. Drug repurposing is time and cost effective since a great deal of effort has already gone into developing and testing a drug that has subsequently already gone through the approval process. Repurposing may also be a mechanism to obtain pharmaceuticals that treat neglected diseases that would not otherwise create a profitable market for pharmaceutical companies, such as, for instance, in the case of Eflornithine (originally developed as an anti-cancer drug) that was repositioned and successfully used to treat human African trypanosomiasis, a tropical disease (Croft, 2005). Here again, computational tools that explore the complete polypharmacological space of existing drugs can greatly accelerate the repurposing of approved drugs.

Drug targets can also be repositioned since many drug targets are pleiotropic. This is similar to, and has overlap with, drug repositioning, but can be unique when a drug target for the disease being investigated has not yet been discovered but was previously studied as a relevant target for an alternative disease. As clinical target validation rates are low (Emig et al., 2013), computational tools to predict and identify proteins that are involved in a disease pathway, as well as candidate drug targets, are also useful for improving the efficiency of drug discovery.

## Towards a Systems Biology (Network-Based) Approach to Drug Discovery

Network-based approaches have been developed to identify drug targets, both novel and for repositioning. In (Emig et al., 2013), genes expressed differentially for a disease of interest are overlaid on a molecular interaction network and network analysis methods used to identify drug targets associated with a disease of interest. Since drug targets may highly influence a disease specific expression response, the combination of (experimental) expression data and knowledge-based data such as molecular interaction networks can give new insights on drug targets. Identified targets can then be used to develop novel drugs for a specific

disease. Alternatively, if the identified target is already used in the treatment of another disease, it can be evaluated for target repositioning. In this context, a computational framework, drugCIPHER, has been developed for predicting drug-target interactions and side-effects on a genome-wide scale (Emig et al., 2013; Simon et al., 2012; Shiwen Zhao & Li, 2010). This framework uses both pharmacological space (i.e. drug therapeutic and chemical similarities) and genomic space (i.e. protein-protein interaction networks) to predict new interactions on a large scale. The power of the method, however, is limited by the quality and incompleteness of current protein-protein interaction data needed as inputs for this approach.

These above examples show instances in which computational tools have been used to orient and facilitate drug discovery and the characterization of medically-relevant pathways by combining biomedical data, polypharmacological properties of drugs and the recognition that disease phenotypes are the result of an underlying network of interactions. Computational approaches are based on the mining and understanding of the many-to-many relationship between the set of existing (and possible) pharmaceuticals and the set of proteins defining the druggable genome.

In the future, the above information will be able to be exploited for purposes beyond the drug discovery and design process, and directly used for patient care in a clinical setting. As described in (Boran & Iyengar, 2010), an ideal therapeutic strategy would involve an individual screen for each patient that includes their mutations and genomic signature to identify misregulated elements in the underlying network that will be the target of a specialized treatment plan. This would, however, require a full understanding of the polypharmacological profiles of available drugs.

## Computational Docking to study Polypharmacology

The previous sections illustrate how the ever-growing wealth of experimental and clinically-obtained biological and medical data can be used for knowledge discovery in drug research. In drug discovery, as in most contemporary biology (and indeed as in most contemporary science), another source of data utilized originates from numerical experiments, such as, for instance, molecular simulations. There are many in silico techniques that can be used to study the interaction between a drug candidate and target protein, including extremely computationally intensive approaches of simulating the behavior of every atom in the protein-ligand complex in solution, and extracting from these simulations thermodynamic quantities, such as protein:ligand binding free energies. For instance, in a recent major computational achievement, the cancer drug dasatinib was simulated to bind in its experimentally determined binding pocket during an unguided molecular dynamics simulation (Shan et al., 2011) that sampled all possible protein:ligand interactions and described the binding pathway of a

pharmaceutical in its protein target at an atomistic level of detail. These techniques are very insightful in determining how a small molecule interacts with its target, but they are too time- and computationally intensive to be used in a high-throughput manner comparable to that used experimentally to identify new hits in libraries of chemicals. Cost effective methods are needed that can virtually screen a large number of drug-protein complexes quickly. Such a method is virtual docking, an efficient computational process that aims at predicting the bound conformation of a protein-ligand complex and how well it binds through a scoring algorithm (Shoichet, 2004; Werner et al., 2012). Autodock 4 (Morris et al., 1998) and Autodock Vina (O. Trott & Olson, 2010) are two open source and freely available docking tools commonly used in academic pharmaceutical research. Our laboratories developed high-throughput tools utilizing these docking engines and the Message Passing Interface (MPI) libraries to efficiently distribute a massive number of docking calculations to supercomputers, namely Autodock4.lga.MPI (Collignon et al., 2011) and VinaMPI (Sally R. Ellingson, Smith, et al., 2013), respectively. Docking applications and scoring functions have been compared in reviews (Sally R. Ellingson, Dakshanamurthy, Brown, Smith, & Baudry, 2013; Moitessier et al., 2008; Warren et al., 2006). The scoring functions commonly used in docking applications use approximations to rapidly estimate protein:ligand binding affinities and the resulting computational efficiency makes these applications useful for virtual high-throughput screens (vHTS) in which millions of drug-protein complexes can be tested quickly (in a matter of days or hours) on sufficiently powerful supercomputers.

In addition to being used for hit discovery (or lead optimization), vHTS, because of its potential to produce and analyze large amounts of molecular and biological data, can be used to address many of the challenges and opportunities of polypharmacology introduced above. For instance, a recent study used docking scores to relate complex drug-protein interaction profiles from DrugBank (Knox et al., 2011) with effect profiles (Simon et al., 2012). The information was combined using correlation and classification methods to generate an effect probability matrix or drug profile, and gives a probability that each drug has any given effect. While powerful, this method is limited by the need to know *a priori* the effects of the drugs. A tool is needed that can make predictions about possible side-effects of novel drugs during the early stages of drug discovery.

Polypharmacology is rationalized in (Moya-García & Ranea, 2013) as a result of protein domains serving as drug targets. It is assumed that there are a limited number of domain types that can be combined to form different proteins of different function (Kummerfeld & Teichmann, 2009). This concept implies that drugs bind to multiple proteins because they target a common domain shared between proteins that may otherwise be lacking overall structural and sequence homology. This idea has been used in (Durrant et al., 2010) to identify potential secondary protein targets by looking for binding site similarities. In this work, a

workflow which involves molecular docking into a filtered subset of the Protein Data Bank (PDB) (Berman et al., 2000) was developed to detect polypharmacological targets. The workflow includes 1) sequence homology clustering of all protein chains in the PDB, 2) selection of one representative structure from each cluster to create a subset of the PDB in which each structure is at least somewhat dissimilar, 3) assessment of binding site similarity between potential binding sites in each of the structures in the PDB subset and the known target, and 4) docking of the drug candidate into the structures containing similar active sites. This approach has led to the identification of secondary targets for an inhibitor of TbREL1 from *T. brucei*, the causative agent of African sleeping sickness. This should be implemented early in the drug discovery pipeline, before lead optimization, in order to identify potential undesirable secondary targets and optimize the specificity of the lead molecule. The challenges of this approach are 1) the very high number of docking calculations needed to be performed, 2) the introduction of false-positives due to shortcomings in the docking and scoring algorithms, 3) the dependence on sequence-homology clustering to reduce the number of protein structures to be processed because of computational limitations, and 4) the ability to scale this solution to a library of compounds, and not just one candidate compound. To overcome these limitations vHTS/docking tools are needed that can dock libraries of drug candidates into large numbers of protein structures with reasonable accuracies.

