

Heuristics for Scaling up Distributed Protein Docking

Prachi Pradeep
Marquette University

Recommended Citation

Pradeep, Prachi, "Heuristics for Scaling up Distributed Protein Docking" (2011). *Master's Theses (2009 -)*. Paper 107.
http://epublications.marquette.edu/theses_open/107

HEURISTICS FOR SCALING UP
DISTRIBUTED PROTEIN DOCKING

by

PRACHI PRADEEP

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

August 2011

ABSTRACT

HEURISTICS FOR SCALING UP
DISTRIBUTED PROTEIN DOCKING

Prachi Pradeep
Marquette University, 2011

Docking is a computational technique which predicts the interaction between a protein and a potential drug compound. Virtual screening is a tool, which employs docking, to investigate huge libraries of compounds and predicts potential drug molecules that bind favorably to the protein of interest. The size of one such commercially available library is about 13 million compounds. It would take approximately 400 years of CPU time to examine this library! As an alternative a high performance computing application with a distributed docking strategy is needed, which can efficiently predict the favorable compounds and can eventually be scaled for huge libraries.

In this thesis, IncreDock, a scoring based incremental docking software has been developed to improve the efficiency of virtual screening process. IncreDock provides two approaches to the distributed docking problem. First, it allows for a completely parallel implementation where the entire library is explored simultaneously in an unordered fashion. Second, and more important, is the ordered incremental parallel implementation where the library is explored in increments and a scoring function is used to determine the order of dockings.

IncreDock was used to perform docking studies on four different proteins using a library of 10,573 compounds. The results suggest that IncreDock is able to predict better compounds in early increments of dockings. IncreDock, thus, provides an effective initial strategy to sample out good ligands in less compute time and forms a good precursor to a tool that can proficiently investigate huge chemical libraries.

ACKNOWLEDGEMENTS

Prachi Pradeep

I would like to express my heartfelt gratitude to my committee members Dr. Stephen Merrill , Dr. Daniel Sem and Dr. Craig Struble for their continuous guidance, support and encouragement, throughout this thesis work.

I would like to thank Terrence Neumann, graduate student in Chemistry, for helping me understand the basics of protein docking process.

I want to thank the National Science Foundation awards OCI-0923037 “MRI: Acquisition of a Parallel Computing Cluster and Storage for the Marquette University Grid (MUGrid)” and CBET-0521602 “Acquisition of a Linux Cluster to Support College-Wide Research & Teaching Activities.” which funded the Père cluster on which the experiments were performed.

Last but not the least, I would like to take this opportunity to thank my family and friends for being wonderfully patient and supportive all through.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
LIST OF FIGURES	v
CHAPTER	
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Statement of Problem	2
1.3 Summary of Results	3
1.4 Organization of Thesis	3
2 BACKGROUND	4
2.1 Drug Discovery - The basic science	4
2.1.1 Virtual Screening	4
2.1.2 Docking	6
2.1.3 Distributed Protein Docking	8
2.1.4 Lipinski's Rule-of-Five	8
2.1.5 Chemical Libraries	9
2.2 Tools and Software	9
2.2.1 AutoDock	10
2.2.2 Condor	10
2.2.3 Condor DAGMan	11
3 APPROACH	15
3.1 Basic Approach	15
3.1.1 Data Partitioning	16
3.1.2 Job Submission and Control	17

3.1.3	Aggregation of Results	19
3.2	Incremental Approach	19
3.2.1	X-DAG	20
3.2.2	Data Partitioning	22
3.2.3	Job Submission and Control	25
3.2.4	Aggregation of Results	26
4	EVALUATION AND RESULTS	28
4.1	Chemical Library	28
4.2	Proteins	28
4.3	Performance Metrics	30
4.4	Experimental Setup	32
4.5	Results	33
4.5.1	<i>Dihydrofolate Reductase</i> (DHFR)	33
4.5.2	<i>Dihydropicolinate reductase</i> (DHPR)	38
4.5.3	<i>Thioredoxin Reductase</i> (TrxR)	43
4.5.4	<i>Human Dual Specificity Phosphatase 5</i> (DUSP5C)	48
4.5.5	<i>Human Dual Specificity Phosphatase 5</i> (DUSP5R)	53
4.6	Discussion	58
4.6.1	Average Energy of a round	59
4.6.2	Ligand Enrichment and Cumulative Ligand Enrichment	60
4.6.3	Significance Testing	60
5	CONCLUSIONS AND FUTURE WORK	62
5.1	Conclusion	62
5.2	Future Work	63
	BIBLIOGRAPHY	65
	APPENDIX	
A	STEPS FOR PREPARATION OF INPUT FILES FOR DOCKINGS	69

A.1	Steps for preparing the Ligand Molecule	69
A.2	Steps for preparing the Protein Molecule	69
A.3	Combining the input data to create the input tar file	70
B	BAYESIAN SCORING FUNCTION	71

LIST OF FIGURES

2.1	Illustration of the Virtual Screening workflow. The protein (blue) and potential compound (red, green) structure information is input to the software which calculates the docking scores. The compounds with good scores are then tested in the laboratory for actual activity	5
2.2	An illustration of the docking process. The ligand (blue) interacts with the protein receptor (pink) to produce the protein-ligand complex.	6
2.3	An illustration of the distributed docking process. The individual docking results from each processor are compiled in the end to determine good dockings.	7
2.4	An illustration of the docking process. The ligand (green) interacts with the protein receptor (red) to produce the protein-ligand complex.	8
2.5	Condor resource management system.	10
2.6	An illustrative of a Diamond DAG. After job A is completed , jobs B and C are launched. After jobs B and C are completed, job D is launched.	12
2.7	An illustrative of the Condor DAGMan input file. Each job is defined by the keyword <i>JOB</i> . The relationship between jobs is defined by keywords <i>PARENT</i> and <i>CHILD</i>	12
2.8	An illustrative of the elements of a DAG Node - The PRE and POST scripts, which are optional. Failure of any one of these elements leads to the failure of the node. .	13
2.9	Node A is considered complete only when the PRE, JOB and POST elements are completed successfully.	13
2.10	An illustrative of the Condor DAGMan input file with PRE and POST script. Keywords <i>SCRIPT PRE</i> and <i>SCRIPT POST</i> are used to define the pre and post script, respectively, for job A.	14

3.1	An illustration of the architecture of IncreDock. An input file containing the files necessary for a bulk docking experiment is provided. IncreDock partitions the data into single docking events and assigns one docking experiment to each available Condor worker. Upon completion, Condor aggregates the results from the individual workers to present the user with a final single result file.	16
3.2	The input Tar File is utilized to create the output experiment directory upon launching the dockings.	17
3.3	A sample of DAG input file.	18
3.4	An illustrative of parallel submission of the dockings.	18
3.5	A sample of the aggregated output result file from the last summarize step.	19
3.6	An Illustrative of the X-DAG. Jobs B, C, D are launched when job A is completed. Job E is launched when jobs B, C, D are completed. Jobs F, G, H are launched when job E is completed. Job I is launched when jobs F, G, H are completed.	21
3.7	An illustrative of the Condor DAGMan input file for a X-DAG. Each job is defined by the keyword <i>JOB</i> . The relationship between jobs is defined by recursive usage of keywords <i>PARENT</i> and <i>CHILD</i>	21
3.8	An illustrative of the parallel submission of dockings for the Incremental approach.	22
3.9	A sample of DAG input file for the Incremental approach.	25
3.10	A sample output result file for Incremental approach using Bayesian scoring.	26
4.1	Plot between average energy of a round and round number for DHFR. round 2 and 3 shows a significant drop in average energy.	34
4.2	Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.	34
4.3	The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red) till 35% ligands are docked and reasonable higher for complete docking.	35
4.4	Docking Complex of DHFR with the ligand 103 in round 2.	37

4.5	Ligand Enrichment for top 10 ligands.	37
4.6	Cumulative Ligand Enrichment for top 10 ligands.	38
4.7	Plot between average energy of a round and round number for DHPR. Round 2 shows a significant drop in average energy.	38
4.8	Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.	39
4.9	The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red) for the entire docking process.	40
4.10	Complex of DHPR with the ligand 210 in round 2.	42
4.11	Ligand Enrichment for top 10 ligands.	43
4.12	Cumulative Ligand Enrichment for top 10 ligands.	43
4.13	Plot between average energy of a round and round number for TrxR. round 2 shows a decreased average energy but there is no great change in the average energy of other rounds.	44
4.14	Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.	45
4.15	The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is almost the same as in case of random selection (Red) for the entire docking process.	45
4.16	Docking Complex of TrxR with the ligand 410 in round 2.	47
4.17	Ligand Enrichment for top 10 ligands.	48
4.18	Cumulative Ligand Enrichment for top 10 ligands.	48
4.19	Plot between average energy of a round and round number for DUSP5C. Round 2 shows a significant drop in average energy and almost constant average energy for subsequent rounds.	49

4.20	Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.	50
4.21	The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red) till 75% of the ligands are docked.	50
4.22	Docking Complex of DUSP5C with the ligand 270 in round 2.	52
4.23	Ligand Enrichment with top 10 ligands.	53
4.24	Cumulative Ligand Enrichment with top 10 ligands.	53
4.25	Plot between average energy of a round and round number for DUSP5R. Round 2 shows a significant drop in average energy and almost constant change in average energy for subsequent rounds.	54
4.26	Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.	55
4.27	The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red).	55
4.28	Docking Complex of DUSP5R with the ligand 51 in round 2.	57
4.29	Ligand Enrichment with top 10 ligands.	58
4.30	Cumulative Ligand Enrichment with top 10 ligands.	58

CHAPTER 1

INTRODUCTION

Drug discovery has evolved from the days when natural products were intuitively used as a source of medicine. With advancements in science and commercialization, these natural products were later developed into marketable drugs. The increase in knowledge of the structure and function of proteins and the understanding of signal transduction pathways has helped in the evolution of drug discovery process.

Modern drug discovery is a “lengthy, expensive, difficult, and inefficient process” with low rate of new therapeutic discovery [1]. Currently, the research and development cost of each new molecular entity is approximately US \$1.8 billion [2]. Considering the amount of time and money (average \$800 million) to introduce a new drug in the market, methods to enhance the drug discovery process are strongly needed.

Conventional experimental procedures such as medicinal chemistry and high throughput screening (HTS) are still the most accurate methods for rapid identification of drug leads. HTS involves performing individual biochemical assays using large libraries of chemical compounds, to test their ability to bind to a target protein. Compounds with good binding ability are then shortlisted as a potential drug lead. However, there is still huge time and cost associated with HTS which can be reduced if we can predict some expected behavior beforehand. In this context, computational methods are now being used to enhance the drug development process [3, 4]. The development of computational algorithms to accelerate the process of screening compounds before they can go for experimental analysis, is what led to the development of virtual screening (VS).

1.1 Motivation

VS is a tool which can explore a given chemical library of compounds and predict the binding energy of each compound to the protein of interest, utilizing a process called docking. The availability of computing clusters allows for a highly parallel activity, where a feasible number of these processes can be run simultaneously. However, there is a constraint on the the number of processors available and most of the compounds analyzed will still not be good drug candidates.

There exist standard software which predict if a compound is a good fit as a drug candidate. It was observed that every single docking process takes around 30 minutes to 90 minutes of compute time on Intel(R) Xeon(R) CPU X5550 @ 2.67GHz processor. The Chemical Proteomics Facility at Marquette houses around 10,000 of such potential compounds [5]. In order to predict the good compounds from this set it would take about $10,000 \times 40$ minutes, which is close to 111 days on a single such processor!

The actual number of such compounds available in ZINC is not commercial, a free database of commercially-available compounds for virtual screening, is close to 13 million [6]. On a single processor, docking all compounds is not the most preferable option. Utilizing a parallel system with 500 dedicated cores can reduce this time-scale to 5.5 hours for 10,000 compounds but still 9.6 months for 13 million compounds. So, the huge size of the chemical library available for drug discovery studies has ever increased the need for a process to reduce the time and cost associated with the screening process.

These facts trigger a thought of how can this process be improved, so that the time of predicting suitable drug leads can be further reduced. Since VS process utilizes a computer algorithm, it takes into account some information about the nature of the compound or protein. This idea can be used to develop a strategy to prioritize and generate a subset of compounds from a given chemical library. This will help in reducing the computational time for VS and allow for early laboratory testing and validation. VS methodology can, thus, be combined with a selection algorithm to serve as a fast and cost effective way of drug discovery.

1.2 Statement of Problem

As stated in Section 1.1, the number of compounds available for docking studies is increasing as the chemical structure of more and more compounds is being elucidated. The Chemical Proteomics Facility at Marquette aims at docking the complete ZINC Database for expanding the scope of docking studies. The time and computational complexity involved in this process make it a difficult task. In order to get around this problem, an algorithm to selectively dock compounds is needed.

In this thesis, I propose heuristics for the scaling up the entire virtual screening process to deal with large chemical libraries. The software developed should implement a mechanism for incremental docking, by prioritizing and preferentially selecting ligands to dock.

1.3 Summary of Results

The results of the thesis work are summarized as follows:

- A software tool, IncreDock, is developed to implement automated parallel protein docking in increments, using a given chemical library.
- A Bayesian scoring function is developed for implementing incremental docking. The scoring function learns from docking information of previous runs, to predict better ligands for next run.
- IncreDock is tested for the performance of incremental docking using four different proteins and a chemical library of 10,573 compounds. It is observed that IncreDock gives significantly good results in initial runs of incremental docking. The average overall energy of the docking complexes and the frequency of appearance of potential drug leads is significantly improved in the first few runs, as compared to randomly docking the entire chemical library.

1.4 Organization of Thesis

In this thesis, I describe the problem addressed, the background and the implementation of IncreDock. Chapter 2 provides a background on the VS process, basics of protein docking and the high performance computing tool used in developing IncreDock. Chapter 3 describes the approach used in designing IncreDock. Chapter 4 describes the experiments, results and discussion to validate IncreDock. Finally, Chapter 5 presents a conclusion and summary of the work along with possible future work in this direction.

CHAPTER 2

BACKGROUND

As established in Chapter 1, the primary goal of this thesis is to provide a computational tool to perform virtual screening studies on large chemical libraries, such as ZINC. This chapter covers the basic background knowledge required to build such a tool. The background is discussed in two broad categories, the basic science behind drug discovery, and the tools and software used for the development of IncreDock.

2.1 Drug Discovery - The basic science

Drug discovery is defined as the process of discerning or designing new drugs [7, 8]. In the past, most drugs were identified by observing the therapeutic nature of natural products and their derivatives. Modern age drug discovery relies more on computational power for screening potential drug candidates and taking it forward to experimental trials. In order to implement a computation process, which can assist in drug discovery, it is important to understand the science behind the process.

Sections 2.1.1 and 2.1.2 describe the basics of the virtual screening process in the current paradigm, with a focus on the specific ideas relevant to this thesis. Section 2.1.4 introduces the Lipinski's rule-of-five, which describes the basic properties of a good drug candidate. Finally, Section 2.1.5 describes the idea of chemical libraries.

2.1.1 Virtual Screening

Virtual Screening (VS) is defined as the "Use of high-performance computing to analyze large databases of chemical compounds, in order to identify possible drug candidates" [9]. In view of this definition, drug discovery research relies heavily on the process of VS. As compared to HTC, which gives information about compounds that actually bind to a protein of interest, VS predicts the binding ability of a ligand as calculated by a computer algorithm.

There are two basic techniques of performing VS, ligand based and structure based. Ligand based VS involves molecular modeling of the compound to predict different binding poses.

On the other hand, structure based VS (sbVS) utilizes the known three dimensional structure of the compound, to predict its binding affinity to the protein of interest [10, 11]. This thesis work utilizes sbVS to evaluate the binding affinity of each compound to the protein under study.

Applicability of sbVS depends upon the knowledge and availability of three things, viz., a target protein, a library of potential compounds and computer software which predicts the binding affinity of the compound with the protein [8, 10]. The VS process is illustrated in Figure 2.1 [12]. Each of the available compounds is sampled in different structural conformations and scored on the basis of its complementarity to the receptor protein (blue) molecule. The compounds which return most favorable results are then tested in the laboratory for actual activity.

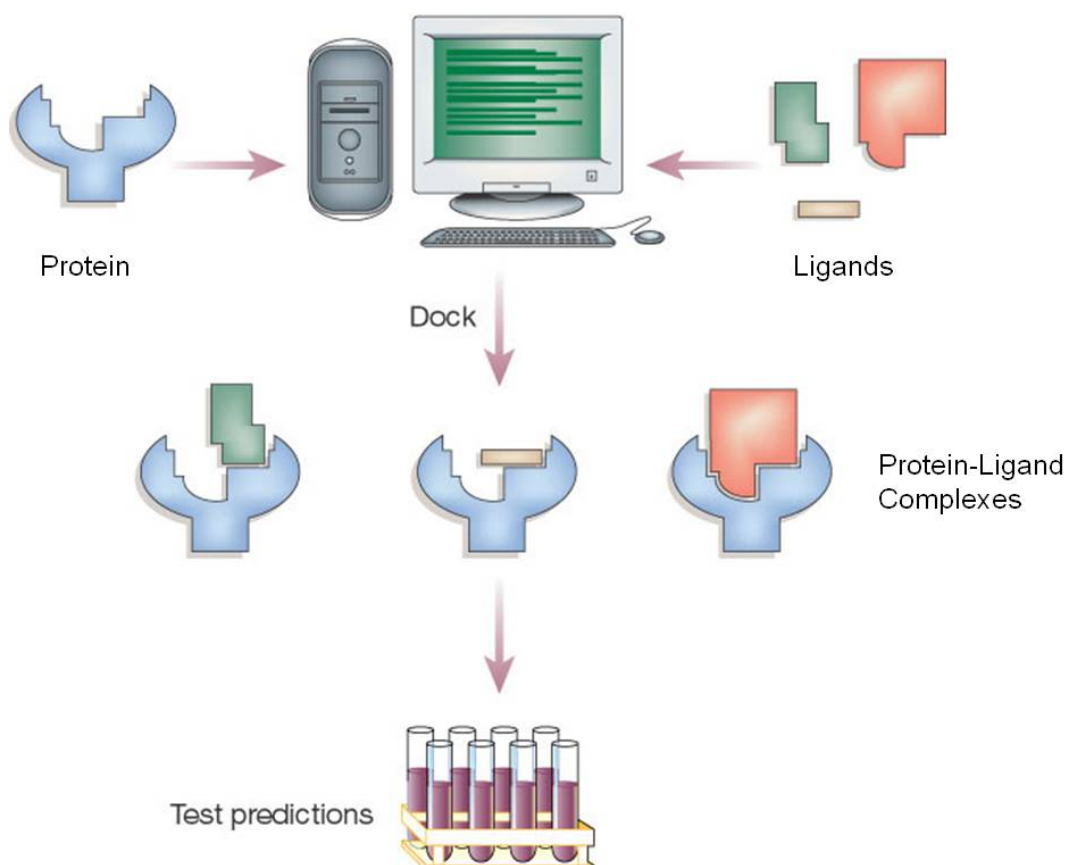


Figure 2.1: Illustration of the Virtual Screening workflow. The protein (blue) and potential compound (red, green) structure information is input to the software which calculates the docking scores. The compounds with good scores are then tested in the laboratory for actual activity

VS can, thus, be used in the very initial phase of drug discovery process to filter out the compounds of lesser potential from a huge available chemical library. This strategy helps in premature analysis of the binding affinity of a compound with the protein under study and helps in

reducing the number of compounds for eventual laboratory testing. The goal is to help the chemists utilize the structural and computational information in making decisions.

A positive advancement in this direction has been the availability of over 13 million purchasable compounds for virtual screening, which make the ZINC database. Protein targets are screened against these compounds to see which of them bind most strongly to the target. A *hit* occurs when there is a good interaction between the compound and protein. If there is a hit with a particular compound, it is considered a potential drug target and can be sent for further laboratory testing.

This thesis focuses on the initial step of VS, which involves docking various compounds into a protein. A strategy is formulated for deciding the order or sequence of docking trials. The next section describes the process of docking in detail.

2.1.2 Docking

A *ligand* is a molecule which is capable of binding to a biological molecule, protein here, and forms a complex which triggers some biological activity. A *pose* is the orientation of the ligand with the protein when they form a complex [13]. *Docking* is a computational process which predicts the preferred pose of the complex formed between the protein and the ligand (Figure 2.2). It models the intermolecular complex formed between two molecules by calculating the free energy of the complex in different poses. The ligand found in the process is called a *lead* compound, which is then modified and refined for more favorable interactions [14, 15, 16]. Docking, hence, forms an integral part of VS.

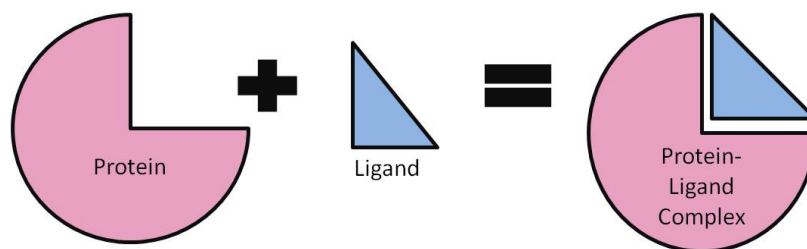


Figure 2.2: An illustration of the docking process. The ligand (blue) interacts with the protein receptor (pink) to produce the protein-ligand complex.

In reality, proteins and ligands have a complex 3 dimensional structure are in constant motion and move between different physical conformational states (Figure 2.3). This allows for

different poses in which a ligand can bind to a given protein and, hence, accounts for various protein-ligand complexes that can be formed for a given ligand. This makes docking a computationally expensive process.

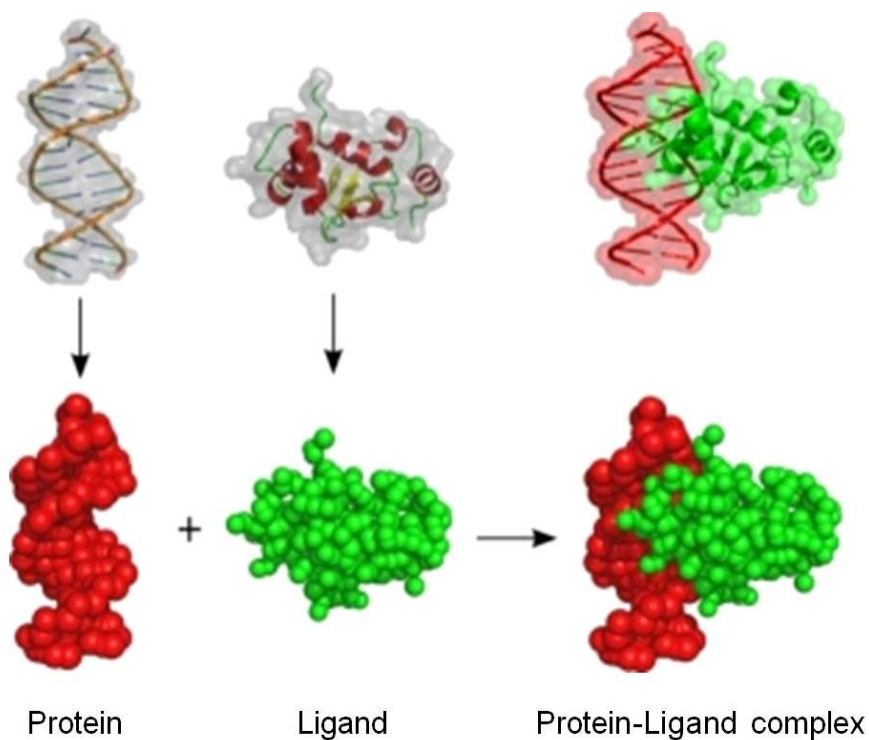


Figure 2.3: An illustration of the distributed docking process. The individual docking results from each processor are compiled in the end to determine good dockings.

The docking process is comprised of two major components; a docking algorithm and a scoring function. A *docking algorithm* is responsible for exploring the conformational degrees of freedom of a compound in generating different binding poses. Most docking algorithms assume a fixed protein and a flexible ligand molecule. The evaluation of how good a given pose is what forms the other component, *scoring function* of the docking process. A scoring function differentiates a good pose from all the other poses explored, by evaluating the free energy of binding of these poses. The pose with the lowest binding energy is the best hit for that protein-ligand complex [16].

Different docking protocols are contrasted on the basis of different schemes of docking algorithm and scoring functions used. DOCK was the first ever docking algorithm written by

Kuntz et al [17]. A lot many docking algorithms followed eventually, AutoDock, GOLD and FlexX, to name a few [18, 19, 20, 21].

2.1.3 Distributed Protein Docking

Each protein-ligand docking in the VS process is an independent step. Distributed parallel computing is an enhancement to the VS process where each docking can be performed in parallel, utilizing a cluster of processors. In Figure 2.4, the master processor distributes each protein-ligand docking job to a different processor. As the dockings are completed, individual results are gathered and aggregated into a single result file.

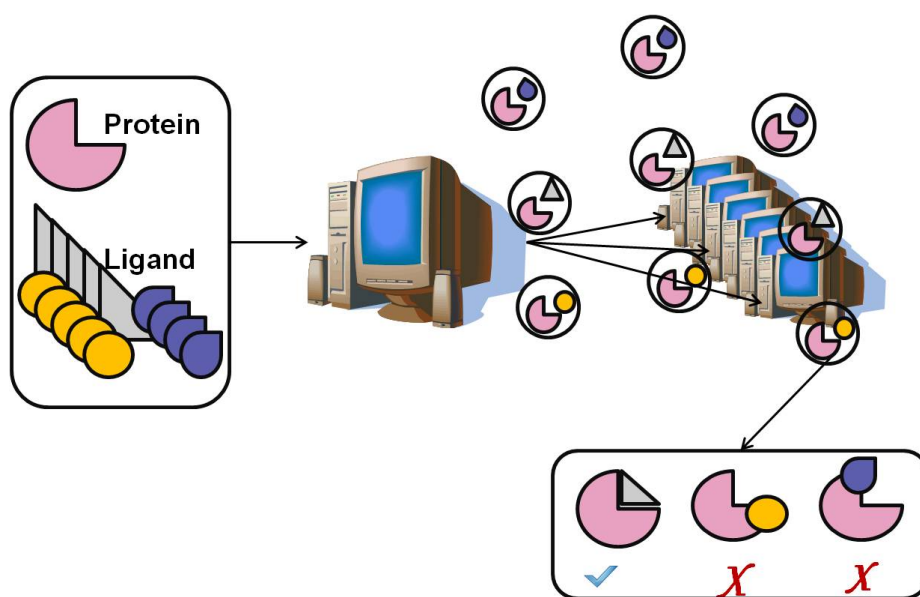


Figure 2.4: An illustration of the docking process. The ligand (green) interacts with the protein receptor (red) to produce the protein-ligand complex.

2.1.4 Lipinski's Rule-of-Five

Pharmaceutical industry heavily relies on Christopher Lipinski's rule-of-five analysis for predicting if a compound is suited to be a drug candidate or not. These rules were experimentally established and are a measure of compound permeation and solubility [22, 23]. The rule says that for a compound to have *drug likeliness*, it should meet the following criteria [24]:

- Its Molecular Weight should be under 500 Daltons.

- It should have limited Lipophilicity ($\log P < 5$, with $P = [drug]_{org}/[drug]_{aq}$).
- It should have a maximum of 5 Hydrogen-bond donors.
- It should have a maximum of 10 Hydrogen-bond acceptors

In this thesis, the properties emphasized in Lipinski's rule-of-five have been used in predicting the good binders, while implementing a selection criterion of preferably docking some ligands.

2.1.5 Chemical Libraries

A library of the chemical structure of physically available ligands, is another vital component of VS. The initial results from VS are useful, only when the results can be validated experimentally, by laboratory testing. So, chemical libraries consist of compounds which are commercially available for laboratory testing and verification.

ZINC is not commercial, a free database of commercially-available compounds for virtual screening, is one such chemical library [6]. It contains the structural information of over 13 *million* compounds, as needed for the virtual screening process.

2.2 Tools and Software

In this thesis, AutoDock, open source software commonly used for protein docking, is used. Section 2.2.1 gives an introduction to AutoDock. In the beginning of Chapter 1, it was demonstrated that docking ligands on a single computer is infeasible at the scale necessary for VS, which might involve tens of thousands or millions of ligands. Thus, to explore the 13 *million* compounds in ZINC on a single desktop machine is not realistic. Instead, VS typically employs distributed computing systems consisting of hundreds or thousands of computers. Distributed computing systems require several basic software tools to manage large scale computations. Condor distributed computing platform, which is an integral component of this thesis, is described in Sections 2.2.2 and 2.2.3.

2.2.1 AutoDock

AutoDock is freely available suite of automated docking tools. It predicts the how the ligands bind to the pre-calculated docking area (grid) on the target protein. These grids can be viewed and aid the physical realization of the docking site and binding [18, 19]. Lower predicted binding energy implies better ligand. AutoDock is one of the most widely used docking software and is, therefore, the docking software of choice for this thesis.

2.2.2 Condor

Condor is a resource management and scheduling system for executing computation-intensive jobs harnessing idle compute power [25]. Users submit their compute jobs to Condor - Condor takes care of the queuing mechanism, scheduling policy, priority scheme, and resource classifications. Once the jobs are completed, Condor informs the user of the results (Figure 2.5).

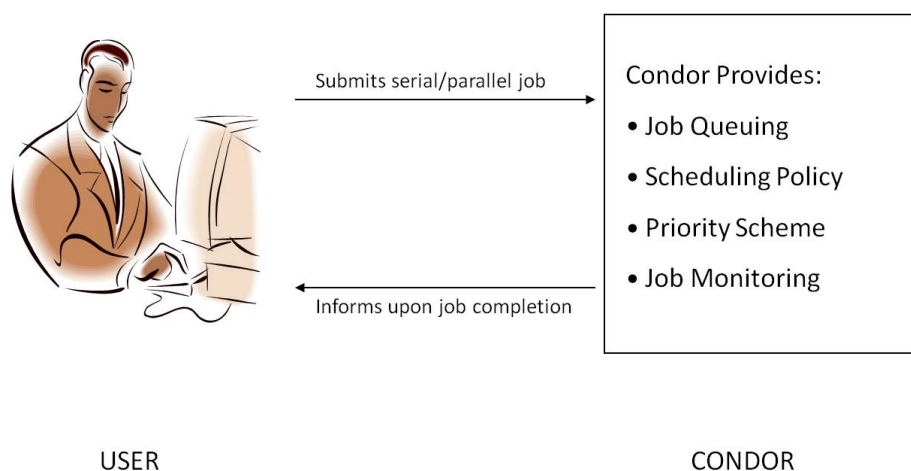


Figure 2.5: Condor resource management system.

Condor system design follows certain rules like transparent placement of background job, automatic restart of a job at new remote location if it fails at one remote location, fair access to workstation to original users while giving remote execution facility and implementation should consume little capacity. A Condor job is scheduled by using amalgamation of centralized and distributed approaches [26].

Submitting a Condor job requires a Condor *submit* script which describes the details of what and how to execute the job. A typical Condor submit file has the following arguments:

- Universe: Specifies the runtime environment to use. It usually depends upon the type of job and the requirements with respect to migration, remote system calls, checkpoint etc.
- Executable: The program to run
- Arguments: Any command line arguments to the executable
- Log: The log file where Condor writes the standard log messages
- Output: The file where the standard output during the program execution is written
- Err: The file where the errors during the program execution are written
- Should_transfer_files: Flag that specifies if the files should be transferred back to the working directory
- Transfer_input_files: The files that need to be transported to the directory where the program execution takes place

2.2.3 Condor DAGMan

The DAGMan (Directed Acyclic Graph Manager) is a meta-scheduler for Condor jobs [27]. It is specially designed to provide a scheduling mechanism for jobs which have a dependency on each other. This dependency is depicted in the form of a Directed Acyclic Graph (DAG).

A DAG is a data structure that represents a workflow of jobs with dependencies [28]. A simple DAG is shown in Figure 2.6. So jobs B and C will not start until job A is completed and job D will not start until jobs B and C are completed.

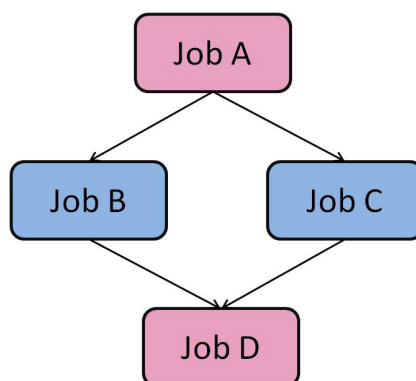


Figure 2.6: An illustrative of a Diamond DAG. After job A is completed , jobs B and C are launched. After jobs B and C are completed, job D is launched.

In the DAGMan terminology, job A is the *Parent* of job B and job C; job B and job C are *Child of* job A. Similarly, job B and job C are *Parents* of job D; job D is the *Child* of job B and job C. Such a DAG is called a *Diamond* DAG. The condor input file used by DAGMan is called a DAG *input* file and Figure 2.7 shows the corresponding input file for the diamond DAG.

```

# DAG.submit file

JOB A A.condor
JOB B B.condor
JOB C C.condor
JOB D D.condor
PARENT A CHILD B C
PARENT B C CHILD D
  
```

Figure 2.7: An illustrative of the Condor DAGMan input file. Each job is defined by the keyword *JOB* . The relationship between jobs is defined by keywords *PARENT* and *CHILD* .

These jobs form the nodes within a DAG. These nodes can further be comprised of sub jobs or elements referred to as PRE script and POST script. They are usually required for simple processing steps that may be required before the actual JOB can be run or for any clean up after the JOB has finished execution. Figure 2.8 shows the elements of a node. Each element needs to be successfully completed to ensure successful completion of a node.

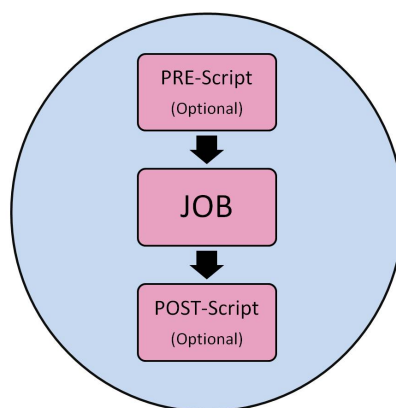


Figure 2.8: An illustrative of the elements of a DAG Node - The PRE and POST scripts, which are optional. Failure of any one of these elements leads to the failure of the node.

In the diamond DAG example earlier, PRE and POST scripts can also be specified.

Figure 2.9 represents an example of such a workflow. So, in this case job A will not start until the PRE script *A1.sh* is successfully completed. After job A is completed, the POST script *A2.sh* is started, irrespective of whether job A was successful or not. If *A1.sh*, A and *A2.sh* are all successfully completed then the node A is considered successful and jobs B and C can be started. Figure 2.10 shows the corresponding input file.

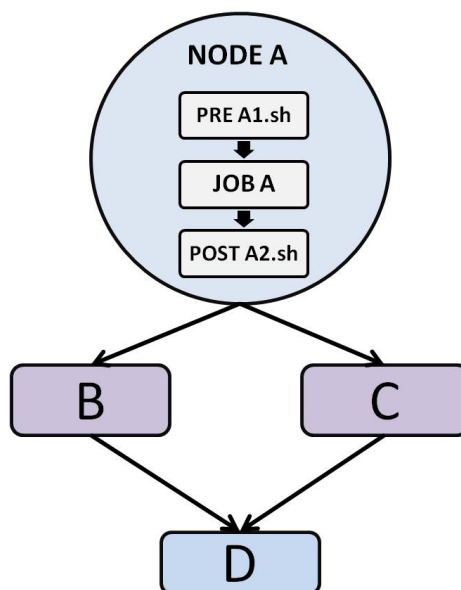
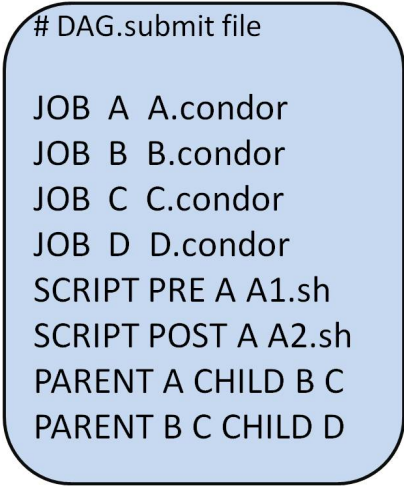


Figure 2.9: Node A is considered complete only when the PRE, JOB and POST elements are completed successfully.



```
# DAG.submit file

JOB A A.condor
JOB B B.condor
JOB C C.condor
JOB D D.condor
SCRIPT PRE A A1.sh
SCRIPT POST A A2.sh
PARENT A CHILD B C
PARENT B C CHILD D
```

Figure 2.10: An illustrative of the Condor DAGMan input file with PRE and POST script. Keywords *SCRIPT PRE* and *SCRIPT POST* are used to define the pre and post script, respectively, for job A.

The DAGman submit file is submitted to Condor using a command *condor_submit_dag* . The DAGMan then submits jobs to Condor in the way these jobs are described in the submit file. A DAGMan, thus, serves as a very handy tool in managing jobs that are components of a huge workflow.

CHAPTER 3

APPROACH

As established in the Section 1.1, a software system is needed which can test a huge chemical library, in a reasonable amount of time. Specifically, we want to have a system which can take as input the complete set of ligands and the protein structure information, and run the docking process for each ligand in parallel on a different processor. After this is achieved, the next step is to predict and implement some selection criteria to preferentially dock some ligands over others, so that we get better hits earlier. In this work, I propose a parallel and distributed software application which makes use of the Marquette computational grid, Pèrè, to execute parallel jobs with minimum human intervention. The software specifically takes care of the following issues:

- Data Partitioning
- Job Submission and Control
- Aggregation of Results

This chapter is presented in two main sections the Basic Approach (Section 3.1) and the Incremental Approach (Section 3.2). Each of the above topics mentioned above are discussed in greater detail in both sections.

3.1 Basic Approach

The basic framework of IncreDock utilizes a cluster of machines or computing resources with a resource scheduling system (Condor). Figure 3.1 shows the basic framework and the overall system design of IncreDock. The input data is in the form of a tar file which is split into multiple independent docking jobs. These jobs are then assigned to individual worker nodes. After each of the processors has completed docking, the results are sent back to the Central Manager or the *head* node. The head node takes these results and after processing combines them into a single result file, which can then be analyzed for potential drug candidates.

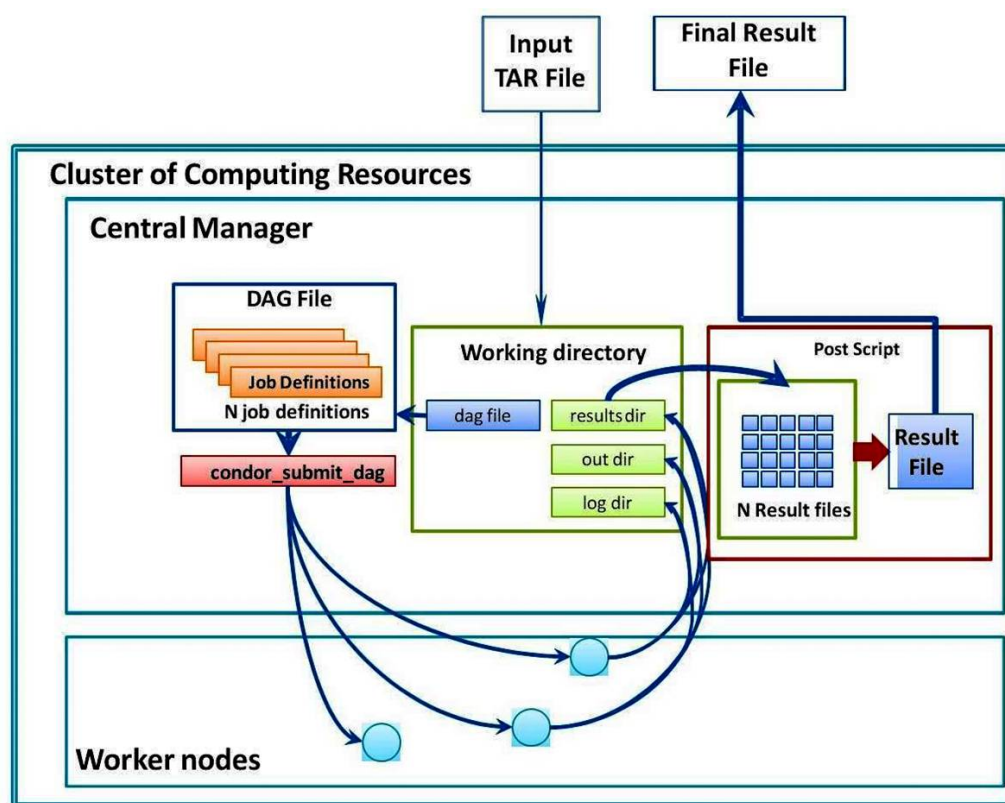


Figure 3.1: An illustration of the architecture of IncreDock. An input file containing the files necessary for a bulk docking experiment is provided. IncreDock partitions the data into single docking events and assigns one docking experiment to each available Condor worker. Upon completion, Condor aggregates the results from the individual workers to present the user with a final single result file.