## Chemogenomic Level Understanding of Polypharmacology

Chemogenomics is the systemic study of the effects of large libraries of drug compounds against a wide variety of macromolecular targets (di Bernardo et al., 2005; Faulon, Misra, Martin, Sale, & Sapra, 2008; Rognan, 2007). Cerep, a biotechnology company, developed BioPrint, a suite of proprietary data and analysis tools to assist in drug discovery (Krejsa et al., 2003). They provide pharmacological activity data between their library of in-house chemical compounds and a number of protein targets. This binding affinity data can be clustered to identify classes of proteins that interact with similar compounds. This clustering by pharmacological activity is used to identify “hotspots” of therapeutic and off-target effects of different compounds. The ability to produce such data on a chemogenomic scale would not only be invaluable to the pharmaceutical industry but it would also lead to a better understanding of polypharamcology, and in combination with systems biology, a better understanding of disease pathology and biological mechanisms of diseases.

There are over 21 millions commercially-available molecules that can be used in screening for drug candidates in the ZINC database (Irwin et al., 2012). Considering also the chemistry yet to be synthesized, an estimated novemdecillion ($10^{60}$) small molecules are theorized to exist in the chemical universe (Reymond & Awale, 2012). In addition, there is about 1,500 human drug targets, representing the intersection of the druggable genome and disease

altering genes (Hopkins & Groom, 2002) and as many as 10,000 ligand binding domains (Bailey et al., 2001) - without including bacterial or viral protein targets. This creates a super-massive drug discovery space that cannot be explored and validated using experimental screening approaches. Only contemporary supercomputing power has the potential to serve as an exploratory vessel.

## Limitations of Computational Docking for Chemogenomic Level Polypharmacology

The power of vHTS/docking to be successfully used for hit discovery has been demonstrated in many studies involving relatively small scale projects (low number of targets, relatively low number of drug candidates) (Durrant et al., 2010; Jenwitheesuk, Horst, Rivas, Van Voorhis, & Samudrala, 2008; Kinnings et al., 2009; Rognan, 2007; Simon et al., 2012; Wei et al., 2008). With today's computational power and docking technologies such as developed in our laboratories (Sally R. Ellingson, Smith, et al., 2013), several millions of compounds can be virtually screened in only one day. Figure 14 shows the evolution of docking capabilities achieved by our laboratories to date. Our docking technology Autodock4.lga.MPI, based on Autodock4 was able to perform 300,000 dockings in a 24 hour period while utilizing 8k processing cores (Collignon et al., 2011). By increasing the core count to 65k, the performance per core is reduced but this method successfully screened one million compounds in a 24 hour period (Sally R. Ellingson et al., 2012; Sally R. Ellingson, Dakshanamurthy, et al., 2013). Our more recently developed VinaMPI approach focused on the ability to scale the docking program Autodock Vina at larger core counts. In benchmarks this code ran on 3/4$^{th}$ of the Kraken supercomputer (i.e., on 85k cores) with a continued decrease in time-to-completion of the job (Sally R. Ellingson, Smith, et al., 2013). Recent improvements on the task-to-worker ratio mean that we estimate that nearly 40 million compounds can be screened on the Department of Energy's Titan Supercomputer, presently the most powerful supercomputer in the United States, using 180k cores in a 24 hour period.

However, in silico vHTS still has significant drawbacks. The scoring algorithms do not always generate scores that correlate well enough with experimentally measured binding affinities (Gilson & Zhou, 2007). When millions of compounds are being processed, the number of false positives can be in the thousands. In addition, to reduce the computational complexity of the problem, protein structures are usually kept rigid, or mostly rigid, which essentially limits these numerical experiments to the investigation of – at best – an "induced fit" binding mechanism or – at worst – a "lock-and-key" oversimplification of protein:ligand binding mechanisms. Approaches that sample efficiently the dynamic flexibility of many protein targets are needed to investigate "conformational selection" binding mechanisms in which drug candidates bind in a "selected" protein conformation otherwise accessible at room temperature.

Areas for potential improvement in scoring functions include more advanced potential energy models and better incorporation of solvent effects and configuration entropy. Another approach is to perform more computationally rigorous free energy methods on top scoring vHTS docked compounds to generate more accurate scores and weed out false positives. Reviews that address these directions include (Gilson & Zhou, 2007; Michel & Essex, 2010; Pohorille, Jarzynski, & Chipot, 2010; Shirts, Mobley, & Chodera, 2007).

The dynamics of protein targets, controlling many biological processes such as molecular recognition and catalytic activity, may be obtained from molecular dynamics simulations. While all atom simulations of large proteins are very computationally expensive, the ability to efficiently model active site flexibility can greatly improve virtual docking and indeed allow for a “conformational selection” binding mechanism to be included in the virtual screening process. This has been conceptualized in (Lin, Perryman, Schames, & McCammon, 2002) and has led to successful applications in which potential drug candidates were identified that would not have been found through traditional virtual screenings using only a static, experimentally-solved structure of the protein, as demonstrated in (Amaro et al., 2008). These alternative conformations can also be used to find novel binding sites not existing in the crystal structure (Wassman et al., 2013). When dealing with large chemical databases of potential drug candidates, our laboratories have also observed that the use of selected snapshots from a molecular dynamics simulation of a protein target leads to significantly improved database enrichment over that obtained using only a static (crystal) structure (see Figure 15 for an example using human tyrosine-protein kinase c-src (PDB ID 2SRC) and its set of ligands and decoys from the Directory of Useful Decoys (DUD) (Huang et al., 2006)). However, the derivation of a method for extracting snapshots that represent conformational states relevant to drug binding is still an active area of research.

## Conclusions

The promiscuous (polypharmacological) nature of drugs can be exploited to both repurpose existing drugs and design better, more effective drugs. However, the search space of all drug possibilities and protein targets is too large to thoroughly explore experimentally. Efficient and accurate computational methods for exploring this space could revolutionize the pharmaceutical industry. In this regard, virtual docking holds great promise as a lynchpin of the future drug repurposing pipeline. As advances are made in docking and scoring methods, the combination of the massive amount of interaction information that can be generated via simulation and extreme computational power available with supercomputers with ever growing sources of genomic, disease and drug profile data will pave the way for a new generation of pharmaceutical discovery and personalized medicines (Fernald, Capriotti, Daneshjou, Karczewski, & Altman, 2011).

## Acknowledgements

For computational time: National Institute for Computational Science (NICS) contract grant number: TG-MCA08X032; for sponsorship and funding: NIH contract grant number: 1KL2RR031974.

# Appendix

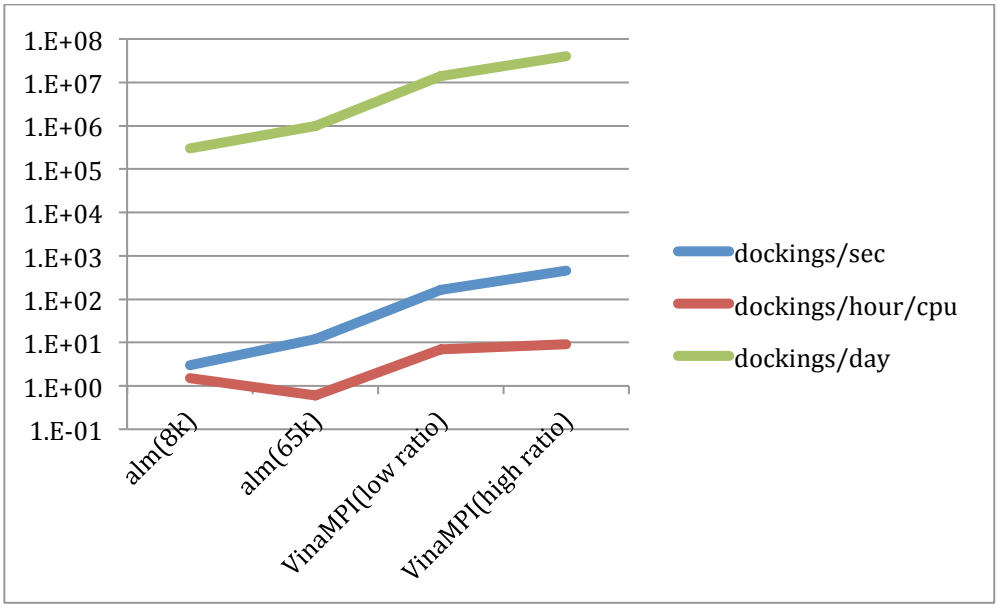


Figure 14. Docking capabilities achieved to date.[12]

[12] The y-axis is number of dockings per different units (blue line: seconds; red line: hours/cpu; green line: day). The x-axis represents different docking technologies and job set-ups. alm(8k) and alm(65k) represent autodock4.lga.MPI (Collignon et al. 2011), using the corresponding core counts. VinaMPI(low ratio) represents VinaMPI on 85k supercomputer cores with a low task-to-worker ratio and VinaMPI(high ratio) represents VinaMPI on 180k supercomputer cores with a high task-to-worker ratio (See Ellingson et al., 2013).

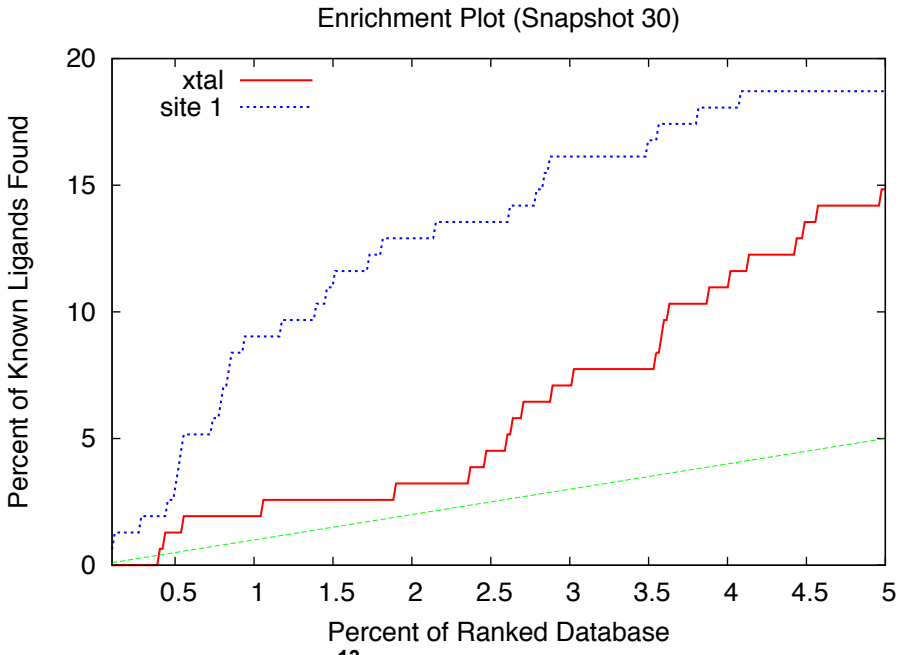


Figure 15. src enrichment.[13]

[13] The percent of active compounds identified in the ranked compound library after docking. Red: enrichment using the crystal structure; Blue: enrichment using a snapshot obtained from a molecular dynamics trajectory; Green: random enrichment

# CHAPTER IV
# ASSESSING METHODS AND USEFULNESS OF HIGH-THROUGHPUT MULTICONFORMER DOCKING

A version of this chapter is being prepared for submission by Sally Ellingson, Jerome Baudry, and Jeremy C. Smith to a special issue in honor of William L. Jorgensen Festschrift in The Journal of Physical Chemistry B.

The work and writing presented in this paper was done by Sally Ellingson.

## Abstract

In this chapter, large-scale ensemble docking is investigated using a subset of proteins from the Directory of Useful Decoys (dud.docking.org). Molecular dynamic trajectories are obtained for each protein and an ensemble of representative conformational structures are extracted from these trajectories. Dockings are then performed using these benchmark sets in order to determine if commonalities can be identified among protein snapshots with the highest enrichment factors, snapshots that identify the most active compounds in the top portion of their ranked databases. We did not identify solid rules for extracting high-scoring protein conformations. However, for the set of proteins used in this study, we did find that some snapshots always perform better than the crystal structures and the use of snapshots always increases the diversity of identified compounds.

## Introduction

Proteins are not static structures. Protein flexibility plays an important role in many biological processes such as molecular recognition and catalytic activity. Studying these processes is important for a number of fields, including drug discovery. Simulations are often used to gather dynamic protein information that cannot be obtained from experimentally solved structures. However, all atom simulations of large proteins are very computationally expensive since they have so many degrees of freedom. In order to efficiently model protein flexibility, known information can be used to reduce this search space. The ability to efficiently model active site flexibility without a significant loss in accuracy would greatly improve important tools such as virtual docking, a computational tool extensively used in the drug discovery process.

### *Binding Theory for Small Molecules and Proteins*

The lock-and-key model of ligand binding was first proposed in 1894 by Fischer (Fischer, 1894). Since then it has been repeatedly demonstrated that proteins undergo a range of motions upon ligand binding. The theory was later replaced by Koshland's induced-fit model, proposed in 1958 (Koshland, 1958), in which the ligand induces conformational change in the protein. In the late 1990s, researchers proposed a new theory in which a protein fluctuates between multiple low energy states, even when unbound. A ligand would be capable of binding to any one of these states in which it has favorable interactions and stabilize that conformation. The conformational selection and induced fit models are not mutually exclusive. It is likely that both effects contribute to binding. A

protein fluctuates between multiple low energy conformations. A ligand binds and stabilizes a subpopulation of those states. After initial binding, small induced fit changes make the complex even more stable.

The premise behind using small molecules as a drug is the idea that although a receptor has evolved to recognize a specific ligand, better binding ligands can be identified. So, a binding site can accommodate many molecules and thus "should" rearrange itself with little penalty to make this accommodation. A ligand is a good binder if it has a moderate binding affinity to the lowest energy protein conformation or if it has a high binding affinity to a less populated conformational state. The best solution, although probably less frequent, is a ligand that binds with a high affinity to the lowest energy protein conformation. This case would demonstrate the historic view of a lock-and-key binding mechanism (Carlson, 2002).

Molecular flexibility can contribute to a favorable change in free energy of binding in two different ways: a favorable change in enthalpy by optimizing noncovalent interactions between the ligand and receptor or a minimization in the decrease in entropy by increasing the flexibility in regions of the protein or ligand. Protein-ligand complexes undergo a wide range of motions including small changes in binding site residues to large scale motions of entire protein domains. There are some cases in which conformational rearrangements are so great that binding is better linked with protein folding (Durrant & McCammon, 2010).