3.1.1 Data Partitioning

The first step, after the user submits the input data, is the input data partitioning. IncreDock takes as input a tar file, which contains the protein and ligands in the ready to dock format, natural binders (ligands) for comparison and the name of the experiment directory. The natural binders are expected to be named in a format to distinguish them from the test data set. After the software is given all these input details, it generates a working directory for the experiment, the submit description files and a DAG file which contains the job definitions. The Figure 3.2 shows the creation of experiment directory as the first step of using IncreDock.

Generating the DAG definition file requires parsing the input tar file to extract all the ligands, label the ligand data files and the output result files appropriately. The order of these ligand dockings is not dependent on any specific criteria. They are defined in the DAG file based

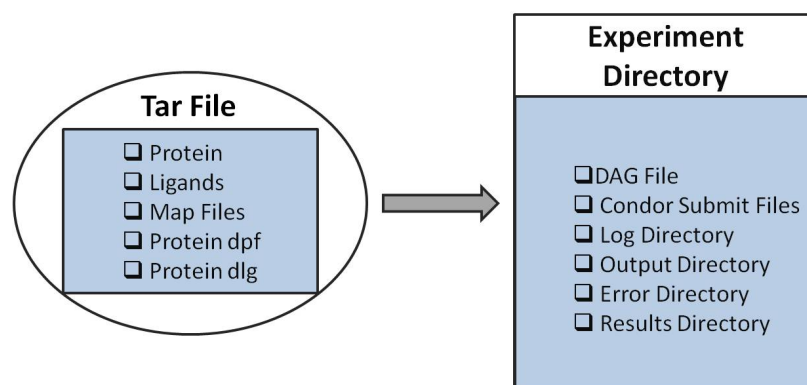


Figure 3.2: The input Tar File is utilized to create the output experiment directory upon launching the dockings.

on the order they are encountered in the input tar file. Finally, invocation of the software at this level presents a command line definition for launching the docking experiment to the user.

3.1.2 Job Submission and Control

After the experiment directory is created and the DAG file generated, the jobs can be submitted by the command generated in the previous step. The submission actually consists of utilizing condor's submit command (*condor_submit_dag*) which takes as input the DAG file. The DAG file in this case is a simple diamond DAG. It shows the dependency between all the individual docking events and the summarize step, which collects the results of all these dockings and combines them into a single result file.

Figure 3.3 shows an extract of the DAG file. The job SUMMARIZE is defined in the summarize.submit file. Each individual ligand gets a job name based on the ligand name, eg. *DOCK_l_cpfm1_p_2A87_a_strip_H* (ligand name: *l_cpfm1_p_2A87*) and has an associated submit file (dock.submit), which is the same for all ligands. The dock.submit file takes as input variables for each ligand and that is specified in the VARS. Finally, a post script for each job is specified which moves the result file to the results directory on the head node. The last line for each definition describes the parent-child relationship between the individual dockings and the last SUMMARIZE job, defined earlier. The DAG file contains the definition for each ligand in the input tarball.

```

### DOCK.DAG ###

JOB SUMMARIZE summarize.submit

JOB DOCK_I_cpfm1_p_1DF7_strip_H dock.submit
VARS DOCK_I_cpfm1_p_1DF7_strip_H COMPOUND="I_cpfm1"
PROTEIN="p_1DF7_strip_H" DPF="I_cpfm1_p_1DF7_strip_H.dpf"
DLG="I_cpfm1_p_1DF7_strip_H.dlg"
Script POST DOCK_I_cpfm1_p_1DF7_strip_H /bin/mv
summary_I_cpfm1_p_1DF7_strip_H.dlg I_cpfm1_p_1DF7_strip_H.dlg results

PARENT DOCK_I_cpfm1_p_1DF7_strip_H CHILD SUMMARIZE
JOB DOCK_I_cpfm10_p_1DF7_strip_H dock.submit
VARS DOCK_I_cpfm10_p_1DF7_strip_H COMPOUND="I_cpfm10"
PROTEIN="p_1DF7_strip_H" DPF="I_cpfm10_p_1DF7_strip_H.dpf"
DLG="I_cpfm10_p_1DF7_strip_H.dlg"
Script POST DOCK_I_cpfm10_p_1DF7_strip_H /bin/mv
summary_I_cpfm10_p_1DF7_strip_H.dlg I_cpfm10_p_1DF7_strip_H.dlg results
:
:
:
N definitions

```

Figure 3.3: A sample of DAG input file.

Figure 3.4 shows the job flow corresponding to the DAG submit file. Each of the N jobs are completed before the job SUMMARIZE is started.

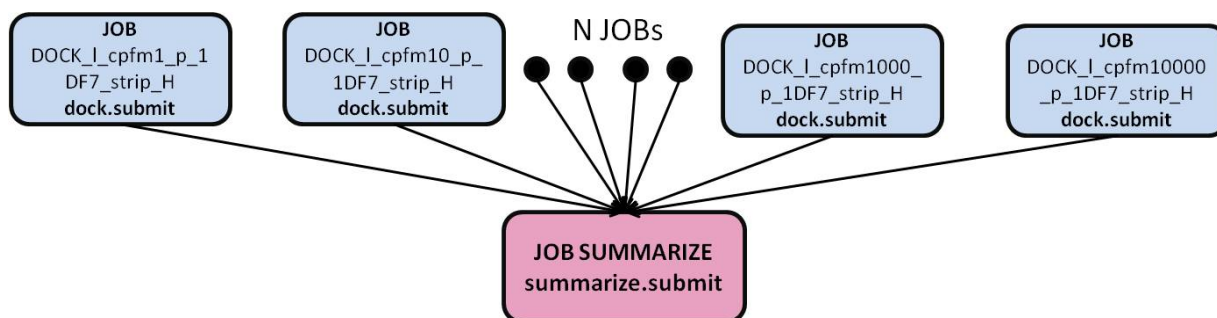


Figure 3.4: An illustrative of parallel submission of the dockings.

After the DAG file is submitted to condor DAGMan, it interacts with the condor scheduler to submit these jobs to the Condor queue, based on the dependencies outlined in the DAG file. The files needed for each individual job or docking experiment along with the executable are transferred to the worker node, which makes it a completely independent docking event.

3.1.3 Aggregation of Results

After the docking is done for each ligand Condor brings back the output file generated, to the head node. As specified in the DAG, the post script moves this file to the results directory on the head node.

After all these ligands have been docked and results for all the dockings obtained, the last step in the DAGMan is a summarize step. The summarize step reads all these results files and sorts them in order of the best energies and ligands, and writes a final summarized output file. This file is what is given back to the user, for further analysis and to continue with other downstream steps in the process of VS. Figure 3.5 shows a sample output file.

```
#dlgn #incluster  #LE  #rmsd #ats #tors
l_MTX_p_1DF7_strip_H, 4,-11.6600, 2.6329, 33, 8
l_MTX_p_1DF7_strip_H, 3,-11.5300, 3.2256, 33, 8
l_MTX_p_1DF7_strip_H, 2,-11.3300, 2.1819, 33, 8
l_MTX_p_1DF7_strip_H, 4,-11.3300, 1.9094, 33, 8
:
:
:
l_cpfm10294_p_1DF7_strip_H, 8, -8.1900, 25.6232, 24, 4
l_cpfm10294_p_1DF7_strip_H, 1, -7.8700, 22.8493, 24, 4
l_cpfm10294_p_1DF7_strip_H, 13, -7.6900, 24.5098, 24, 4
l_cpfm10294_p_1DF7_strip_H, 1, -7.6500, 26.2388, 24, 4
:
:
:
l_cpfm9999_p_1DF7_strip_H, 2, -7.9500, 30.0578, 24, 5
l_cpfm9999_p_1DF7_strip_H, 1, -7.8700, 29.1780, 24, 5
l_cpfm9999_p_1DF7_strip_H, 1, -7.8400, 28.3645, 24, 5
l_cpfm9999_p_1DF7_strip_H, 1, -7.5900, 27.3243, 24, 5
```

Figure 3.5: A sample of the aggregated output result file from the last summarize step.

3.2 Incremental Approach

Section 3.1 laid down the basic structure of a parallel workflow for the docking process. It has been emphasized that the library of such drug like compounds is becoming larger. In order to

scan the entire chemical library using the previous approach, we would need to dock all of these compounds and then look at the results in the end. However, the ambition is to dock an available library, ZINC, which is actually of the order of over 13 *million* compounds. To adopt the approach outlined earlier, a much larger computational facility is needed to get results within a reasonable amount of time. Moreover, it would also be a great misuse of computational power to screen a few potential compounds from this huge drug library. So, the next step was to modify the current architecture to include a way of docking in small increments and look at the results intermittently. With respect to this idea, I developed an incremental approach, utilizing an X-DAG structure, to scale the software for larger input data sets. The process is implemented hoping for screening better compounds in fewer runs. Section 3.2.1 describes the X-DAG structure in more detail.

3.2.1 X-DAG

Figure 3.6 shows another form of a DAG structure. With this kind of dependency, jobs B, C, D will not start until job A is completed; job E will not start until jobs B, C, and D are completed; jobs F, G, H will not start until job E is completed; job I will not start until jobs F, G and H are completed and likewise. This workflow can be visualized like a continued diamond DAG with continued dependencies. Figure 3.7 shows the DAG submit file for such a DAG [29].

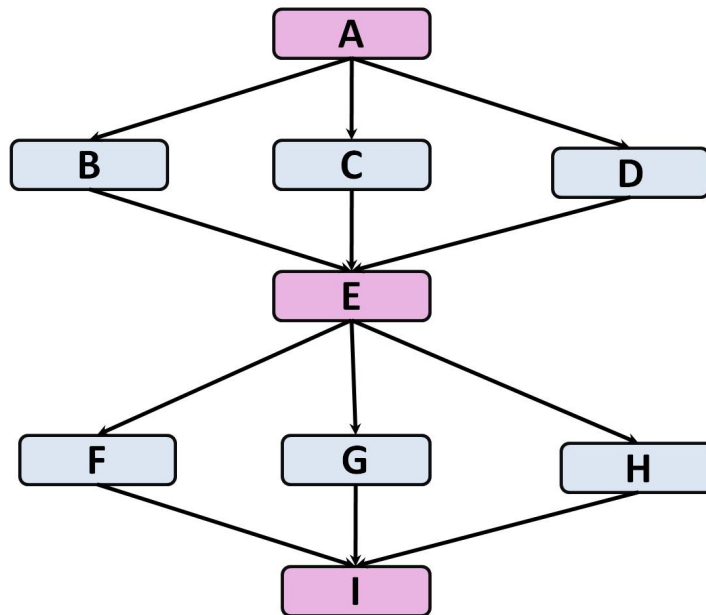


Figure 3.6: An Illustrative of the X-DAG. Jobs B, C, D are launched when job A is completed. Job E is launched when jobs B, C, D are completed. Jobs F, G, H are launched when job E is completed. Job I is launched when jobs F, G, H are completed.

```

# xDAG.submit file

JOB A A.condor
JOB B B.condor
JOB C C.condor
JOB D D.condor
JOB E E.condor
JOB F F.condor
JOB G G.condor
JOB H H.condor
JOB I I.condor

PARENT A CHILD B C D
PARENT B C D CHILD E
PARENT E CHILD F G H
PARENT F G H CHILD I
  
```

Figure 3.7: An illustrative of the Condor DAGMan input file for a X-DAG. Each job is defined by the keyword *JOB* . The relationship between jobs is defined by recursive usage of keywords *PARENT* and *CHILD* .

3.2.2 Data Partitioning

The experiment directory is setup in the same way as described in the basic approach. However, in this approach the order of the ligand dockings is dependent on the results of the previous step. The idea is to selectively dock N ligands in each stage and aggregate the results of these N dockings. The X-DAG approach uses variables as placeholders for actual ligands. The value for each of these placeholders is the ligand name, which is generated dynamically at run time. Figure 3.8 shows a schematic of the incremental approach workflow.

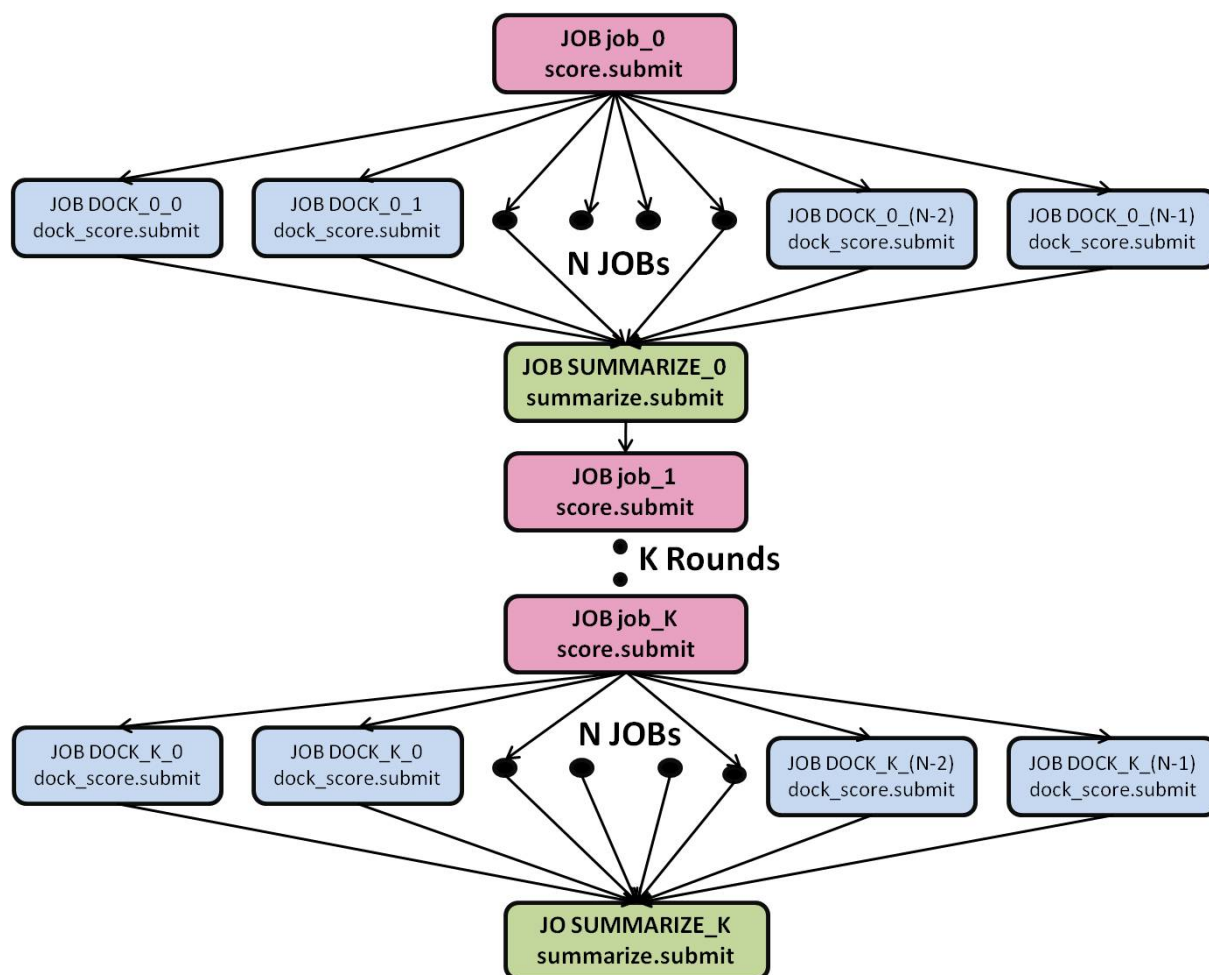


Figure 3.8: An illustrative of the parallel submission of dockings for the Incremental approach.

The first round of dockings is performed by using ligands, selected by a random process. The subsequent rounds are performed by using ligands selected by a scoring function. Two scoring functions were implemented in IncreDock, Random and Bayes:

- **Random Selection:** As seen in the figure, after $(n-1)^{th}$ round the next condor job is an invocation of *score.submit* file. For random selection, this *score.submit* file launches a script, which takes as input the set of ligands not docked yet, and the number of ligands, N that go into the n^{th} round. A list of N ligands is randomly generated and is written to the file that is a mapping between the ligand names and the placeholders in the DAG file. This process is repeated until all the ligands are docked. After each stage of dockings the summarize step is repeated.
- **Bayes Selection:** The chemical nature of a compound is a result of its physical and chemical attributes. As stated in Chapter 2, the Lipinski's Rule-of-Five establishes the importance of four different properties which make a good drug candidate. In order to implement the Bayes selection, these properties (*Molecular Weight, AlogP, Number of Hydrogen Acceptors and Number of Hydrogen Donors*) were used to establish the similarity between *good* ligands and the potential *good* ligands.

The ligands were grouped together based on these properties by dividing the properties into equal sized bins (*numberofbins* = 10). The bins for each of these ligands were determined prior to launching the experiment. For Bayes selection, the *score.submit* file launches a script which implements the Bayesian scoring function. The results file from the $(n-1)^{th}$ stage are sorted, the energies extracted and divided into equal sized bins (*numberofbins* = 5). The energy bin number for each of the ligands docked is then determined. These energy bins and the property bins of all the ligands are then used to calculate and predict the probability of each new ligand falling into a particular energy bin. The probabilities are determined using the following equations:

$$\begin{aligned}
E_L &= \arg \max_E P(E|L) \\
P(E|L) &= \frac{P(L|E)P(E)}{P(L)} \\
P(L|E) &\propto \sum_{i=1}^k P_L(A_i|E) \\
L &= \langle l_1, l_2, \dots, l_k \rangle \\
P_L(A_i|E) &= \frac{N_{(A_i=l_i, E)}}{N_E} \\
P(E) &= \frac{N_E}{N} \\
P(L) &= \alpha
\end{aligned}$$

where, E is the energy bin, L is the ligand for which the energy bin is being determined, E_L is the predicted energy bin for ligand L , $P(E|L)$ is the probability of the ligand being in energy bin E , $P(L|E)$ is the probability of an energy bin given a ligand, $P(E)$ is the probability of any energy bin, $P(L)$ is the probability of occurrence of any ligand, A_i is the i^{th} attribute of a docked ligand, k is the total number of ligand attributes, P_L is the probability of A_i given an energy bin E , l is the i^{th} attribute of an undocked ligand L , N_E is the total number of docked ligands in energy bin E , N is the total number of docked ligands in the round and α is a constant.

If the number of ligands docked or the number of ligands in an energy bin is *zero*, the above probabilities are calculated by introducing a smoothing factor $\lambda = 0.1$. Hence, the new probabilities are:

$$\begin{aligned}
P_L(A_i|E) &= \frac{N_{(A_i=l_i, E)} + \lambda}{N_E + k\lambda} \\
P(E) &= \frac{N_E + \lambda}{N + k\lambda}
\end{aligned}$$

This calculation is done for every ligand that has not been docked yet. The ligands are then sorted based on the value of their expected energy bins. The ones that are predicted to have lower energies (better binders) are then selected to be docked in the next round.

3.2.3 Job Submission and Control

After the DAG file and the X-DAG placeholders created, the DAG is submitted to Condor using the *condor_submit_dag* command. As each stage is completed, the results are used by the scoring function to generate the ligands that go for docking in the next round. Figure 3.9 shows an extract of the DAG file generated for the Incremental approach.

```

### DOCK.DAG ###

JOB job_0 bayes.submit
VARS job_0 JOBID="0"

JOB job_1 bayes.submit
VARS job_1 JOBID="1"
JOB SUMMARIZE_0 summarize.submit
PARENT SUMMARIZE_0 CHILD job_1

JOB job_2 bayes.submit
VARS job_2 JOBID="2"
JOB SUMMARIZE_1 summarize.submit
PARENT SUMMARIZE_1 CHILD job_2
:
JOB DOCK_0_0 dock_score.submit
VARS DOCK_0_0 JOBID="0" NODE="0"
Script POST DOCK_0_0 /bin/mv summary_DOCK_0_0.dlg DOCK_0_0.dlg results
PARENT job_0 CHILD DOCK_0_0
PARENT DOCK_0_0 CHILD SUMMARIZE_0

JOB DOCK_0_1 dock_score.submit
VARS DOCK_0_1 JOBID="1" NODE="0"
Script POST DOCK_0_1 /bin/mv summary_DOCK_0_1.dlg DOCK_0_1.dlg results
PARENT job_0 CHILD DOCK_0_1
PARENT DOCK_0_1 CHILD SUMMARIZE_0
:
:
n levels of definition

```

Figure 3.9: A sample of DAG input file for the Incremental approach.