### *Studies of Active Site Flexibility*

Many studies have investigated the issue of active site flexibility from different directions. In a thorough study of the Protein Data Bank (Berman et al., 2000) in 2000 (Najmanovich, Kuttner, Sobolev, & Edelman, 2000), Najmanovich estimated that 85% of the time 3 or fewer residues are involved in conformational change in the active site upon ligand binding. They found the following order of amino acids and their propensity for side-chain flexibility: Lys > Arg, Gln, Met > Glu, Ile, Leu > Asn, Thr, Val, Tyr, Ser, His, Asp > Cys, Trp, Phe. When normalizing for the number of rotatable bonds, the differences narrowed, but the order was preserved. Conformational changes between apo-structures of the same protein sequence were more rare but showed the same order as given above, suggesting this flexibility scale is an intrinsic property of the amino acids. In this study, the authors were looking for changes in rotameric state. They looked at the side-chain dihedral angles of the binding pocket residues and used 45°, 60°, and 75° as the threshold values denoting change. There was a small difference in the number of flexible residues found with the different thresholds, but the same probability trends were seen.

B-values or crystallographic temperature factors, represent the smearing of electron density of an atom around an equilibrium point. This is a result of

thermal motion within the crystal or positional disorder if the atom in in multiple states in the protein. Therefore, B-values can be used to obtain insight about protein flexibility. In a study looking at 69 unrelated apo-enzymes, the active site residues consistently have lower B-values then non-active site residues. This holds true when taking into account the secondary structure (α-helical, β-sheet, or random coil), protein region (interior or surface), as well as when including near neighbors to annotated active-site residues which most likely are not catalytic, but aid in binding (Yuan, Zhao, & Wang, 2003). A large-scale analysis of B-values resulted in a method to predict protein flexibility from sequence alone (Schlessinger & Rost, 2005). This method was able to reproduce the same conclusion about active site flexibility.

In a structure-based thermodynamic stability analysis of 16 structurally non-related proteins of distinct functions, all of the proteins were found to have regions of high-stability and regions of low-stability (Luque & Freire, 2000). Low stability regions are often loop regions which become stable after ligand binding. High-stability regions are often characterized by the catalytic residues.

In a study similar to a rotamer survey (S Zhao, Goodsell, & Olson, 2001), paired proteins were used to look at the flexibility within a protein, not just across distinct proteins. This studies highlights the importance of both small side-chain fluctuations and changes in rotameric states. The study was not restricted to active site residues. The flexibility scale they found for exposed residues was Ser > Gln, Glu > Met > Lys > Arg > Leu > Val > Asn > Asp > Thr > Ile > His > Trp > Phe, Tyr > Cys. This scale was calculated by the range of motion of the χ1 angle for 90% of the pairs.

### *Virtual Docking in Drug Discovery*

The goal of virtual docking is to predict the most stable conformation of a protein-ligand complex and assign a binding energy or score to the conformation. Typically, docking programs are designed to be efficient and robust in order to screen a large number of different molecules against a given protein. Therefore, the important aspects of a docking program are the efficient exploration of the conformational space of the complex and a suitable scoring function in which to evaluate the sampled poses. Initially, docking programs treated the ligand and receptor both as rigid for simplicity. Currently, ligands are usually modeled as fully flexible and random or discrete dihedral rotations are made to its relatively few rotatable bonds. The receptor is often still modeled as rigid or very selectively flexible by specifying in advance one or a few side-chains that can rotate and the states in which they can adopt.

### *Existing Ways to Model Receptor Flexibility*

Some of the approaches already used to incorporate protein flexibility in docking include soft docking, rotamer exploration, and MPS (multiple protein structures).

Soft docking uses a rigid protein but allows for some overlap of the ligand and protein in the scoring function by reducing van der Waals penalties at short distances to dock to soft structures or reducing the van der Waals optimal distance and thus uniformly enlarging the binding site. This is very efficient since it is only a change in the scoring function, but it only allows for minor side-chain movements. Some known issues with soft docking are that the ligand and receptor often interact too tightly and the “soft” region becomes too large, inhibiting the true complex from being able to form (Cavasotto, Orry, & Abagyan, 2005). The use of rotamer libraries to model the side-chain movements restricts possible conformations to those in the library. Studies have reported that even very complete rotamer libraries fail to sample side-chain conformations finely enough to produce collision free structures in test complexes where only side-chain movement is expected. However, this study also showed that biasing smaller rotations underestimates changes in rotameric states (Zavodszky & Kuhn, 2005). Using multiple structures for one protein, either multiple experimentally determined structures or computationally predicted structures, allows for any kind of conformational change but is also limited by the discrete number of structures chosen (Totrov & Abagyan, 2009). Molecular Dynamic (MD) simulations are an appealing method to generate a full continuum of structures. It is typically limited to at most the low-microsecond timescale which may not sample all conformations and increases the computational complexity of the problem. Multiple structures can be averaged to find a consensus structure or docked to individually. Consensus structures may not represent a true state of the protein and treating each structure separate considerably increases the computational time. Another hurdle when individually docking into multiple structures is identifying ways to increase your true positive rate of computationally identifying active compounds without increasing the false-positives and false-negatives at the same rate (this is depicted in Figure 16). A way of identifying a smaller number of highly druggable conformations prior to docking would circumvent this problem. Energy refinement techniques allow for a full spectrum of motions post-docking. However, the local minimum closest to the initial docked pose may not always be the global minimum and relies on force fields that may not be accurate.

Many methods to sample protein flexibility are hybrid in nature and include different aspects of the methods listed here. They are often not tested and verified on a large scale and do not get incorporated into easy-to-use software packages that can be utilized by the pharmaceutical community.

## Conclusions and Future Work

In this chapter, the importance of considering protein dynamics in ligand docking is discussed. Since proteins may exist in many different druggable states, an efficient computational method to identify and incorporate these states into the docking process could greatly increase the translational usefulness of docking for

drug discovery. A trajectory of protein motion obtained from molecular dynamics is a computational tool that can be used to study protein conformations. However, a means of identifying which frames (snapshots) from a trajectory are the most pharmaceutically relevant is still an open problem. Ongoing and future work will involve the use of the developed software to attempt to identify characteristics of protein conformations extracted from molecular dynamics trajectories that correlate with enrichment scores obtained when docking into the particular conformation.

# Appendix

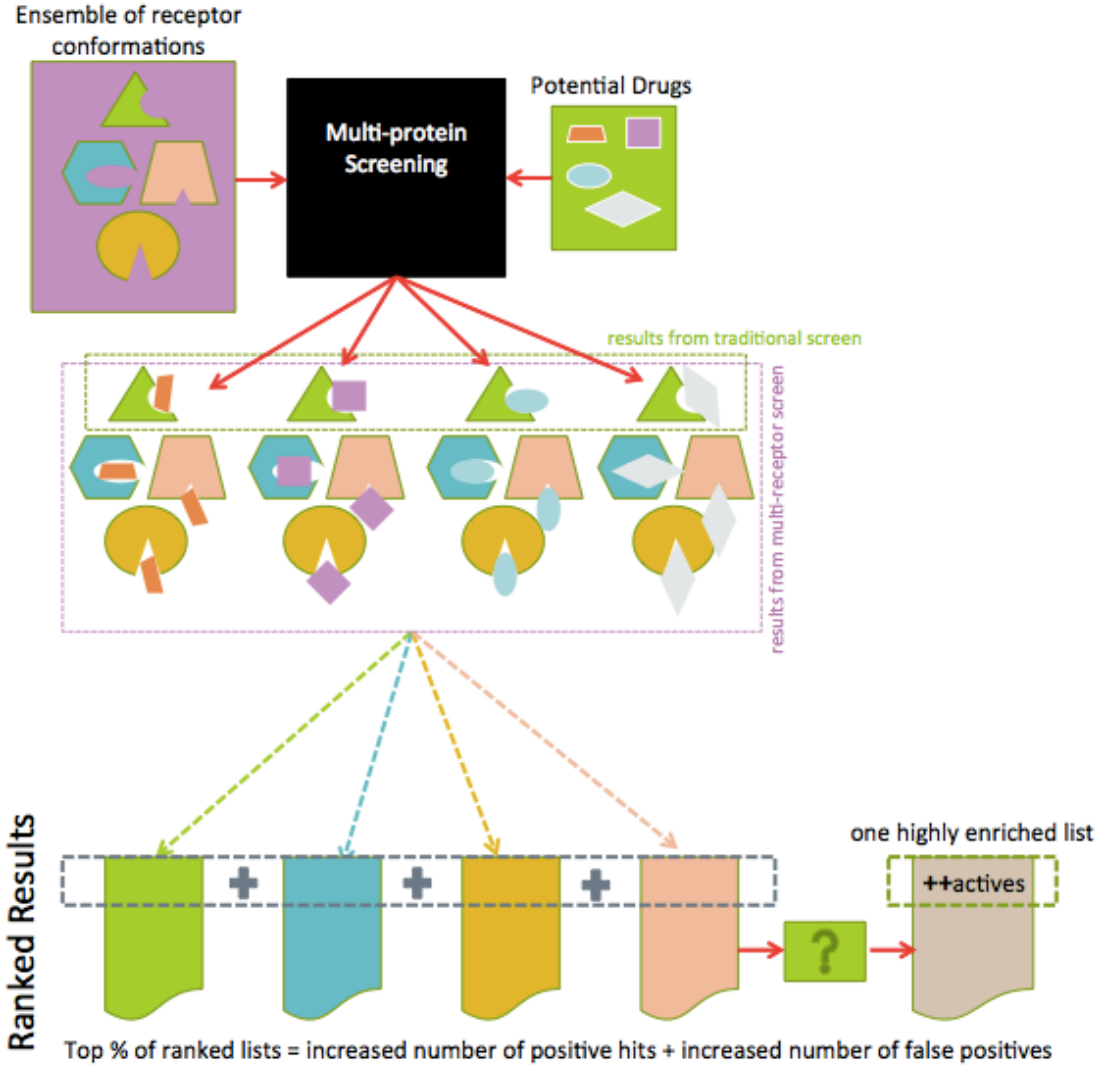


Figure 16. Hurdle for multiconformer screens.

# CONCLUSION

The use of supercomputers as a tool for computational drug discovery by means of massive virtual screenings was explored in this work. In Chapter 1, a review was given about the state-of-the-art of large-scale virtual screens and the development of workflows to facilitate large screens was presented. Also, a case study was presented of performing a one million compound library screen. While this screening far surpassed the abilities of many other high-throughput screening technologies, the software used could only scale to 65,000 computer cores, only a fraction of the current capabilities. Additionally, this software was developed to work with only one receptor at a time, but multi-receptor screenings have received far less attention and may provide new opportunities for drug discovery if they can be performed at a large scale.

Based on lessons in this experiment, new software was developed and reported on in Chapter 2. The purpose of this software was to utilize the largest number of computer cores to take advantage of massive computational power and to easily incorporate multiple receptors in a screening. The newly developed software, VinaMPI, successfully ran on the Kraken supercomputer using 85,000 cores with a better time-to-completion performance. It later ran on 180,000 cores of the Titan supercomputer (reported in Chapter 4). VinaMPI also easily and efficiently incorporates multiple receptors in a screening.

In Chapter 3 the future directions and potential of this work is explored. Polypharmacology, the ability of a drug to interact with multiple proteins, can be exploited to repurpose drugs and develop better drugs. Studying the network of drug-protein interactions can also lead to a better understanding of system biology. Docking is a promising tool to study polypharmacology and has been done already on a smaller scale. With the ever-growing computational power at our fingertips and the refinement of computational methods, these tools could revolutionize the pharmaceutical industry.

Finally, in Chapter 4, the role of protein dynamics in ligand binding is explored. Since proteins are not static structures in our bodies, they may exist in many different states, each of which may be druggable and preferred by different compounds. Therefore, docking into a static crystal structure may not produce the best results for drug discovery. While the acceleration and accessibility of high-throughput multiconformer docking provides a tool to revolutionize the pharmaceutical industry, a better understanding of the limits and usefulness of these methods is still needed.

# REFERENCES

Amaro, R. E., Schnaufer, A., Interthal, H., Hol, W., Stuart, K. D., & McCammon, J. A. (2008). Discovery of drug-like inhibitors of an essential RNA-editing ligase in Trypanosoma brucei. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(45), 17278–83. doi:10.1073/pnas.0805820105

AutoDockTools (ADT). (n.d.). Retrieved from http://mgltools.scripps.edu/

Bailey, D., Zanders, E., & Dean, P. (2001). The end of the beginning for genomic medicine. *Nature Biotechnology*, *19*, 207–209. doi:10.1038/85627

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., … Bourne, P. E. (2000). The Protein Data Bank. *Nucleic acids research*, *28*(1), 235–42.

Boran, A., & Iyengar, R. (2010). Systems approaches to polypharmacology and drug discovery. *Curr Opin Drug Discov Devel.*, *13*(3), 297–309.

Carlson, H. a. (2002). Protein flexibility and drug design: how to hit a moving target. *Current opinion in chemical biology*, *6*(4), 447–52. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/12133719

CAS: A Division of the American Chemical Society. (n.d.). Retrieved from http://www.cas.org/content/at-a-glance

Cavasotto, C., Orry, A., & Abagyan, R. (2005). The Challenge of Considering Receptor Flexibility in Ligand Docking and Virtual Screening. *Current Computer Aided-Drug Design*, *1*(4), 423–440. doi:10.2174/157340905774330291

Chen, Y. Z., & Ung, C. Y. (2001). Prediction of potential toxicity and side effect protein targets of a small molecule by a ligand-protein inverse docking approach. *Journal of molecular graphics modelling*, *20*(3), 199–218.

Cheng, L. S., Amaro, R. E., Xu, D., Li, W. W., Arzberger, P. W., & McCammon, J. A. (2008). Ensemble-based virtual screening reveals potential novel antiviral compounds for avian influenza neuraminidase. *Journal of medicinal chemistry*, *51*(13), 3878–94. doi:10.1021/jm8001197

Collignon, B., Schulz, R., Smith, J. C., & Baudry, J. (2011). Task-parallel message passing interface implementation of Autodock4 for docking of very large databases of compounds using high- performance super computers. *Journal of computational Chemistry*, *32*(6), 1202–1209. doi:10.1002/jcc

Connolly, H., & Crary, J. (1997). Valvular Heart Disease Associated with Fenfluramine–Phentermine. *New England Journal of Medicine*, *337*(9), 581–588.