3.2.4 Aggregation of Results

The scoring function in Incremental approach requires intermediate result files, which are used for predicting the ligands to be docked in the next increment. These intermediate result files are generated from the normal summary files generated after each docking. As seen in the original summary file, each ligand is docked in several poses which are clustered together and the energy for each of these clusters is recorded in the result file.

The energy for each ligand is sorted based on the number of poses in each cluster and the lowest energy within the highest cluster size is chosen as the most favorable energy for a particular ligand. This energy is then called the energy of the ligand and is the one used in the probability calculations. Figure 3.10 shows a sample intermediate result file.

```
dock_id,ligand,energy,num_in_cluster
DOCK_0_0,l_cpfm419,-7.030000,7,
DOCK_0_1,l_cpfm6153,-8.800000,34,
DOCK_0_2,l_cpfm9895,-8.990000,2,
DOCK_0_3,l_cpfm369,-8.930000,18,
DOCK_0_4,l_cpfm8540,-7.800000,19,
:
:
DOCK_10_0,l_cpfm9505,-8.560000,3,
DOCK_10_1,l_cpfm9280,-8.240000,1,
DOCK_10_2,l_cpfm3100,-7.480000,6,
DOCK_10_3,l_cpfm1012,-6.930000,7,
:
:
DOCK_20_496,l_cpfm1432,-5.860000,23,
DOCK_20_497,l_cpfm683,-5.640000,1,
DOCK_20_498,l_cpfm7672,-5.730000,1,
DOCK_20_499,l_cpfm9780,-6.290000,2,
```

Figure 3.10: A sample output result file for Incremental approach using Bayesian scoring.

For a complete docking experiment there can be constraints on time, resources and cost. IncreDock gives the user flexibility in choosing the way he wants to run his dockings. The Basic approach can be used if the user is interested in seeing the results of all the dockings and then choosing the best ones from the summarized results. The Incremental approach can be used if the users have a constraint on time and resources. They can choose to run several rounds of dockings

and look at results from each round. If they are satisfied with the results they can just stop the dockings else continue with the next round. Thus, IncreDock is built to provide choices of launching an experiment which take into consideration user constraints.

CHAPTER 4

EVALUATION AND RESULTS

Chapters 1 and 2 discussed the background and the approach to address the problem of incremental docking of large ligand libraries. The main objective of the experiments conducted is to demonstrate that earlier rounds of docking are enriched for better binding ligands, than if docked in a random order. Six experiments on four distinct proteins were performed, to evaluate the performance of the incremental docking approach described in Chapter 3.

In this chapter, Sections 4.1 and 4.2 discuss the chemical library and the proteins used for experiments. Several metrics were used to evaluate changes in binding energy and enrichment of top scoring compounds. Section 4.3 describes these metrics in detail. Section 4.4 describes the computational environment used to carry out these experiments. Finally, Sections 4.5 and 4.6 present the results for each protein and a summary discussion of results for all proteins used in the study.

4.1 Chemical Library

The size of available chemical space is very huge and it is important to limit the size to get a focused, synthesizable chemical library. This thesis work utilizes the in-house physical collection of 10,573 ligands in the Chemical Proteomics Facility at Marquette (CPFM) [5]. The library contains drug-like molecules, selected on the basis of their predicted binding to dehydrogenases and kinases. The ligands were converted into a ready-to-dock format using in Autodock tools [30]. The procedure for preparing the ligands is given in Appendix A.1.

4.2 Proteins

The CPFM ligand data set was used to perform docking experiments on four different proteins. The proteins were chosen, so that the drugs related to them have implications to two different types of diseases/health issues. The first set of proteins is Dihydrofolate Reductase (DHFR), Dihydrodipicolinate reductase (DHPR) and Thioredoxin Reductase (TrxR). These

proteins are related to the disease Tuberculosis. Roughly, a third of the world's population has been infected with *M. tuberculosis* (the bacteria that is the causative agent of Tuberculosis). New Tuberculosis infections occur at a rate of one per second [31]. However, global Tuberculosis control is challenged by the emergence of drug-resistant strains. Discovery of new drugs is, therefore, imperative for a global cause.

DHFR catalyzes the NADPH-dependent reduction of dihydrofolate to tetrahydrofolate and is essential for the synthesis of several amino acids for growth and sustenance. Its inhibition leads to arrest of DNA synthesis and eventual cell death. DHFR is a potential target of the most widely used anti-tuberculosis drugs, Isoniazid to which there is growing resistance [32]. Hence, inhibitors of DHFR can be potential drug candidates for treatment of tuberculosis. DHPR catalyzes the reduced pyridine nucleotide-dependent reduction of the α,β -unsaturated cyclic imine, dihydrodipicolinate, to generate tetrahydrodipicolinate. It is involved in the bacterial biosynthetic pathway that generates meso-diaminopimelate, a component of bacterial cell walls [33]. Inhibition of DHPR may lead to cell death. TrxR catalyses the NADPH-dependent reduction of thioredoxin disulfide and other oxidized cell constituents. It is vital to keep the cell in a reduced state and its inhibition can prevent survival from oxidative stress which makes it a good drug target [34]. The 3D structure and functional properties of *M. tuberculosis* DHFR and TrxR reveal new opportunities for designing new drugs [35, 36].

The last protein Human Dual Specificity Phosphatase 5 (DUSP5) is an enzyme in humans encoded by the DUSP5 gene. It inactivates the kinases in the Mitogen Activated Protein (MAP) kinase family, which have a function in cell growth and differentiation [37, 38]. Inhibition of DUSP5 can lead to loss of cellular proliferation, which is important in arresting tumor growth and other infectious diseases [39]. Hence, DUSP5 makes a good drug target and needs further exploration. DUSP5 protein has two domains and each of these domains participate in ligand binding. For the experiment purposes, each of these domains was tested individually. These domains are named as DUSP5R and DUSP5C.

The protein structural files are not in the ready-to-dock format, because of missing atoms and chain breaks, to name a few. Hence, they need to be edited and prepared using Autodock Tools and Autogrid, which allow the visualization, analysis and preparation protein molecules for docking. The procedure for preparing the proteins is given in Appendix A.2 and Appendix A.3.

After the datasets were prepared, IncreDock was used to perform docking experiments using the Basic approach and then the Incremental approach with Bayesian scoring function. The results for the Basic Approach were sorted and the top 1000 ($\sim 10\%$) best binders were identified. The Incremental approach was validated against these best binders using the performance metrics described in the next section.

4.3 Performance Metrics

As explained in Chapter 3 , each ligand is sorted based on the number of poses in each cluster. The lowest energy within the highest cluster size, is then chosen as the most favorable energy for a particular ligand. This energy is defined as the *energy of the ligand* . To establish the proof of concept and to assess the performance of the Incremental approach, following parameters were evaluated:

- **Average Energy of a round:** For each round of dockings, the average value of the energy for all the ligands in that round is calculated. This is defined as the *Average Energy of a round* and is calculated as:

$$\text{Average Energy} = \frac{\sum \text{Energy of each ligand in the round}}{\text{Number of ligands in the round}}$$

- **Ligand Enrichment:** The results from the Basic Approach were sorted in an ascending order by the energy of the ligands. *Best* ligands are defined as the best binders for given a protein. The best 10% (≈ 1000) of these are selected and called the top 10% ligands. *Ligand Enrichment* is then defined and calculated as:

$$\text{Ligand Enrichment} = \text{Number of top 1000 ligands docked in a round.}$$

It is a measure of how the Incremental approach enriches the process docking by docking more of the best binders in initial rounds of docking.

- **Cumulative Ligand Enrichment:** It is an overall measure of how much each round is enriched in the process of incremental docking. This value is expected to attain the top 10%

number or contain all the ligands that are the top 10% ligands in the early rounds of dockings. It is defined as:

Cumulative Ligand Enrichment = Total number of top 1000 ligands docked so far.

- **Statistical Test of Significance:** A binomial parameter test was done to find if the scoring function of IncreDock has any effect on enrichment, specifically in rounds 2 and 3 [40]. The test provides information to establish the independence or dependence of enrichment in round 1 and combined enrichment in rounds 2 and 3. The following steps were performed to calculate the value of t :

1. State the null hypothesis, H_o :

$$H_o : \pi = \pi_0$$

2. State the alternate hypothesis, H_a :

$$H_a : \pi > \hat{\pi}$$

3. Calculate the standard deviation, $\sigma_{\hat{\pi}}$:

$$\sigma_{\hat{\pi}} = \sqrt{\frac{\hat{\pi}(1 - \hat{\pi})}{n}}$$

4. Calculate Test Statistic, z :

$$z = \frac{\hat{\pi} - \pi_0}{\sigma_{\hat{\pi}}}$$

5. Based on the z value accept or reject the null hypothesis, H_o .

where, π_0 is the fraction of *good* ligands left after docking round 1, π is the actual fraction of *good* ligands docked in rounds 2 and 3, $\hat{\pi}$ is the expected fraction of *good* ligands docked in rounds 2 and 3, $\sigma_{\hat{\pi}}$ is the standard deviation, and n is the total number of ligands docked in rounds 2 and 3.

Rejecting the null hypothesis or accepting the alternate hypothesis means that scoring function implemented in Incredock has an effect of selecting better binders, specifically in rounds 2 and 3.

- **Top 0.01 % Ligands:** The actual number of ligands than can be experimentally tested in a laboratory is much smaller than 10% of the library, and can be of the order of 0.01% or less. Hence, the benefit of the Incremental Approach is when it can identify and dock the top 0.01% ligands in early rounds. So, Ligand Enrichment and Cumulative Enrichment were re-calculated with a threshold level of 0.01% (≈ 10). Hence,

$$\text{Ligand Enrichment} = \text{Number of top 10 ligands docked in a round.}$$

$$\text{Cumulative Ligand Enrichment} = \text{Total number of top 10 ligands docked so far.}$$

Further, to support the results and get a visual representation of the fact that the dockings gave good results, a docking complex picture of the best docking in the second round for each protein generated using Pymol [41].

4.4 Experimental Setup

Marquette University's Père Cluster was used for performing the experiments. The cluster is composed of 1284 nodes, $2 \times$ quad core Intel Xeon X5550, total of 1024 cores. The processors feature 24 GB RAM per node, DDR Infiniband backbone, 20 Gb/s, Red Hat Enterprise Linux 5.3.

IncreDock and other custom scripts for the analysis of Incremental Approach was installed on Père Cluster. Dockings were performed using a freely available tool, Autodock4. The proteins and ligands were processed and using the MGLTools [42]. Condor 7.4.4 for X86_64-LINUX_RHEL5 is used a resource scheduler for implementing High Throughput Computing. Finally, Python 2.4.3 was used as the language for development [43].

4.5 Results

Dockings were performed for each of the proteins listed in Section 4.2 using the Incremental approach described in Chapter 3. The number of ligands docked in each round is 500. The ligand library has 10,573 ligands, so the total number of rounds was 22, with the last round having 73 ligands. If the dockings were performed in a random fashion, it would be expected that the the number of best ligands in each round is uniform. So, if 1000 best ligands are identified from a total of 10,573 ligands; the probability, p , of choosing one best ligand can be calculated as:

$$p = \frac{1000}{10573} \approx 0.095$$

Hence, the number of top ligands in each random round is estimated as:

$$\text{Number of best ligands in each round} = 500 \times p \approx 50 \text{ ligands}$$

So, each round will contribute almost 50 best ligands in a random selection process. The proteins described in Section 4.2 were docked against the CPFPM ligand library. Performance metrics were evaluated for each of these proteins and compared with the expected random performance. Subsections 4.5.1 to 4.5.5 presents the results, observations and discussion for each of the proteins, followed by Section 4.6 which presents the overall results and discussion.

4.5.1 *Dihydrofolate Reductase* (DHFR)

It is observed that the average energy for the first random round is -7.72366 and it drops significantly in rounds 2 and 3. After that it starts increasing but is still lower than round 1 till around round 5. After round 5 the average energy keeps increasing and hits its maximum value in round 22. Figure 4.1 shows the plot between the average energy of a round and round number.

Ligand Enrichment can be seen in Figure 4.2, which shows the plot between the number of top 1000 ligands found in a round against the percentage of ligands docked. It is observed that docking only 5% of the ligand library in round 1 gives an enrichment of 44 ligands which is close to the expected value 50 ligands (indicated by the red line of random selection process). However, round 2 and 3 are greatly enrichment by top ligands; almost 33% of the best 1000 ligands can be

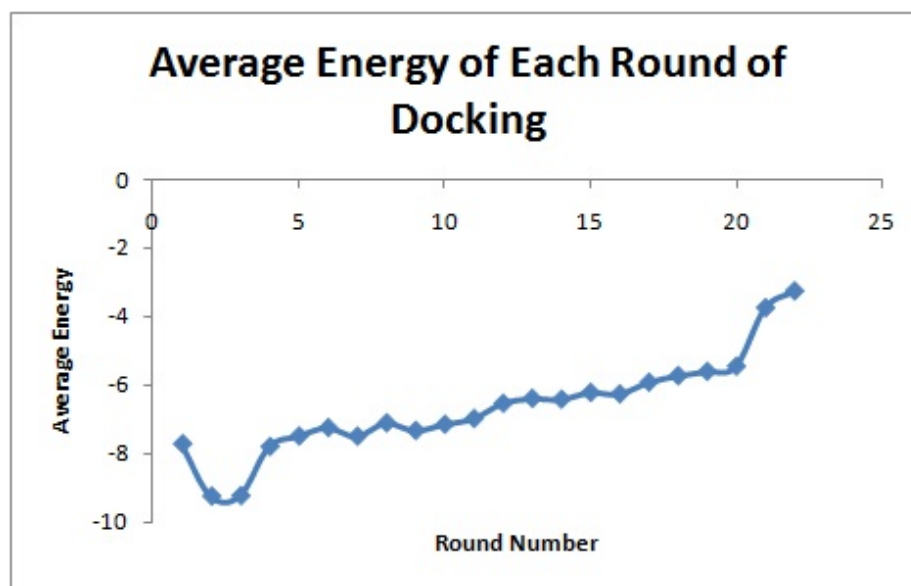


Figure 4.1: Plot between average energy of a round and round number for DHFR. round 2 and 3 shows a significant drop in average energy.

extracted by docking only 14% of the ligand library. All other rounds have an enrichment which is actually lesser than the random expectation of 50 ligands.

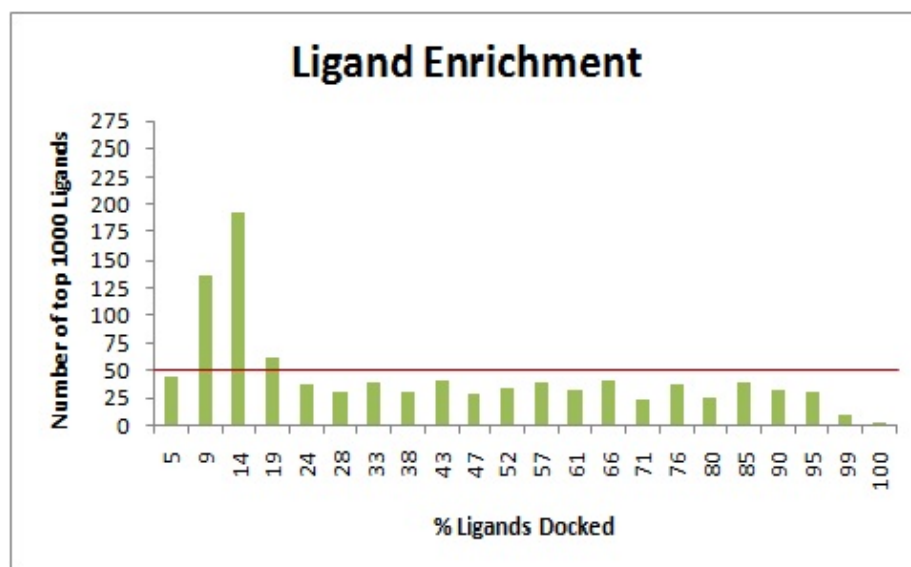


Figure 4.2: Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.

Cumulative Ligand Enrichment is yet another representation of ligand enrichment process.

Figure 4.3 shows the plot between the cumulative number of top 1000 ligands found and the

percentage of ligands docked. It can be seen that the slope of the curve increases rapidly between rounds 2 and 3, indicating higher rate of enrichment. After that, it attains a gradual slope and eventually flattens out depicting little or constant change in the rate of enrichment.

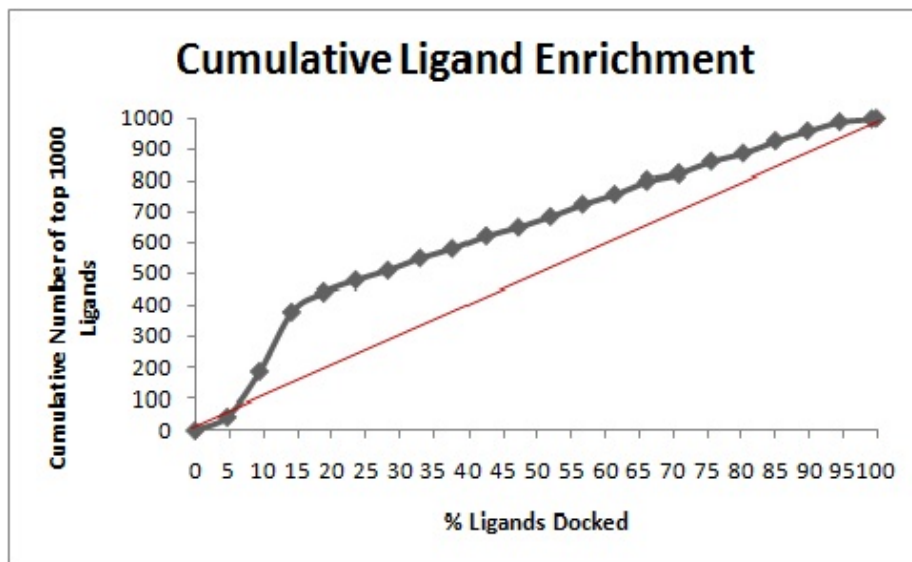


Figure 4.3: The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red) till 35% ligands are docked and reasonable higher for complete docking.

Next, the test statistic for DHFR is computed. Let G be the total number of best ligands for testing, G_i be the number of best ligands in round i , N be the total number of ligands and N_i be the total number ligands docked in round i , then

$$G = 1000$$

$$G_1 = 44$$

$$G_2 = 137$$

$$G_3 = 193$$

$$N = 10573$$

$$N_1 = 500$$

$$N_{(2+3)} = 1000$$

$$\begin{aligned}
\pi_0 &= \frac{G - G_1}{N - N_1} \\
&= \frac{1000 - 44}{10573 - 10073} \\
&= 0.095 \\
\hat{\pi} &= \frac{G_2 + G_3}{N_{(2+3)}} \\
&= \frac{137 + 193}{1000} \\
&= 0.33 \\
\sigma_{\hat{\pi}} &= \sqrt{\frac{\hat{\pi}(1 - \hat{\pi})}{N_{(2+3)}}} \\
&= \sqrt{\frac{0.095(1 - 0.095)}{1000}} \\
&= 0.0093 \\
z &= \frac{\hat{\pi} - \pi_0}{\sigma_{\hat{\pi}}} \\
&= \frac{0.33 - 0.095}{0.0093} \\
&= 25.27
\end{aligned}$$

The value of test-statistic $z = 25.27$, which is greater than 3.71. Hence, we can reject the null hypothesis at 0.001 level and say that the scoring function in Incredock does a highly significant selection of top ligands in rounds 2 and 3 for DHFR.

Figure 4.4 shows the predicted docking complex of DHFR with the best ligand in the second round of docking. It is observed that the ligand (represented by stick model) is strategically placed in the binding pocket of the protein molecule (represented by blue). The relevant amino acid side chains are shown to illustrate potential intermolecular binding interactions.

Looking at the results for DHFR, it can be suggested that the docking can be interrupted after round 4, as almost 40% of the top 1000 ligands can be extracted by docking only about 15% of the ligand library.

Finally, Ligand Enrichment (Figure 4.5) and Cumulative Ligand Enrichment (Figure 4.6) was evaluated for top 10 (0.01%) ligands.

PyMOL for evaluation only.