Croft, S. L. (2005). Public-private partnership: from there to here. *Transactions of the Royal Society of Tropical Medicine and Hygiene*, *99 Suppl 1*, S9–14. doi:10.1016/j.trstmh.2005.06.008

Desmond-Hellmann, S. (2013). The Cost Of Creating A New Drug Now $5 Billion, Pushing Big Pharma To Change. *forbes.com*. Retrieved from http://www.forbes.com/sites/matthewherper/2013/08/11/how-the-staggering-cost-of-inventing-new-drugs-is-shaping-the-future-of-medicine/

Di Bernardo, D., Thompson, M. J., Gardner, T. S., Chobot, S. E., Eastwood, E. L., Wojtovich, A. P., … Collins, J. J. (2005). Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nature biotechnology*, *23*(3), 377–83. doi:10.1038/nbt1075

Durrant, J. D., Amaro, R. E., Xie, L., Urbaniak, M. D., Ferguson, M. a J., Haapalainen, A., … McCammon, J. A. (2010). A multidimensional strategy to detect polypharmacological targets in the absence of structural and sequence homology. *PLoS computational biology*, *6*(1), e1000648. doi:10.1371/journal.pcbi.1000648

Durrant, J. D., & McCammon, J. A. (2010). Computer-aided drug-discovery techniques that account for receptor flexibility. *Current opinion in pharmacology*, *10*(6), 770–4. doi:10.1016/j.coph.2010.09.001

Durrant, J. D., & McCammon, J. A. (2011). Molecular dynamics simulations and drug discovery. *BMC biology*, *9*(71).

Ellingson, S.R., & Baudry, J. (2011). High-throughput virtual molecular docking: Hadoop implementation of AutoDock4 on a private cloud. In *Proceedings of the second international workshop on Emerging computational methods for the life sciences*. ACM. Retrieved from http://doi.acm.org/10.1145/1996023.1996028

Ellingson, Sally R., Dakshanamurthy, S., Brown, M., Smith, J. C., & Baudry, J. (2012). Accelerating Virtual High-Throughput Ligand Docking: Screening One Million Compounds Using a Petascale Supercomputer. *Proceedings of the third international workshop on Emerging computational methods for the life sciences (ECMLS)*.

Ellingson, Sally R., Dakshanamurthy, S., Brown, M., Smith, J. C., & Baudry, J. (2013). Accelerating virtual high-throughput ligand docking: current technology and case study on a petascale supercomputer. *Concurrency Computat.: Pract. Exper. (2013)*. doi:10.1002/cpe.3070

Ellingson, Sally R., Smith, J. C., & Baudry, J. (2013). VinaMPI: Facilitating Multiple Receptor High-Throughput Virtual Docking on High-Performance Computers. *J. Comp. Chem.*, *34*(25), 2212–21.

Emig, D., Ivliev, A., Pustovalova, O., Lancashire, L., Bureeva, S., Nikolsky, Y., & Bessarabova, M. (2013). Drug target prediction and repositioning using an integrated network-based approach. *PloS one*, *8*(4), e60618. doi:10.1371/journal.pone.0060618

Estrada, T., Armen, R., & Taufer, M. (2010). Automatic selection of near-native protein-ligand conformations using a hierarchical clustering and volunteer computing. *BCB '10 Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, 204–213.

Faulon, J.-L., Misra, M., Martin, S., Sale, K., & Sapra, R. (2008). Genome scale enzyme-metabolite and drug-target interaction predictions using the signature molecular descriptor. *Bioinformatics (Oxford, England)*, *24*(2), 225–33. doi:10.1093/bioinformatics/btm580

Fernald, G. H., Capriotti, E., Daneshjou, R., Karczewski, K. J., & Altman, R. B. (2011). Bioinformatics challenges for personalized medicine. *Bioinformatics (Oxford, England)*, *27*(13), 1741–8. doi:10.1093/bioinformatics/btr295

Fischer, E. (1894). Einfluss der Configuration auf die Wirkung der Enzyme. *Ber. Dtsch. Chem. Ges.*, *27*, 2985–2993.

Giaever, G., Chu, A. M., Ni, L., Connelly, C., Riles, L., Véronneau, S., … André, B. (2002). Functional profiling of the Saccharomyces cerevisiae genome. *Nature*, *418*(6896), 387–91. doi:10.1038/nature00935

Gilson, M. K., & Zhou, H.-X. (2007). Calculation of protein-ligand binding affinities. *Annual review of biophysics and biomolecular structure*, *36*, 21–42. doi:10.1146/annurev.biophys.36.040306.132550

Guerrero, G., Perez-Sanchez, H. E., Cecilia, J. M., & Garcia, J. M. (2012). Parallelization of Virtual Screening in Drug Discovery on Massively Parallel Architectures. *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*.

Harvey, M. J., & Fabritiis, G. De. (2012). A survey of computational molecular science using graphics processing units. *WIREs Computational Molecular Science*. doi:10.1002/wcms.1101

Hopkins, A. L., & Groom, C. R. (2002). The druggable genome. *Nature reviews. Drug discovery*, *1*(9), 727–30. doi:10.1038/nrd892

Huang, N., Shoichet, B. K., & Irwin, J. J. (2006). Benchmarking sets for molecular docking. *Journal of medicinal chemistry*, *49*(23), 6789–801. doi:10.1021/jm0608356

Hui-fang, L., Qing, S., Jian, Z., & Wei, F. (2010). Evaluation of various inverse docking schemes in multiple targets identification. *Journal of molecular graphics & modelling*, *29*(3), 326–30.

Humphrey, W., Dalke, a, & Schulten, K. (1996). VMD: visual molecular dynamics. *Journal of molecular graphics*, *14*(1), 33–8, 27–8. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/8744570

Hutcheson, J. D., Setola, V., Roth, B. L., & Merryman, W. D. (2011). Serotonin receptors and heart valve disease--it was meant 2B. *Pharmacology & therapeutics*, *132*(2), 146–57. doi:10.1016/j.pharmthera.2011.03.008

Introducing Titan: Advancing the Era of Accelerated Computing. (n.d.). Retrieved from http://www.olcf.ornl.gov/titan/

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., & Coleman, R. G. (2012). ZINC: A Free Tool to Discover Chemistry for Biology. *Journal of chemical information and modeling*. doi:10.1021/ci3001277

Jacob, R., Anderson, T., & McDougal, O. M. (2012). Accessible High-Throughput Virtual Screening Molecular Docking Software for Students and Educators. *PLoS Computational Biology*, *8*(5), e1002499. Retrieved from http://dx.plos.org/10.1371/journal.pcbi.1002499

Jacq, N., Breton, V., Chen, H., & Ho, L. (2007). Virtual screening on large scale grids. *Parallel Computing*. Retrieved from http://www.sciencedirect.com/science/article/pii/S016781910700021X

Jenwitheesuk, E., Horst, J. a, Rivas, K. L., Van Voorhis, W. C., & Samudrala, R. (2008). Novel paradigms for drug discovery: computational multitarget screening. *Trends in pharmacological sciences*, *29*(2), 62–71. doi:10.1016/j.tips.2007.11.007

Katz, D. S., Armstrong, T. G., Zhang, Z., Wilde, M., & Wozniak, J. M. (2012). Many-Task Computing and Blue Waters. *Technical Report CI-TR-13-0911. Computation Institute, University of Chicago & Argonne National Laboratory*, 1–47. Retrieved from http://arxiv.org/abs/1202.3943