Open
File: Toggle Dist
File: Zoom All

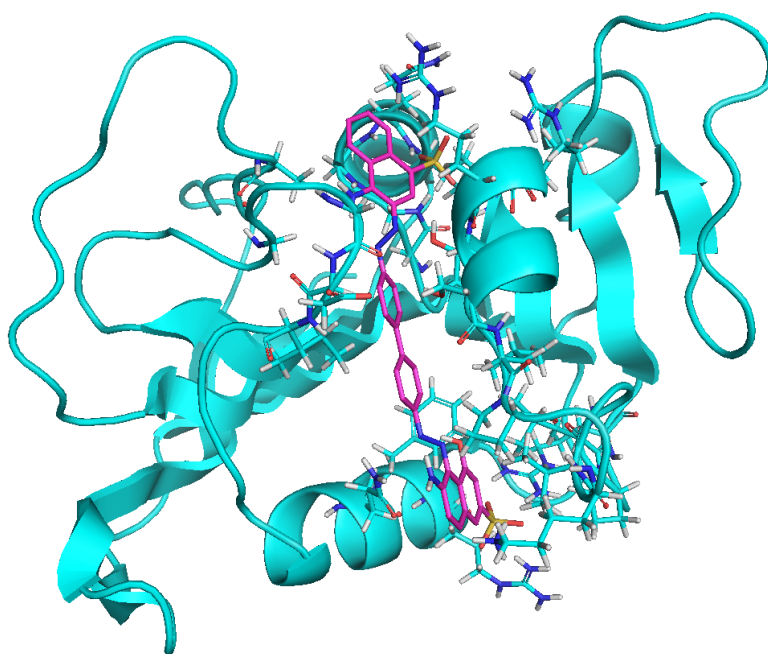


Figure 4.4: Docking Complex of DHFR with the ligand 103 in round 2.

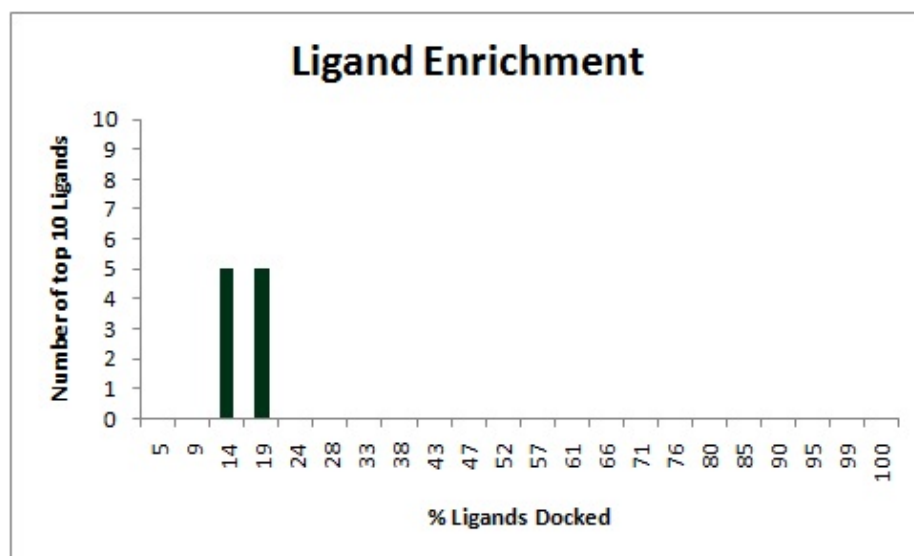


Figure 4.5: Ligand Enrichment for top 10 ligands.

It is observed that 5 top ligands are docked by exploring 14% of the ligand library and all 10 top ligands are docked by exploring only 19% of the entire ligand library. It can be concluded that to test the top 10 binders for DHFR from a library of 10,573 ligands, only 19% of the ligands need to be selectively docked.

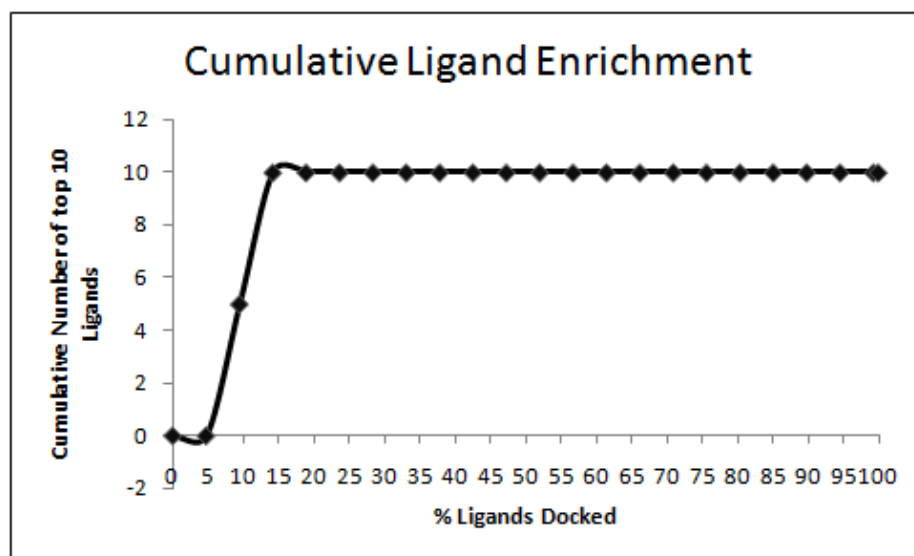


Figure 4.6: Cumulative Ligand Enrichment for top 10 ligands.

4.5.2 Dihydrodipicolinate reductase (DHPR)

The docking results of DHPR show that the average energy for the first random round is -7.31914 . The average energy drops significantly in round 2 and starts increasing till about round 7, where it is similar to the energy in round 1. After round 7, it keeps increasing and hits maximum average energy in round 22. Figure 4.7 shows the plot between the average energy of a round and round number.

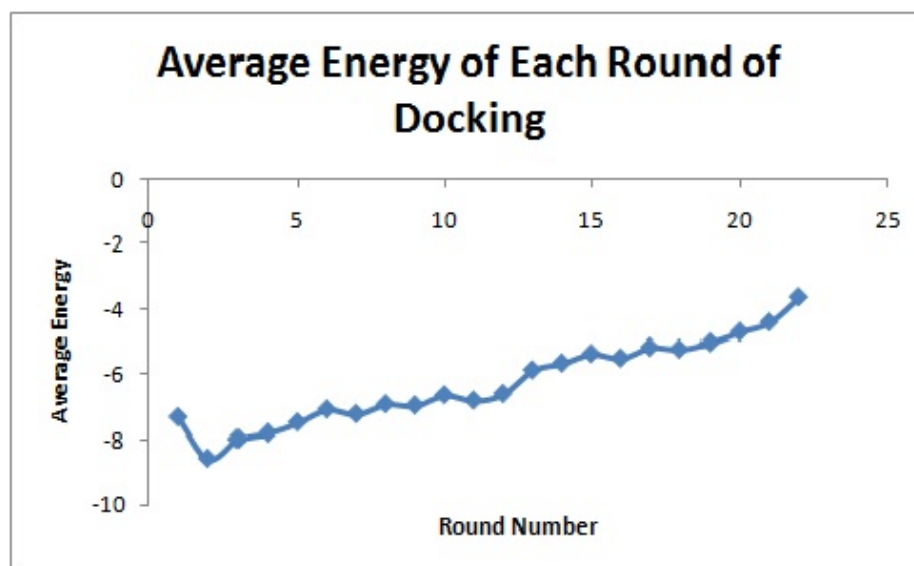


Figure 4.7: Plot between average energy of a round and round number for DHPR. Round 2 shows a significant drop in average energy.

Ligand Enrichment can be seen in Figure 4.8, which shows the plot between the number of top 1000 ligands and the percentage of ligands docked for DHP. It is observed that docking only 5% of the ligand library in round 1, gives an enrichment of 56 ligands, which can be expected in the natural random selection process indicated by the red line (50 ligands). Round 2 gives a massive enrichment of 200 ligands, which alone accounts for 20% of the best 1000. Round 3 and 4 have good enrichment, accounting for another 22%. Almost 50% of the best 1000 ligands can be extracted in the rounds 2, 3 and 4, by docking only 19% of the ligand library. All other rounds up to round 13, have an enrichment which is actually below the random expectation of 50 ligands. It can be seen that after docking almost 71% of the library little or very insignificant enrichment is achieved.

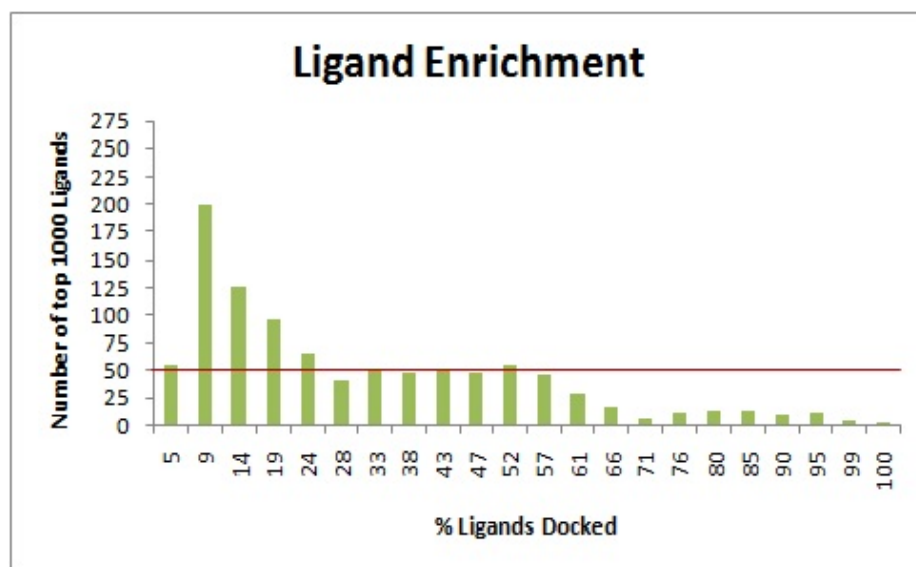


Figure 4.8: Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.

Cumulative Ligand Enrichment is seen in Figure 4.9, which shows the plot between the cumulative number of top 1000 ligands found and the percentage of ligands docked. It is seen that the slope of the curve is high for rounds 2 and 3, and after that it attains a gradual slope indicating high initial enrichment. The curve flattens out after docking almost 70% ligands depicting very little or insignificant enrichment at that point. The plot shows that cumulative enrichment is much better than a constant enrichment, depicted by red line.

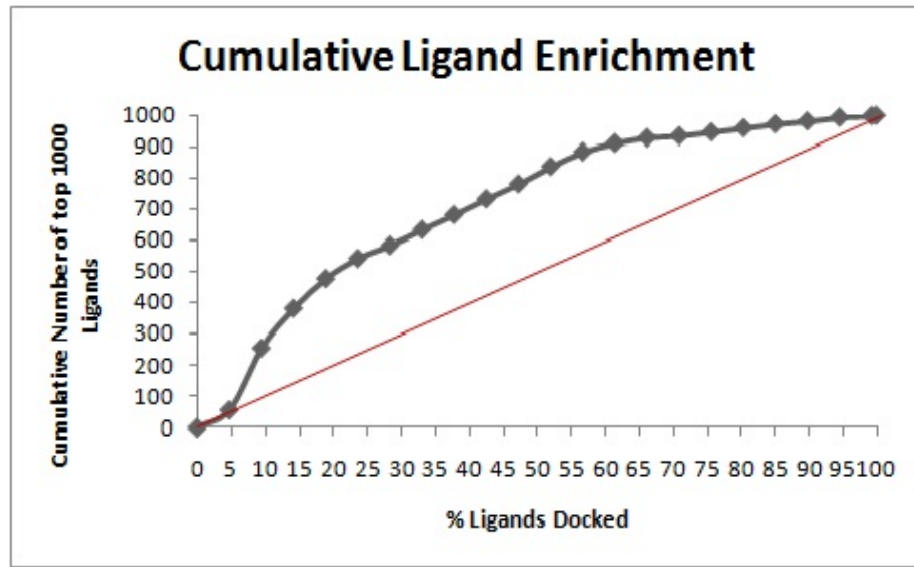


Figure 4.9: The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red) for the entire docking process.

Next, the test statistic for DHPR is computed. Let G be the total number of best ligands for testing, G_i be the number of best ligands in round i , N be the total number of ligands and N_i be the total number ligands docked in round i , then

$$G = 1000$$

$$G_1 = 56$$

$$G_2 = 200$$

$$G_3 = 126$$

$$N = 10573$$

$$N_1 = 500$$

$$N_{(2+3)} = 1000$$

$$\begin{aligned} \pi_0 &= \frac{G - G_1}{N - N_1} \\ &= \frac{1000 - 56}{10573 - 10073} \end{aligned}$$

$$\begin{aligned}
&= 0.0954 \\
\hat{\pi} &= \frac{G_2 + G_3}{N_{2+3}} \\
&= \frac{200 + 126}{1000} \\
&= 0.326 \\
\sigma_{\hat{\pi}} &= \sqrt{\frac{\hat{\pi}(1 - \hat{\pi})}{N_{(2+3)}}} \\
&= \sqrt{\frac{0.094(1 - 0.094)}{1000}} \\
&= 0.0092 \\
z &= \frac{\hat{\pi} - \pi_0}{\sigma_{\hat{\pi}}} \\
&= \frac{0.326 - 0.094}{0.0092} \\
&= 25.22
\end{aligned}$$

The value of test-statistic $z = 25.22$, which is greater than 3.71. Hence, we can reject the null hypothesis at 0.001 level and say that the scoring function in Incrdock does a highly significant selection of top ligands in rounds 2 and 3 for DHPR.

Figure 4.10 shows the predicted docking complex of DHPR with the best ligand in the second round of docking. It is observed that the ligand (represented by stick model) strategically occupies the binding pocket in the protein molecule (represented by purple). It is seen from the picture that the ligand amino acid side chains interact well with the protein alpha sheets and the hairpin bends.

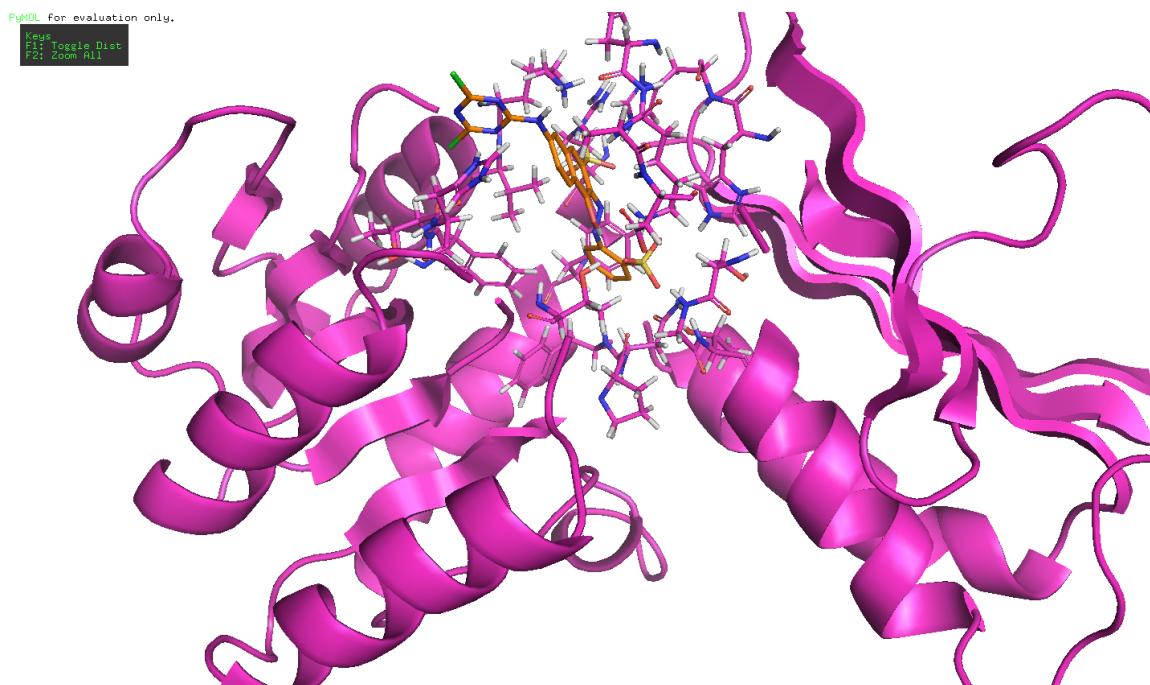


Figure 4.10: Complex of DHPR with the ligand 210 in round 2.

Looking at the results for DHPR, it can be suggested that the docking can be interrupted after round 4, as almost 50% of the best ligands can be extracted by docking only about 20% of the ligand library. Also, the results suggest that the docking can definitely be stopped after round 14 (66% ligands) and the last 8 rounds be omitted, since almost no enrichment occurs then.

Finally, Ligand Enrichment (Figure 4.11) and Cumulative Ligand Enrichment (Figure 4.12) was evaluated for top 10 (0.01%) ligands. It is observed that 7 top ligands are docked by exploring only 9% of the ligand library and all 10 top ligands are docked by selectively docking only 14% of the entire ligand library.

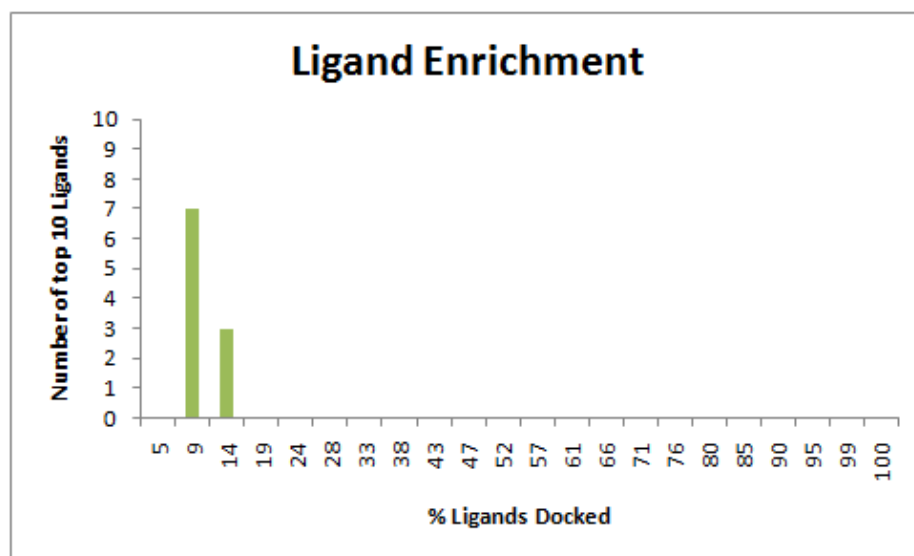


Figure 4.11: Ligand Enrichment for top 10 ligands.

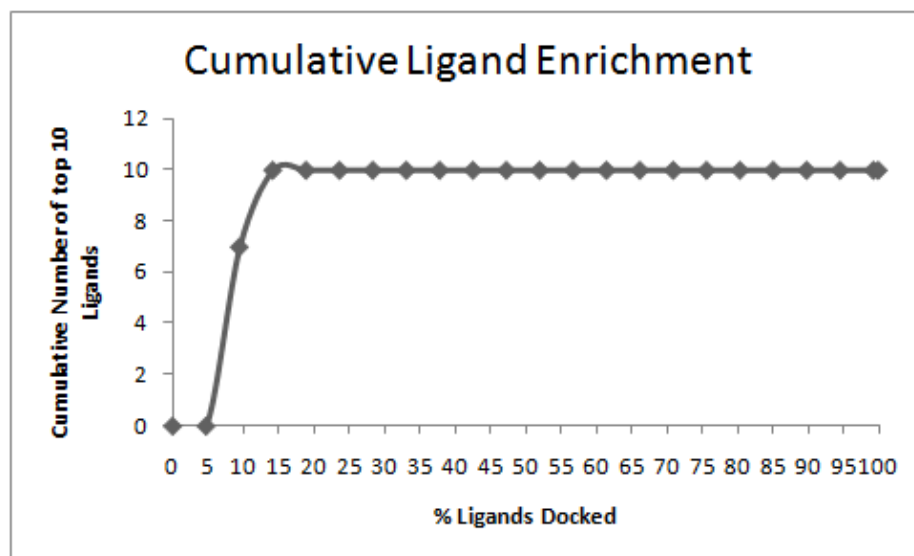


Figure 4.12: Cumulative Ligand Enrichment for top 10 ligands.

4.5.3 *Thioredoxin Reductase* (TrxR)

The docking results of TrxR, show that the average energy for the first random round is -7.21082 and it drops slightly in round 2. After round 3, it starts increasing at a gradual pace and achieves a near constant value in the subsequent rounds. It is observed that the maximum average energy occurs in round 22.

Unlike DHFR and DHPR, after round 2 TrxR does not show any drastic changes in the average energy of a round. Figure 4.13 shows the plot between the average energy of a round and round number.