Keiser, M. J., Roth, B. L., Armbruster, B. N., Ernsberger, P., Irwin, J. J., & Shoichet, B. K. (2007). Relating protein pharmacology by ligand chemistry. *Nature biotechnology*, *25*(2), 197–206. doi:10.1038/nbt1284

Kinnings, S. L., Liu, N., Buchmeier, N., Tonge, P. J., Xie, L., & Bourne, P. E. (2009). Drug discovery using chemical systems biology: repositioning the safe medicine Comtan to treat multi-drug and extensively drug resistant tuberculosis. *PLoS computational biology*, *5*(7), e1000423. doi:10.1371/journal.pcbi.1000423

Kitano, H. (2007). Towards a theory of biological robustness. *Molecular systems biology*, *3*, 137. doi:10.1038/msb4100179

Knox, C., Law, V., Jewison, T., Liu, P., Ly, S., Frolkis, A., … Wishart, D. S. (2011). DrugBank 3.0: a comprehensive resource for “omics” research on drugs. *Nucleic acids research*, *39*(Database issue), D1035–41. doi:10.1093/nar/gkq1126

Koshland, D. E. (1958). Application of a Theory of Enzyme Specificity to Protein Synthesis. *Proceedings of the National Academy of Sciences of the United States of America*, *44*(2), 98–104. Retrieved from http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=335371&tool=pmcentrez&rendertype=abstract

Krejsa, C., Horvath, D., Rogalski, S., Penzotti, J., Mao, B., Barbosa, F., & Migeon, J. (2003). Predicting ADME properties and side effects : The BioPrint approach. *Current Opinion in Drug Discovery & Development*, *6*(4), 470–480.

Kukol, A. (2011). Consensus virtual screening approaches to predict protein ligands. *European journal of medicinal chemistry*, *46*(9), 4661–4. doi:10.1016/j.ejmech.2011.05.026

Kummerfeld, S. K., & Teichmann, S. a. (2009). Protein domain organisation: adding order. *BMC bioinformatics*, *10*, 39. doi:10.1186/1471-2105-10-39

Lee, H., Salzemann, J., Jacq, N., Ho, L.-Y., Chen, H.-Y., Breton, V., … Wu, Y.-T. (2006). Grid-enabled high-throughput in silico screening against influenza A neuraminidase. *IEEE Trans Nanobioscience*, *5*(4), 288–95.

Lin, J.-H., Perryman, A. L., Schames, J. R., & McCammon, J. A. (2002). Computational drug design accommodating receptor flexibility: the relaxed complex scheme. *Journal of the American Chemical Society*, *124*(20), 5632–3.

Luque, I., & Freire, E. (2000). Structural stability of binding sites: consequences for binding affinity and allosteric effects. *Proteins*, *Suppl 4*(July), 63–71. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/11013401

Michel, J., & Essex, J. W. (2010). Prediction of protein-ligand binding affinity by free energy simulations: assumptions, pitfalls and expectations. *Journal of computer-aided molecular design*, *24*(8), 639–58. doi:10.1007/s10822-010-9363-3

Moitessier, N., Englebienne, P., Lee, D., Lawandi, J., & Corbeil, C. R. (2008). Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go. *British Journal of Pharmacology*, *153*, S7–S26. doi:10.1038/sj.bjp.0707515

Molecular Operating Environment (MOE). (2012). Chemical Computing Group Inc. 1010 Sherbooke St. West, Suite #910, Montreal, QC, Canada, H3A 2R7.

Morphy, R., & Rankovic, Z. (2005). Designed multiple ligands. An emerging drug discovery paradigm. *Journal of medicinal chemistry*, *48*(21), 6523–6543.

Morris, G. M., Goodsell, D. S., Halliday, R. S., Huey, R., Hart, W. E., Belew, R. K., … Al, M. E. T. (1998). Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. *Journal of Computational Chemistry*, *19*(14), 1639–1662.

Moya-García, A., & Ranea, J. (2013). Insights into polypharmacology from drug-domain associations. *Bioinformatics (Oxford, England)*, *29*(16), 1934–7. doi:10.1093/bioinformatics/btt321

Najmanovich, R., Kuttner, J., Sobolev, V., & Edelman, M. (2000). Side-chain flexibility in proteins upon ligand binding. *Proteins*, *39*(3), 261–8. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/10737948

Norgan, A., Coffman, P., Kocher, J., Katzmann, D., & Sosa, C. (2011). Multilevel Parallelization of AutoDock 4.2. *Journal of Cheminformatics*, *3*(12).

Olson, H., Betton, G., Robinson, D., Thomas, K., Monro, A., Kolaja, G., … Heller, A. (2000). Concordance of the toxicity of pharmaceuticals in humans and in

animals. *Regulatory toxicology and pharmacology : RTP*, *32*(1), 56–67. doi:10.1006/rtph.2000.1399

Paolini, G., Shapland, R., Hoorn, W. P. van, Mason, J. S., & Hopkins, A. L. (2006). Global mapping of pharmacological space. *Nature Biotechnology*, *24*(7), 805–815.

Paul, S. M., Mytelka, D. S., Dunwiddie, C. T., Persinger, C. C., Munos, B. H., Lindborg, S. R., & Schacht, A. L. (2010). How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nature reviews. Drug discovery*, *9*(3), 203–14. doi:10.1038/nrd3078

Peters, A., Lundberg, M., Lang, P., & Sosa, C. (2008). High throughput computing validation for drug discovery using the DOCK program on a massively parallel system. *An IBM Redpaper publication*. Retrieved from http://www.redbooks.ibm.com/abstracts/redp4410.html?Open

Peters, J.-U. (2013). Polypharmacology - foe or friend? *Journal of medicinal chemistry*, *56*(22), 8955–71. doi:10.1021/jm400856t

Phillips, J. C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., … Schulten, K. (2005). Scalable molecular dynamics with NAMD. *Journal of computational chemistry*, *26*(16), 1781–802. doi:10.1002/jcc.20289

Pohorille, A., Jarzynski, C., & Chipot, C. (2010). Good practices in free-energy calculations. *The journal of physical chemistry. B*, *114*(32), 10235–53. doi:10.1021/jp102971x

Raicu, I. (2008). Many-task computing for grids and supercomputers. *Many-Task Computing on Grids …*, 1–11. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4777912

Raicu, I., Zhang, Z., Wilde, M., & Foster, I. (2008). Toward loosely coupled programming on petascale systems. *SC '08 Proceedings of the 2008 ACM/IEEE conference on Supercomputing*.

Reymond, J.-L., & Awale, M. (2012). Exploring chemical space for drug discovery using the chemical universe database. *ACS chemical neuroscience*, *3*(9), 649–57. doi:10.1021/cn3000422

Riedel, M., Memon, A., Memon, M. S., Mallmann, D., Streit, A., Wolf, F., & Lippert, T. (2008). Improving e-Science with Interoperability of the e-Infrastructures EGEE and DEISA. *Proceedings of the ….*

Riedel, M., & Memon, M. (2011). e-Science Infrastructure Integration Invariants to Enable HTC and HPC Interoperability Applications. *2011 IEEE International Parallel & Distributed Processing Symposium*, 922–931.