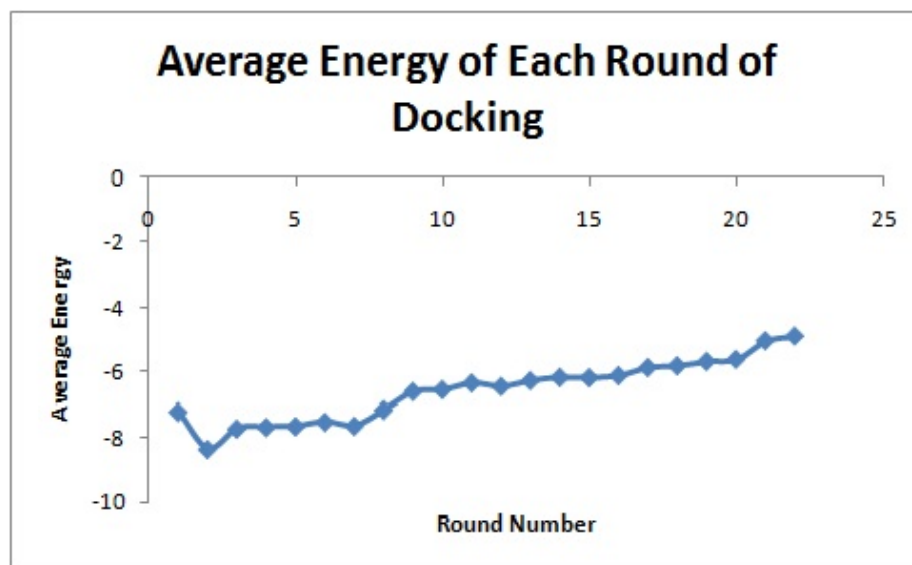


Figure 4.13: Plot between average energy of a round and round number for TrxR. round 2 shows a decreased average energy but there is no great change in the average energy of other rounds.

Ligand Enrichment is seen in Figure 4.14, which shows the plot between the number of top 1000 ligands and the percentage ligands docked. It is observed that docking only 5% of the ligand library in round 1 gives an enrichment of 37 ligands. Round 2 gives a slightly better enrichment of 49 ligands, but no great improvement. It is also observed that the enrichment is not any better in the subsequent rounds, than it would have been if the docking were done randomly. This is illustrated by the fact that the green bars of enrichment are close to the red line depicting constant enrichment.

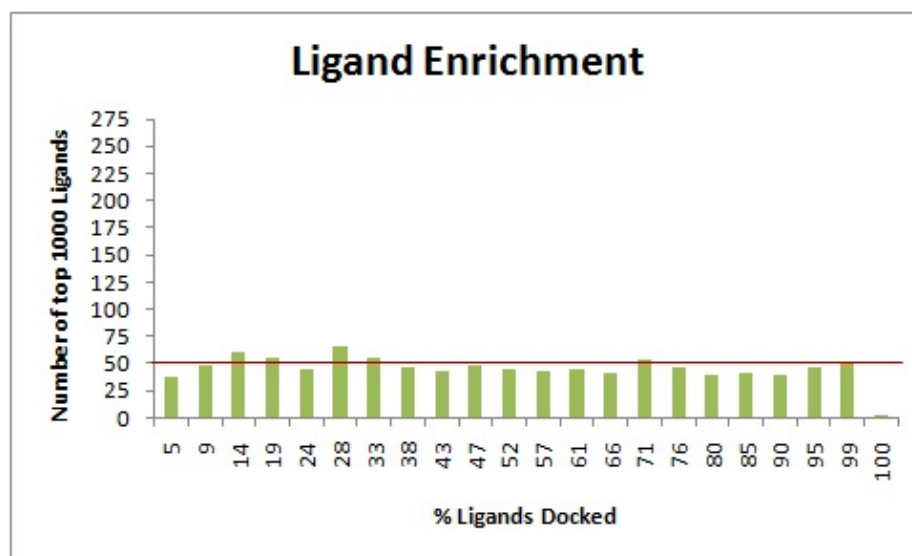


Figure 4.14: Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.

Cumulative Ligand Enrichment, shown in Figure 4.15, depicts that the cumulative growth in prediction of good ligands is more or less linear. It almost overlaps the curve depicting a linear cumulative growth in enrichment (red). This indicates that even though some selection happened, it wasn't significant.

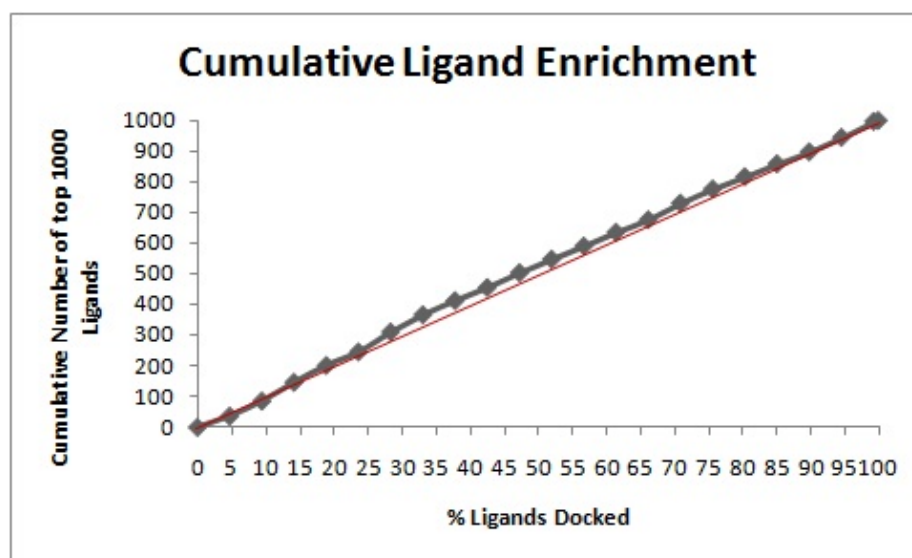


Figure 4.15: The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is almost the same as in case of random selection (Red) for the entire docking process.

Next, the test statistic for TrxR is computed. Let G be the total number of best ligands for testing, G_i be the number of best ligands in round i , N be the total number of ligands and N_i be the total number ligands docked in round i , then

$$G = 1000$$

$$G_1 = 37$$

$$G_2 = 49$$

$$G_3 = 61$$

$$N = 10573$$

$$N_1 = 500$$

$$N_{(2+3)} = 1000$$

$$\begin{aligned}\pi_0 &= \frac{G - G_1}{N - N_1} \\ &= \frac{1000 - 37}{10573 - 500}\end{aligned}$$

$$= 0.096$$

$$\begin{aligned}\hat{\pi} &= \frac{G_2 + G_3}{N_{2+3}} \\ &= \frac{49 + 61}{1000}\end{aligned}$$

$$= 0.11$$

$$\begin{aligned}\sigma_{\hat{\pi}} &= \sqrt{\frac{\hat{\pi}(1 - \hat{\pi})}{N_{(2+3)}}} \\ &= \sqrt{\frac{0.096(1 - 0.096)}{1000}}\end{aligned}$$

$$= 0.0093$$

$$\begin{aligned}z &= \frac{\hat{\pi} - \pi_0}{\sigma_{\hat{\pi}}} \\ &= \frac{0.11 - 0.096}{0.0093}\end{aligned}$$

$$= 1.505$$

The value of test-statistic $z = 1.505$, which is not greater than 1.645. So H_0 cannot be rejected at 0.05 level. The p-value in this case is 0.066, which means that the scoring function in Incredock shows some enrichment but is not significantly better than random dockings in case of TrxR.

Figure 4.16 shows the predicted docking complex of TrxR with the best ligand in the second round of docking. It is observed that the ligand end side chains (represented by stick model) have specific interactions with the protein molecule (represented by green). It is seen from the picture that the ligand molecule is placed in a binding cavity in the protein and has intermolecular binding interactions with the protein side chains.

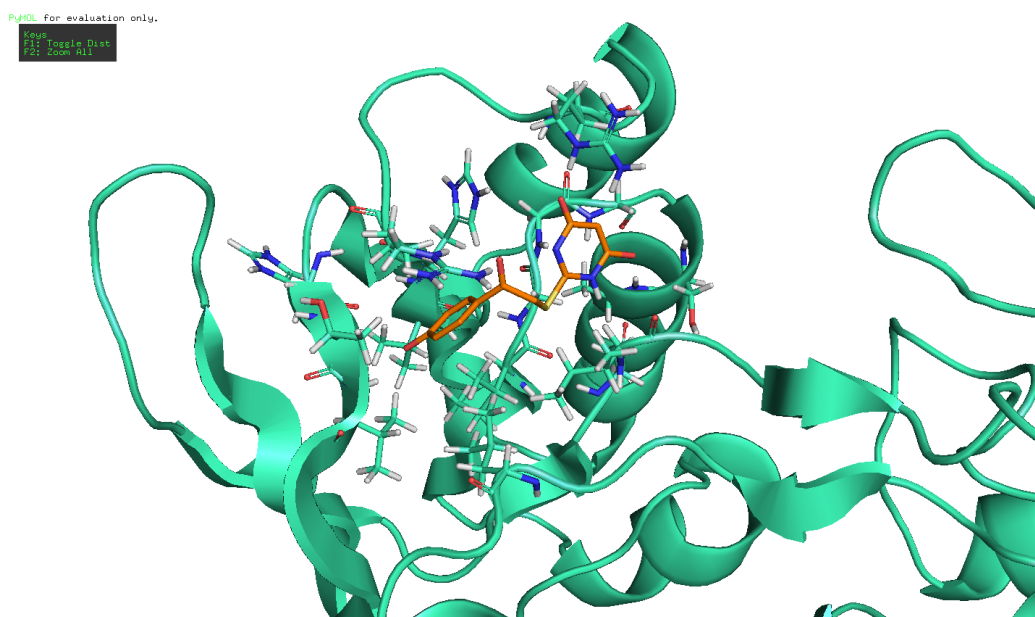


Figure 4.16: Docking Complex of TrxR with the ligand 410 in round 2.

Finally, Ligand Enrichment (Figure 4.17) and Cumulative Ligand Enrichment (Figure 4.18) for top 10 (0.01%) ligands was evaluated. It is observed the top 10 ligands are randomly docked across all the rounds. There is no visible selection in earlier rounds of dockings.

Although, the incremental docking results for TrxR are not significant in terms of ligand enrichment, the average energy improved for round 1. The scoring function predicts ligands based on the their likely energy of binding. It is interesting to observe that the average energy for TrxR does not vary too much across different rounds. This suggests that there are some other features

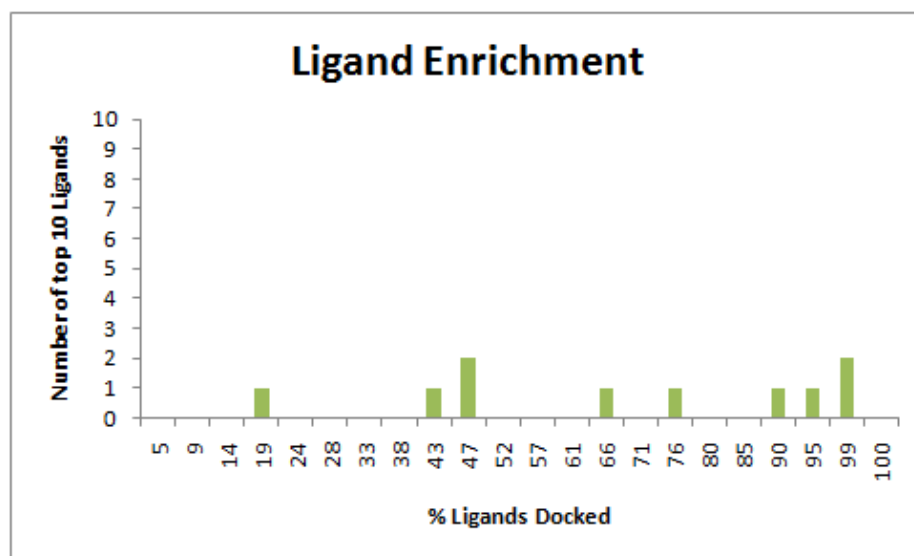


Figure 4.17: Ligand Enrichment for top 10 ligands.

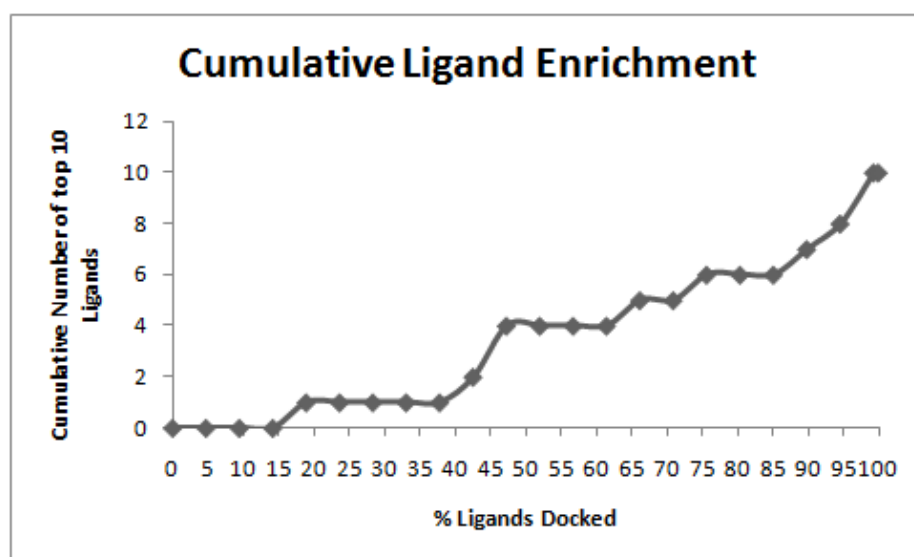


Figure 4.18: Cumulative Ligand Enrichment for top 10 ligands.

about the protein structure or its chemical properties which govern its binding ability with different ligands.

4.5.4 Human Dual Specificity Phosphatase 5 (DUSP5C)

Docking results of DUSP5C show that the average energy for the first random round is -5.97366 . It decreases in round 2 and then attains almost a constant average value for all rounds, with a minor exception to round 7. In terms of average energy, no great improvement is seen after

the initial decrease. Figure 4.19 shows the plot between the average energy of a round and round number for DUSP5C.

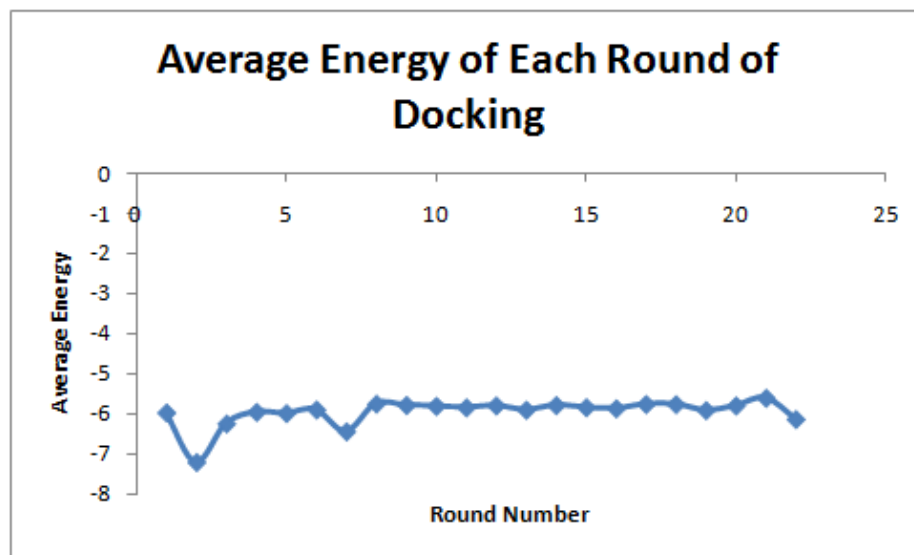


Figure 4.19: Plot between average energy of a round and round number for DUSP5C. Round 2 shows a significant drop in average energy and almost constant average energy for subsequent rounds.

Ligand Enrichment is seen in Figure 4.20, which shows the plot between the number of top 1000 ligands and the percentage ligands docked. It is observed that docking 5% of the ligand library in round 1 gives an expected enrichment of 50 ligands. Round 2 gives a massive enrichment of 270 ligands, which alone accounts for an impressive 27% of the best 1000 ligands and is achieved by docking only 9% of the ligand library. After that, there is little or no significant and all rounds perform poorly as compared to a random selection process.

Cumulative Ligand Enrichment plot in Figure 4.21 depicts the same fact. It can be seen that the slope of the curve changes very rapidly between rounds 1 and 2, indicating a high rate of ligand enrichment. After that it attains a gradual slope and tends to get closer to the constant enrichment line (red). The curve doesn't flatten out but the change in slope is not too high either, suggesting gradual change in further enrichment.

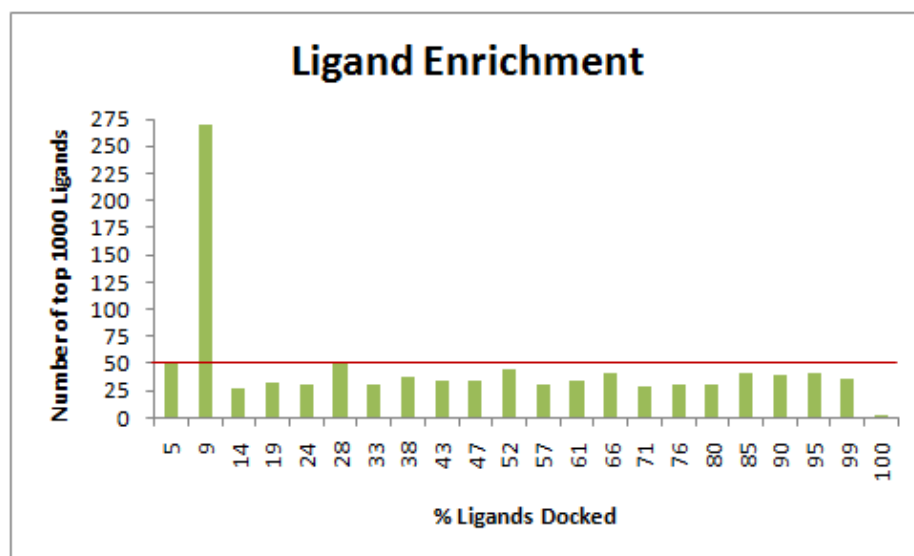


Figure 4.20: Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.

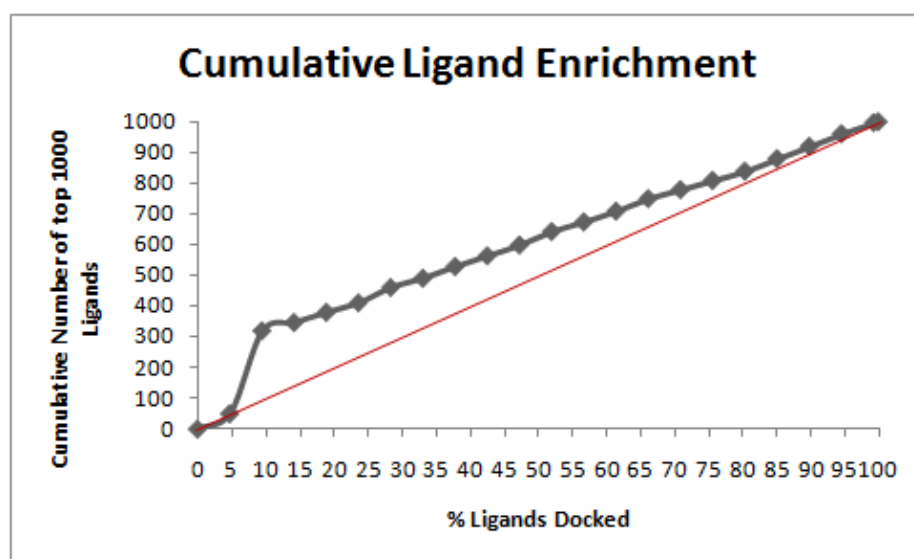


Figure 4.21: The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red) till 75% of the ligands are docked.

Next, the test statistic for DUSP5C is computed. Let G be the total number of best ligands for testing, G_i be the number of best ligands in round i , N be the total number of ligands and N_i be the total number ligands docked in round i , then

$$G = 1000$$

$$G_1 = 50$$

$$G_2 = 270$$

$$G_3 = 27$$

$$N = 10573$$

$$N_1 = 500$$

$$N_{(2+3)} = 1000$$

$$\begin{aligned}\pi_0 &= \frac{G - G_1}{N - N_1} \\ &= \frac{1000 - 50}{10573 - 10073}\end{aligned}$$

$$= 0.094$$

$$\begin{aligned}\hat{\pi} &= \frac{G_2 + G_3}{N_{2+3}} \\ &= \frac{270 + 27}{1000}\end{aligned}$$

$$= 0.297$$

$$\begin{aligned}\sigma_{\hat{\pi}} &= \sqrt{\frac{\hat{\pi}(1 - \hat{\pi})}{N_{(2+3)}}} \\ &= \sqrt{\frac{0.094(1 - 0.094)}{1000}}\end{aligned}$$

$$= 0.0092$$

$$\begin{aligned}z &= \frac{\hat{\pi} - \pi_0}{\sigma_{\hat{\pi}}} \\ &= \frac{0.297 - 0.094}{0.0092}\end{aligned}$$

$$= 22.06$$

The value of test-statistic $z = 22.06$, which is greater than 3.71. Hence, we can reject the null hypothesis at 0.001 level and say that the scoring function in Incrdock does a highly significant selection of top ligands in rounds 2 and 3 for DUSP5C.

Figure 4.22 shows the predicted docking complex of DUSP5C with the best ligand in the second round of docking. It is observed that the ligand (represented by stick model) many side chain interactions with the protein molecule (represented by green). It is seen from the picture that the ligand molecule is very strategically placed in a binding cavity in the protein. It spans between the entire 3D space available for intermolecular interactions.

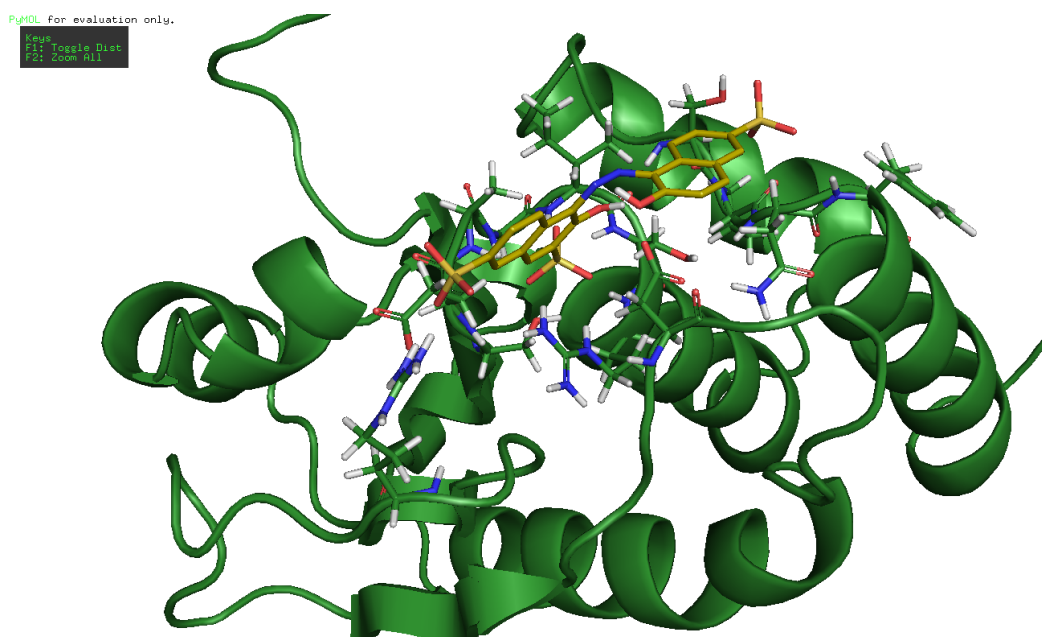


Figure 4.22: Docking Complex of DUSP5C with the ligand 270 in round 2.

Looking at the results for DUSP5C, it is suggested that the docking can be stopped right after round 1. It is observed that 27% of good ligands are found by docking only 9% of the entire ligand library. The rest of the good ligands are more or less constantly distributed across other rounds. Another observation is that the average energy of most of the rounds is similar with a significant exception to round 1.

Finally, Ligand Enrichment (Figure 4.23) and Cumulative Ligand Enrichment (Figure 4.24) for top 10 (0.01%) ligands was evaluated.

It is observed that 9 out of 10 top ligands are docked by exploring only 9% of the ligand library and all 10 top ligands are docked by exploring 52% of the entire ligand library.

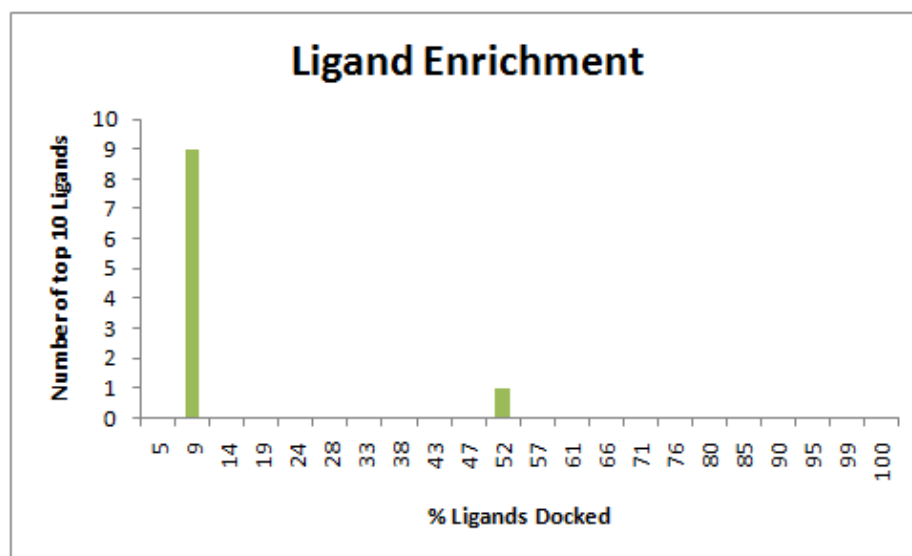


Figure 4.23: Ligand Enrichment with top 10 ligands.

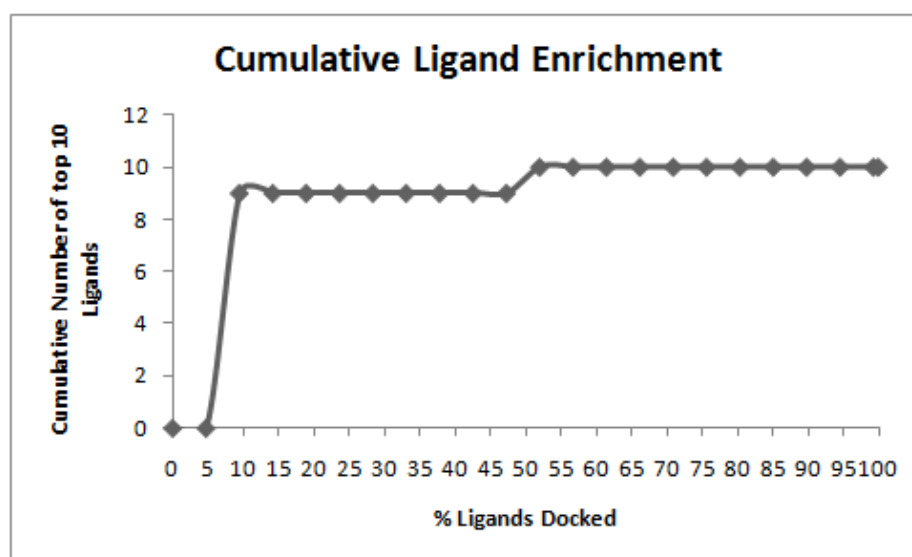


Figure 4.24: Cumulative Ligand Enrichment with top 10 ligands.

4.5.5 Human Dual Specificity Phosphatase 5 (DUSP5R)

The incremental docking results for DUSP5R show that the average energy for the first random round is -7.1805 and it drops significantly in round 2. After round 2, it starts increasing but is still lower than round 1 till round 8. After round 8, it keeps increasing and hits its maximum in round 22. Figure 4.25 shows the plot between the average energy of a round and round number.

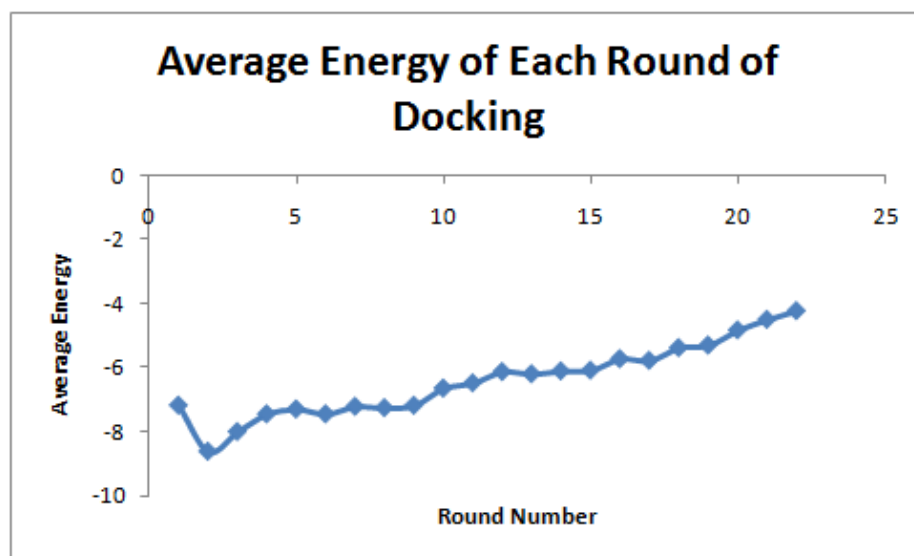


Figure 4.25: Plot between average energy of a round and round number for DUSP5R. Round 2 shows a significant drop in average energy and almost constant change in average energy for subsequent rounds.

Ligand Enrichment is seen in Figure 4.26, which shows the plot between number of top 1000 ligands and percentage ligands docked. It is observed that docking 5% of the ligand library in round 1 gives an enrichment of 40 ligands, which is slightly lesser but close to what the expectation in the random selection process. Round 2 gives a massive enrichment of 240 ligands, and alone accounts for 24% of the best 1000 ligands. Round 3 gives a good enrichment of 127 ligands. Almost 37% of the best 1000 ligands can be extracted in the rounds 2 and 3 by docking only 14% of the ligand library. After that, the enrichment is almost a constant (lesser or close to random) with some enrichment till 90% of the ligands are docked, but after that very insignificant enrichment is seen. There is zero enrichment in the last two rounds.

Cumulative Ligand Enrichment is seen in Figure 4.27, which shows the plot between cumulative number of top 1000 ligands and percentage ligands docked. It can be seen that the slope of the curve is very steep between rounds 1 and 2, and is very high as compared to the random enrichment. The curve almost flattens out after docking about 70%, suggesting little or no change in further enrichment. The plot shows that cumulative enrichment is much better than a constant enrichment, depicted by red line.

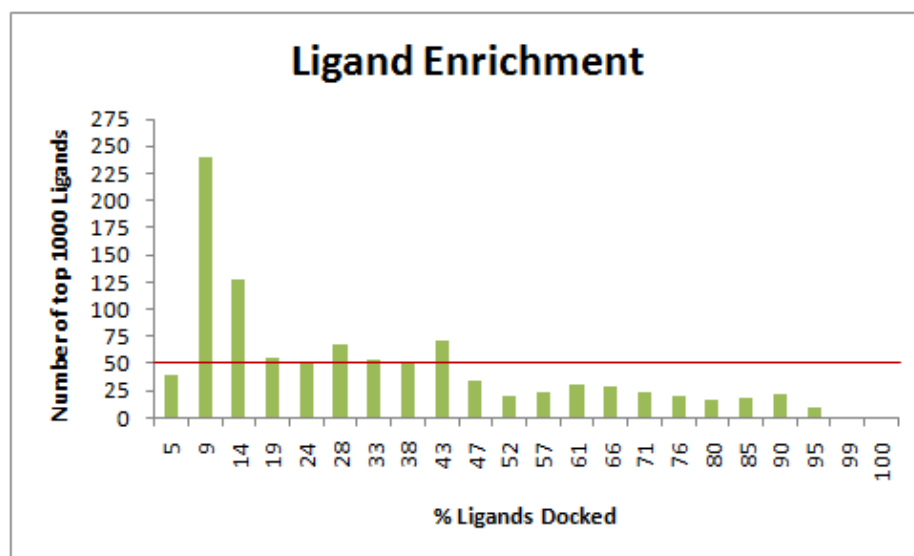


Figure 4.26: Plot of number of top 1000 ligands versus % ligands docked (Green). The horizontal line (Red) depicts the ligand enrichment distribution in case of random selection of ligands for incremental docking.

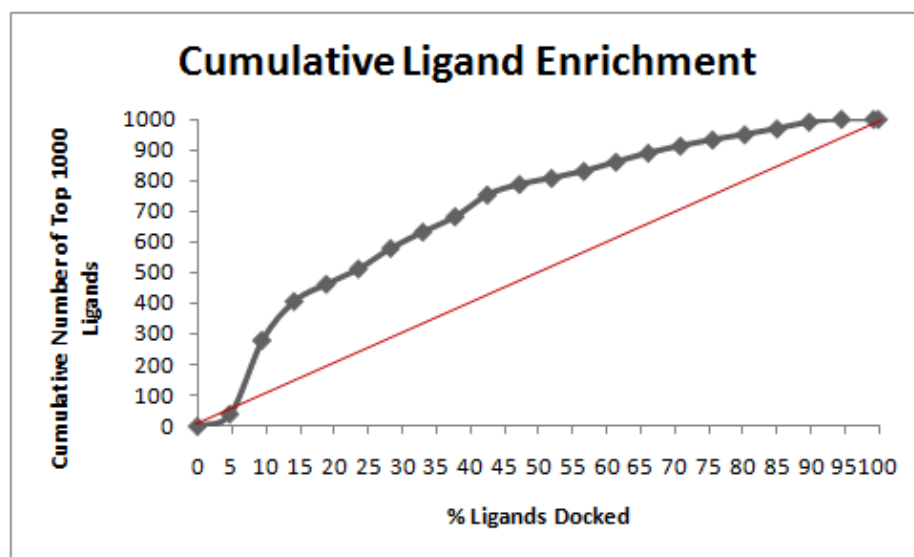


Figure 4.27: The cumulative enrichment of ligands with progressing rounds in incremental approach with Bayesian selection (Grey) is much above the enrichment depicted by a random selection (Red).

Next, the test statistic for DUSP5R is computed. Let G be the total number of best ligands for testing, G_i be the number of best ligands in round i , N be the total number of ligands and N_i be the total number ligands docked in round i , then

$$G = 1000$$

$$G_1 = 40$$

$$G_2 = 240$$

$$G_3 = 127$$

$$N = 10573$$

$$N_1 = 500$$

$$N_{(2+3)} = 1000$$

$$\begin{aligned}\pi_0 &= \frac{G - G_1}{N - N_1} \\ &= \frac{1000 - 40}{10073}\end{aligned}$$

$$= 0.095$$

$$\begin{aligned}\hat{\pi} &= \frac{G_2 + G_3}{N_{2+3}} \\ &= \frac{240 + 127}{1000}\end{aligned}$$

$$= 0.367$$

$$\begin{aligned}\sigma_{\hat{\pi}} &= \sqrt{\frac{\hat{\pi}(1 - \hat{\pi})}{N_{(2+3)}}} \\ &= \sqrt{\frac{0.095(1 - 0.095)}{1000}}\end{aligned}$$

$$= 0.0093$$

$$\begin{aligned}z &= \frac{\hat{\pi} - \pi_0}{\sigma_{\hat{\pi}}} \\ &= \frac{0.367 - 0.095}{0.0093}\end{aligned}$$

$$= 29.25$$

The value of test-statistic $z = 29.25$, which is greater than 3.71. Hence, we can reject the null hypothesis at 0.001 level and say that the scoring function in Incredock does a highly significant selection of top ligands in rounds 2 and 3 for DUSP5R.

Figure 4.28 shows the predicted docking complex of DUSP5R with the best ligand in the second round of docking. It is observed that the ligand (represented by stick model) is partly placed inside the binding pocket of the protein molecule (represented by blue). The picture shows the interactions between the ligand side chains and the relevant protein amino acid side chains.

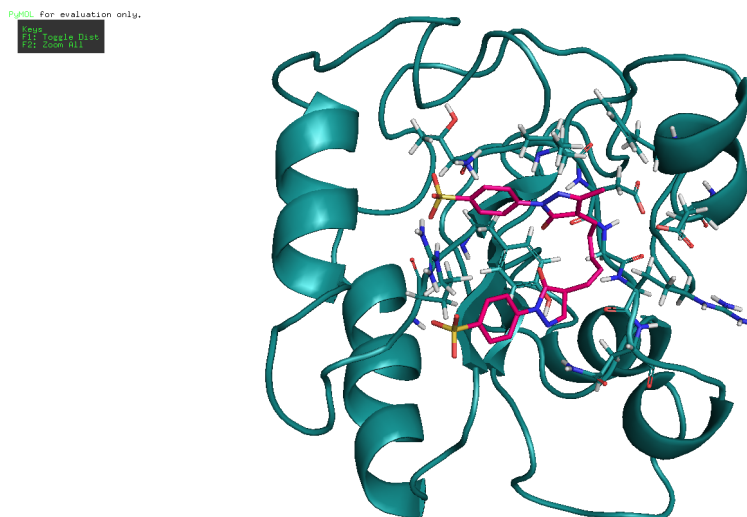


Figure 4.28: Docking Complex of DUSP5R with the ligand 51 in round 2.

Looking at the results of incremental docking for DUSP5R, it is suggested that the dockings can be stopped after almost 15% of the ligands are docked (round 3), by when almost 37% of good ligands are found. In a more conservative approach, stopping after about 50% of the ligand library is docked, also results in almost 75% of top 1000 ligands.

Finally, Ligand Enrichment (Figure 4.29) and Cumulative Ligand Enrichment (Figure 4.30) for top 10 (0.01%) ligands was evaluated. It is observed that 7 top ligands are docked by exploring only 9% of the ligand library and all top 10 ligands are docked by exploring only 14% of the entire ligand library.

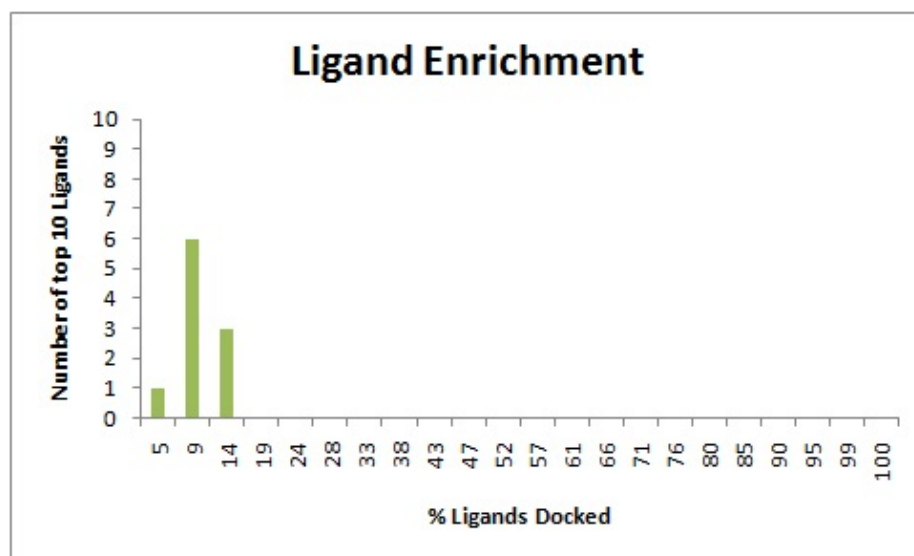


Figure 4.29: Ligand Enrichment with top 10 ligands.

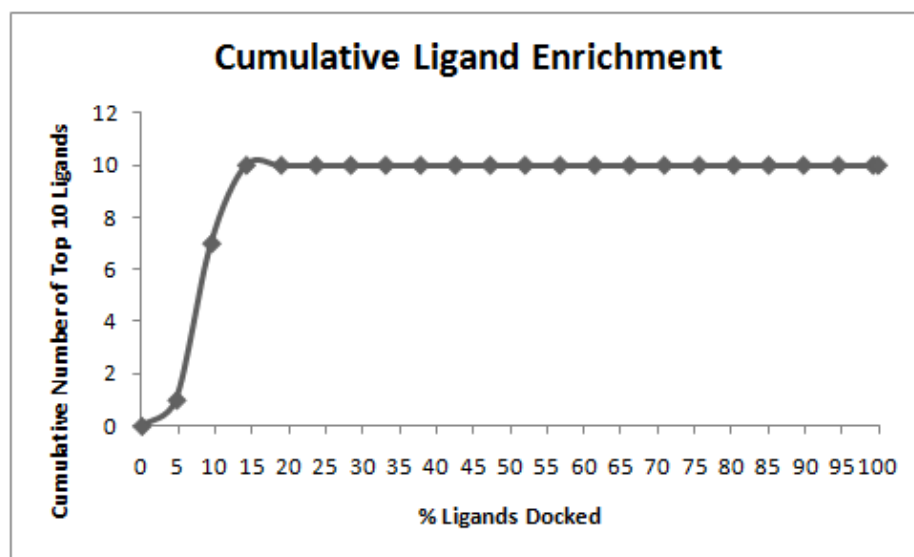


Figure 4.30: Cumulative Ligand Enrichment with top 10 ligands.