Rognan, D. (2007). Chemogenomic approaches to rational drug design. *British journal of pharmacology*, *152*(1), 38–52. doi:10.1038/sj.bjp.0707307

Rothman, R. B., Baumann, M. H., Savage, J. E., Rauser, L., McBride, A., Hufeisen, S. J., & Roth, B. L. (2000). Evidence for Possible Involvement of 5-HT2B Receptors in the Cardiac Valvulopathy Associated With Fenfluramine and Other Serotonergic Medications. *Circulation*, *102*(23), 2836–2841. doi:10.1161/01.CIR.102.23.2836

Sánchez-Linares, I., Perez-Sanchez, H., Guerrero, G. D., Cecilia, J. M., & Garcia, J. M. (2011). Accelerating multiple target drug screening on GPUs. *Proceedings of CMSB*, 95–102. Retrieved from http://dl.acm.org/citation.cfm?id=2037523

Scannell, J. W., Blanckley, A., Boldon, H., & Warrington, B. (2012). Diagnosing the decline in pharmaceutical R&D efficiency. *Nature reviews. Drug discovery*, *11*(3), 191–200. doi:10.1038/nrd3681

Schlessinger, A., & Rost, B. (2005). Protein flexibility and rigidity predicted from sequence. *Proteins*, *61*(1), 115–26. doi:10.1002/prot.20587

Shan, Y., Kim, E. T., Eastwood, M. P., Dror, R. O., Seeliger, M. a, & Shaw, D. E. (2011). How does a drug molecule find its target binding site? *Journal of the American Chemical Society*, *133*(24), 9181–3. doi:10.1021/ja202726y

Shirts, M. R., Mobley, D. L., & Chodera, J. D. (2007). Alchemical Free Energy Calculations: Ready for Prime Time? *Annual Reports in Computational Chemistry*, *3*(07), 41–59. doi:10.1016/S1574-1400(07)03004-6

Shoichet, B. K. (2004). Virtual screening of chemical libraries. *Nature*, *432*(7019), 862–5. doi:10.1038/nature03197

Simon, Z., Peragovics, A., Vigh-Smeller, M., Csukly, G., Tombor, L., Yang, Z., … Málnási-Csizmadia, A. (2012). Drug effect prediction by polypharmacology-based interaction profiling. *Journal of chemical information and modeling*, *52*(1), 134–45. doi:10.1021/ci2002022

Soga, S., Shirai, H., Kobori, M., & Hirayama, N. (2007). Use of amino acid composition to predict ligand-binding sites. *Journal of chemical information and modeling*, *47*(2), 400–6. doi:10.1021/ci6002202

Sridhar, J. K., & Panda, D. K. (2009). Impact of Node Level Caching in MPI Job. In M. Ropo, J. Westerholm, & J. Dongarra (Eds.), *Recent Advances in Parallel Virtual Machine and Message Passing Interface* (pp. 230–239). Springer Berlin Heidelberg. doi:10.1007/978-3-642-03770-2_29

Stearn, B., Bhatia, K., Baldridge, K. K., Li, W. W., Arzberger, P., Chem, O., & Zurich, C.-. (2006). Opal : Simple Web Services Wrappers for Scientific Applications. In *ICWS 2006, IEEE International Conference on Web Services*.

The HDF Group. Hierarchical data format version 5, 2000-2010. (n.d.). Retrieved from http://www.hdfgroup.org/HDF5

The Official UCSF DOCK Web-site: DOCK6. (n.d.). Retrieved from http://dock.compbio.ucsf.edu/DOCK_6/index.htm

Totrov, M., & Abagyan, R. (2009). Flexible ligand docking to multiple receptor conformations: a practical alternative. *Curr Opin Struct Biol.*, *18*(2), 178–184.

Trott, O., & Olson, A. . (2010). AutoDock Vina: Improving The Speed And Accuracy Of Docking With a New Scoring Function, Efficient Optimization, and Multithreading. *Journal of Computational Chemistry*, *31*, 455–461.

Trott, Oleg, & Olson, A. (2011). AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. *J Comput Chem*, *31*(2), 455–461. doi:10.1002/jcc.21334.AutoDock

Warren, G. L., Andrews, C. W., Capelli, A.-M., Clarke, B., LaLonde, J., Lambert, M. H., … Head, M. S. (2006). A critical assessment of docking programs and scoring functions. *Journal of medicinal chemistry*, *49*(20), 5912–31. doi:10.1021/jm050362n

Wassman, C. D., Baronio, R., Demir, Ö., Wallentine, B. D., Chen, C.-K., Hall, L. V, … Amaro, R. E. (2013). Computational identification of a transiently open L1/S3 pocket for reactivation of mutant p53. *Nature communications*, *4*, 1407. doi:10.1038/ncomms2361

Wei, D., Jiang, X., Zhou, L., Chen, J., Chen, Z., Chem, J. M., & Asap, A. (2008). Discovery of Multitarget Inhibitors by Combining Molecular Docking with Common Pharmacophore Matching Discovery of Multitarget Inhibitors by Combining Molecular Docking with Common. *J. Med. Chem.*, *51*(24), 7882–8.

Werner, T., Morris, M. B., Dastmalchi, S., & Church, W. B. (2012). Structural modelling and dynamics of proteins for insights into drug interactions. *Advanced drug delivery reviews*, *64*(4), 323–43. doi:10.1016/j.addr.2011.11.011

Winzeler, E. a. (1999). Functional Characterization of the S. cerevisiae Genome by Gene Deletion and Parallel Analysis. *Science*, *285*(5429), 901–906. doi:10.1126/science.285.5429.901

Yuan, Z., Zhao, J., & Wang, Z.-X. (2003). Flexibility analysis of enzyme active sites by crystallographic temperature factors. *Protein Engineering Design and Selection*, *16*(2), 109–114. doi:10.1093/proeng/gzg014

Zambrowicz, B. P., & Sands, A. T. (2004). Modeling drug action in the mouse with knockouts and RNA interference. *Drug Discovery Today: TARGETS*, *3*(5), 198–207. doi:10.1016/S1741-8372(04)02454-5

Zavodszky, M. I., & Kuhn, L. A. (2005). Side-chain flexibility in protein – ligand binding : The minimal rotation hypothesis. *Protein Science*, *14*, 1104–1114. doi:10.1110/ps.041153605.and

Zhang, X., Wong, S. E., & Lightstone, F. C. (2013). Message passing interface and multithreading hybrid for parallel molecular docking of large databases on petascale high performance computing machines. *Journal of computational chemistry*, 1–13. doi:10.1002/jcc.23214

Zhao, S, Goodsell, D. S., & Olson, a J. (2001). Analysis of a data set of paired uncomplexed protein structures: new metrics for side-chain flexibility and model evaluation. *Proteins*, *43*(3), 271–9. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/11288177

Zhao, Shiwen, & Li, S. (2010). Network-based relating pharmacological and genomic spaces for drug target identification. *PloS one*, *5*(7), e11764. doi:10.1371/journal.pone.0011764

# VITA

Sally R. Ellingson is a Florida native. She completed B.S. degrees in Computer Science and Mathematical Sciences at the Florida Institute of Technology. During her studies, she decided that she wanted to use her technical abilities to aid scientific research. Because of a great opportunity from SCALE-IT (an NSF funded computational biology fellowship), she joined the joint University of Tennessee and Oak Ridge National Laboratory Genome Science and Technology program where she also minored in Computational Sciences. Through this program she met her advisor, Dr. Jerome Baudry, whom she worked with at the Center for Molecular Biophysics.