4.6 Discussion

As established in Chapter 2, the main motive of this thesis is to develop a workflow which can handle large chemical libraries for drug design experiments. The approach implemented in this thesis should ultimately be able to predict or find ligands which will lower the energy of the docking complex and be more favourable potential drug leads. The experiments conducted using

IncreDock illustrate this fact to a reasonable extent. Looking at the results for the individual proteins in Section 4.5, a generalization can be made about performance of IncreDock. Sections 4.6.1 and 4.6.2 present these observations.

4.6.1 Average Energy of a round

The scoring function in IncreDock predicts the binding energies of protein-ligand complexes. Hence, IncreDock should be able to predict ligands with lower binding energies in the initial runs after docking a small fraction of the ligand library. It is observed from the graphs in Section 4.5 that the average energy of the protein-ligand complex decreased substantially in the very first round of Bayesian selection. Specifically, it is observed from the graphs that:

- For all proteins, the average energy in round 2 is the lowest across all rounds. This energy is slightly lower or comparable to round 3.
- For all proteins, the average energy starts to increase after an initial decrease and attains a value, which is higher than round 1 in very few subsequent rounds.
- Round 22 is the last set of ligands docked and it is expected that the energy of these complexes should be least favored. It is actually seen that the average energy of round 22 is the highest and much greater than in round 1, which is selected randomly.
- An interesting observation is made in the case of TrxR. Even though the first round does have a lower average energy, it is not propagated in the subsequent rounds. The average energy stays almost the same in most of the rounds which indicates that all ligands have similar binding energy to TrxR.

Thus, IncreDock does a selection which lowers the average energy of the docking complexes after the first round of docking. The average value of energy increases after the initial decrease and the last few rounds have the highest average energy values. Rounds 2 and 3 for each protein clearly indicate this fact. IncreDock, indeed, does a good job in predicting ligands with lower binding energies to the protein of interest.

4.6.2 Ligand Enrichment and Cumulative Ligand Enrichment

IncreDock should select better binders (ligands) in earlier rounds by docking a small fraction of the ligand library. This can be recognized by identifying the *good* ligands and determining when can they be found by the IncreDock. It is observed from the enrichment graphs in Section 4.5 that:

- For all proteins, 200 of top 1000 ligands are found by docking only 14% of the ligand library.
- 500 of top 1000 are found by docking 28% of the ligand library for 4 proteins. 500 of top 1000 ligands found by docking 47% of the ligand library for the 5th protein (TrxR).
- For all proteins, there is no significant gain in enrichment after docking almost 70% of the ligand library.
- With an exception to TrxR, the enrichment for top 10 ligands in all proteins looks very good. Top 9 or 10 ligands are found within docking 19% of the ligand library.
- In case of TrxR, it is observed that there is almost no gain in enrichment even though the initial average energy of ligands is decreased.

Thus, IncreDock is consistently able to predict better ligands in Rounds 2 and 3. As expected all the *good* ligands should be found in the first few rounds and the last few rounds show minimal enrichment. IncreDock, indeed, does a good job in predicting ligands which are better binders.

4.6.3 Significance Testing

Looking at the results of significance test for all the proteins it is seen that:

- Top 1000 ligands were docked in rounds 2 and 3 with p-value ≤ 0.01 in 4 experiments.
- Top 1000 ligands were docked in rounds 2 and 3 with p-value ≤ 0.066 in 5th experiment.

These results demonstrate that IncreDock certainly has some degree of selection for better binding ligands. The Bayesian scoring function, implemented in IncreDock, has successfully been able to demonstrate better enrichment as compared to a completely random docking experiment.

The scoring model in InCreDock, however, is not ultimate and there is scope for enhancement. A few simple experiments/evaluations can be done to get better heuristic models. At a very basic level, these may include exploring more proteins, varying the number of compounds docked in the first round to gather more information for subsequent prediction, or changing the algorithm of finding the energy of a ligand. More of these possibilities are discussed in Chapter 5.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, an optimization to the virtual screening workflow has been proposed. The main motive is to develop a software system, which can handle large chemical libraries available for docking experiments. A software tool, IncreDock, is developed to implement automated parallel protein dockings in increments utilizing a high performance computing cluster. It, thus, enables large throughput of results in smaller time scale. The tool developed, IncreDock, provides two ways of launching a large-scale docking experiment. The first one is a basic approach where all the ligands available are queued in for individual dockings in parallel. After all the dockings are finished, the results are aggregated and presented to the user. The other approach is an Incremental approach, which utilizes an X-DAG structure and forms the main contribution of the thesis. Here, a fraction of ligands are queued for individual dockings in parallel. After this fraction of ligands is docked, the results are aggregated and analyzed for predicting the next fraction of ligands, using a Bayesian scoring function. The idea is to predict the docking energies of un-docked ligands by comparing them to the docked ones, on the basis of their chemical properties (specifically the ones stated in the Lipinski's rule-of-five). Section 5.1 presents the conclusion and contribution of IncreDock. Section 5.2 proposes the future research options in this direction.

5.1 Conclusion

IncreDock was tested against four different proteins Dihydrofolate Reductase, Dihydrodipicolinate reductase, Thioredoxin Reductase (all three relevant to Tuberculosis) and Human Dual Specificity Phosphatase 5 (relevant to Tumor growth). All these proteins were docked against a ligand library of 10,573 compounds using Bayesian scoring function of IncreDock. The results show that better binding ligands can be identified in early increments of dockings.

Currently, there exist software like Docking@Home, DOVIS and DockFlow, which allow for the parallelization of the entire docking process. However, none of them provide a selective mechanism to identify better ligands without docking the entire chemical library [44, 45, 46]. IncreDock has naively been able to address the problem of incremental docking and has provided

an effective initial strategy to sample out good ligands in less time and computation as compared to random.

Another, advantage of IncreDock is that it can be ported and integrated with other available grids like BOINC and TeraGrid [47, 48]. The basic approach of IncreDock was successfully tested on TeraGrid using the CPFM ligand library of 10,573 compounds.

IncreDock can also be implemented on a commercial cloud. IncreDock was migrated to the cloud to assess the feasibility of such an implementation. The basic approach was tested on a commercially available cloud platform, Amazon EC2, utilizing Dihydrofolate Reductase (DHFR) and a small test-portion of the CPFM ligand library. A detailed performance analysis and comparison was also done to validate against local high performance computing resources. It was found the IncreDock can be implemented on the cloud as there is no overhead required to set up an in-house cluster or grid, software requirements are inexpensive or free, and computing time is rapid based on the number of resources purchased on the cloud.

The performance of IncreDock at the level of ZINC is still to be tested. It remains to be seen that enrichment and scaling obtained in this study can be maintained at the size of ZINC. Nonetheless, these early results are promising and further investigations to see if this approach scales appropriately are needed.

5.2 Future Work

Based on the results and the approach taken, there are possibilities for further refinements and enhancements to the software. To mention a few, I suggest the following future considerations in the approach:

1. In the current implementation, the energy of the ligand is defined based on the cluster size and the minimum energy of a pose in that cluster. A variation to this could be assigning weights to all the clusters and choosing a weighted average energy for each ligand.
2. The prediction algorithm uses the average energy of the previous round. In the current implementation, this is calculated by a simple average. A variation to this could be

determining a weighted average which may actually suggest the energy distribution in that round.

3. In the Bayesian Scoring, the energies and properties of ligands are binned for calculations of predicted energy bins. There can be some experimentation with the number and size of bins to observe any improvements in the current Bayesing scoring.
4. The scoring function gets some data from the random training set (ligands docked in the first round). The size of this training set can be varied to observe any improvements.
5. The way the training set is generated (random) can be replaced with a more diverse data set by selecting ligands in a different way. One possibility could be binning the compounds based on their properties and picking representatives from each property bin.
6. More scoring functions can be implemented in addition to Bayesian Scoring.
7. The experiments can be conducted on more number of proteins to expand the scope and variety of proteins studied.
8. The experiments can be scaled by porting the code on other grids.
9. The docking parameters specified at the time of ligand and protein preparation can be varied to observe any significant changes.
10. In the current implementation the docking software of choice is AutoDock. IncreDock can be made flexible to incorporate other docking software.

To conclude there a lot of possible enhancements that can be made to the IncreDock and more promising results can be expected in the future.

BIBLIOGRAPHY

- [1] B. D. Anson, J. Ma, and J.-Q. He, "Identifying cardiotoxic compounds," *Genetic Engineering & Biotechnology News*, vol. 29, pp. 34–35, 2009.
- [2] S. M. Paul, D. S. Mytelka, C. T. Dunwiddie, C. C. Persinger, B. H. Munos, S. R. Lindborg, and A. L. Schacht, "How to improve r&d productivity: the pharmaceutical industry's grand challenge," *Nature Reviews Drug Discovery*, vol. 9, pp. 203–214, 2010.
- [3] B. Waszkowycz, T. D. J. Perkins, R. A. Sykes, and J. Li, "Large-scale virtual screening for discovering leads in the postgenomic era," *IBM Systems Journal - Deep computing for the life sciences*, vol. 40, pp. 360 – 376, 2001.
- [4] J. Alvarez and B. Shoichet, *Virtual Screening in Drug Discovery*. CRC Press, 2005.
- [5] "Chemical proteomics facility at marquette (cpfm)."
<http://www.marquette.edu/cpfm/resources.shtml>.
- [6] "Zinc homepage." <http://zinc.docking.org/>.
- [7] "Drug discovery." http://en.wikipedia.org/wiki/Drug_discovery.
- [8] "Combinatorial chemistry review." <http://www.combichemistry.com/drug-discovery.html>.
- [9] W. Walters, M. Stahl, and M. Murcko, "Virtual screening-an overview," *Drug Discovery Today*, vol. 3, pp. 160–178, 1998.
- [10] A. S. Reddy, S. P. Pati, P. P. Kumar, H. Pradeep, and G. N. Sastry, "Virtual screening in drug discovery a computational perspective," *Current Protein & Peptide Science*, vol. 8, pp. 329–351, 2007.
- [11] I. Muegge and S. Oloff, "Advances in virtual screening.," *Drug Discovery Today: Technologies*, vol. 3, p. 405411, 2006.
- [12] B. K. Shoichet, "Virtual screening of chemical libraries.," *Nature*, vol. 432, p. 862865, 2004.
- [13] "Docking (molecular)." [http://en.wikipedia.org/wiki/Docking_\(molecular\)](http://en.wikipedia.org/wiki/Docking_(molecular)).

- [14] L. AR, S. BK, and P. CE., "Prediction of protein-ligand interactions. docking and scoring: successes and gaps.," *Journal of Medicinal Chemistry*, vol. 49, pp. 5851–5, 2006.
- [15] R. Taylor, P. Jewsbury, and J. Essex, "A review of protein-small molecule docking methods.," *Journal of Computer-Aided Molecular Design*, vol. 16, pp. 151–166, 2002.
- [16] S. SF, F. PA, and R. MJ., "Protein-ligand docking: current status and future challenges.," *Proteins*, vol. 65, pp. 15–26, 2006.
- [17] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin, "A geometric approach to macromolecule-ligand interactions," *Journal of Molecular Biology*, vol. 161, pp. 269–288, 1982.
- [18] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function.," *Journal of Computational Chemistry*, vol. 19, pp. 1639–1662, 1998.
- [19] "Autodock homepage." <http://autodock.scripps.edu/>.
- [20] G. Jones, P. Willett, and R. C. Glen, "Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation.," *Journal of Molecular Biology*, vol. 245, pp. 43–53, 1995.
- [21] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe, "A fast flexible docking method using an incremental construction algorithm.," *Journal of Molecular Biology*, vol. 261, pp. 470–489, 1996.
- [22] C. Lipinski, F. Lombardo, B. Dominy, and P. Feeney, "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings.," *Advanced Drug Delivery Reviews*, vol. 23, p. 325, 1997.
- [23] "Lipinski's rule." http://www.bioscreening.com/reference/lipinski_rule.htm.
- [24] "Druglikeness." <http://en.wikipedia.org/wiki/Druglikeness>.
- [25] "Condor homepage." <http://www.cs.wisc.edu/condor/>.

- [26] *Condor - A Hunter of Idle Workstations.*, 1988.
- [27] “Dagman applications.”
http://www.cs.wisc.edu/condor/manual/v7.4/2_10DAGMan_Applications.html.
- [28] “Condor dagman: Handling inter-job dependencies..” <http://cs.wisc.edu/condor/dagman/>.
- [29] “X-dag.” http://www.cs.wisc.edu/condor/manual/v7.4/2_10DAGMan_Applications.html.
- [30] “Autodocktools.” <http://autodock.scripps.edu/resources/adt>.
- [31] “Tuberculosis.” <http://en.wikipedia.org/wiki/Tuberculosis>.
- [32] A. Argyrou, M. W. Vetting, B. Aladegbami, and J. S. Blanchard., “Mycobacterium tuberculosis dihydrofolate reductase is a target for isoniazid,” *Nature Structural & Molecular Biology*, vol. 13, pp. 408 – 413, 2006.
- [33] M. Cirilli, R. Zheng, G. Scapin, and J. S. Blanchard, “The three-dimensional structures of the mycobacterium tuberculosis dihydrodipicolinate reductase nadh2,6-pdc and nadph2,6-pdc complexes. structural and mutagenic analysis of relaxed nucleotide specificity.,” *Biochemistry*, vol. 42, p. 1064410650, 2003.
- [34] B. K, G. S, S. RH, and M. S., “Thioredoxin reductase as a pathophysiological factor and drug target.,” *European Journal of Biochemistry.*, vol. 267, pp. 6118–25, 2000.
- [35] L. R, S. R, C. P, S. W, W. J, A. F, T. S, and H. WG., “Three-dimensional structure of m. tuberculosis dihydrofolate reductase reveals opportunities for the design of novel tuberculosis drugs.,” *Journal of Molecular Biology*, vol. 295, pp. 307–23, 2000.
- [36] A. M, S. K, V. C, and M. SC, “Conformational flexibility of mycobacterium tuberculosis thioredoxin reductase: crystal structure and normal-mode analysis.,” *Acta Crystallogr D Biol Crystallogr.*, vol. 61, pp. 1603–11, 2005.
- [37] “Dual specificity protein phosphatase 5.” <http://en.wikipedia.org/wiki/DUSP5>.
- [38] K. SP and D. JE., “Multiple dual specificity protein tyrosine phosphatases are expressed and regulated differentially in liver cell lines.,” *Journal of Biological Chemistry*, vol. 270, pp. 1156–60, 1995.

- [39] K. L. Jeffrey, M. Camps, C. Rommel, and C. R. Mackay, “Targeting dual-specificity phosphatases: manipulating map kinase signalling and immune responses.,” *Nature Reviews Drug Discovery*, vol. 6, pp. 391–403, 2007.
- [40] R. L. Ott and M. Longnecker, *Statistical Methods and Data Analysis*. 2001.
- [41] “Pymol homepage.” <http://www.pymol.org/>.
- [42] “Mgltools website.” <http://mgltools.scripps.edu/>.
- [43] “Python website.” <http://www.python.org/download/releases/2.4.3/>.
- [44] S. Zhang, K. Kumar, X. Jiang, A. Wallqvist, and J. Reifman, “Dovis: an implementation for high-throughput virtual screening using autodock.,” *BMC Bioinformatics*, vol. 9, p. 126, 2008.
- [45] “Docking@home.” <http://docking.cis.udel.edu/about/project/>.
- [46] N. Azam, M. Ghanem, D. Kalaitzopoulos, A. Wolf, V. Kasam, Y. Wang, and M. Hofmann-Apitius., “Dockflow: Achieving interoperability of protein docking tools across heterogeneous grid middleware,” *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 6, pp. 235 – 251, 2010.
- [47] “Boinc homepage.” <http://boinc.berkeley.edu/>.
- [48] “Teragrid homepage.” <https://www.teragrid.org/>.

APPENDIX A STEPS FOR PREPARATION OF INPUT FILES FOR DOCKINGS

A.1 Steps for preparing the Ligand Molecule

1. Load the ligand into the viewer (Ligand > Input > Open Ligand.pdb).
2. Determine which atom fits its idea of the best root (Ligand > Torsion Tree > Detect Root).
3. Determine which rotatable bonds to be active (Ligand > Torsion Tree > Choose Torsions).
4. Set the numbers of bonds to be active (Ligand > Torsion Tree > Set Number of Torsions).
5. Save the prepared Ligand (Ligand > Output > Save as Ligand.pdbqt).
6. Remove from viewer (Edit > Delete Molecule).

A.2 Steps for preparing the Protein Molecule

1. Edit Protein
 - (a) Open the Protein structure file (File > Read Molecule .pdb).
 - (b) Color atoms by chemical element (Color > By Atom Type > All Geometries ; OK).
 - (c) Add Polar Hydrogen Atoms, choose default (Edit > Hydrogens > Add > Select All Hydrogens, no Bond Order, yes > OK).
 - (d) Save the Macromolecule (File > Save > Write PDB).
 - (e) Delete all molecules.
2. Prepare Macromolecule File
 - (a) Add the charges and solvation parameters, choose defaults (Grid > Macromolecule > Open .pdb file saved in saved in 1.e).
 - (b) Save the file (Save as .pdbqt file).
3. Prepare the Grid Parameter File
 - (a) Specify the type of Maps used, Add A, Br, C, Cl, F, H, HD, I, N, NA, OA, P, SA, S (Grid > Set Map Types > Open Ligand).
 - (b) Select the grid for Docking (Grid > Grid Box > Choose to center on ligand or atom, Make the grid box bigger to encapsulate the entire ligand > Delete Molecule > Close saving current).
 - (c) Save the Grid Parameter File (Grid > Output > Save GPF).
 - (d) Start Autogrid4 in the directory where .gpf file is located (autogrid4 p grid.gpf l grid.glg &).
 - (e) Delete Molecule from ADT.
4. Prepare Docking Parameter File (dpf): This step is to be done for all the ligands in the ligand library. In this work, this step is scripted to prepare the dpf file for all ligands at once.

- (a) Open the edited Macromolecule (Docking > Macromolecule > Set Rigid Filename).
- (b) Select the current Ligand (Docking > Ligand > Choose Ligand).
- (c) Select Docking Algorithm, choose 50-100 GA Runs (Docking > Search Parameters > Genetic Algorithm).
- (d) Select Default docking parameters (Docking > Docking Parameters).
- (e) Choose the Algorithm (Docking > Output > Lamarckian GA).
- (f) Save as .dpf

A.3 Combining the input data to create the input tar file

1. Copy all the prepared ligands and the protein in one directory.
2. Prepare the dpf file for all the ligands by scripting step 4 in protein preparation.
3. Copy all the map files to be used for docking.
4. Tar the directory.
5. Repeat these steps for all the proteins used.

APPENDIX B

BAYESIAN SCORING FUNCTION

```

#Author — Prachi Pradeep
#!/usr/bin/python
"""
This program generates a list of ligands which go into the next round of docking
experiment in the X-Dag based on bayesian selection.
"""

import random
import sys
import csv
import numpy
import operator
import os

# Find the docking round number, if it is 0 (round 1) then generated a random
  set of ligands to dock.

node=sys.argv[1]
N=500

if int(node)==0:
    infile="ligand_%s.txt" % int(node)
    outfile="ligand_%s.txt" % (int(node)+1)
    dockfile="dock_ligand_%s.txt" % int(node)
    file=open(infile,"rw")
    lines=file.readlines()
    num_lines=len(lines)
    a=range(num_lines)

    file1=open(dockfile,'w')
    if N < num_lines:
        n=0
        dock=random.sample(a,N)
        for i in dock:
            print >> file1, "DOCK_%s_%s, %s" % (int(node),n,lines[i]),
            n=n+1

        map(a.remove,dock)
        file2=open(outfile,'w')

        for i in a:
            print >> file2, "%s" % lines[i],

    else:
        n=0
        for line in lines:
            print >> file1, "DOCK_%s_%s, %s" % (int(node),n,line),
            n=n+1

# Else if the round number is 1 or more, generate the ligands to be docked by
  predicting the docking energies of ligands by Bayesian selection over
  compound properties

```

```

elif int(node) > 0:
    infile="ligand_%s.txt" % (int(node))
    outfile="ligand_%s.txt" % (int(node)+1)
    dockin="best_summary.txt"
    dockout="dock_ligand_%s.txt" % (int(node))
    property="properties_binned"

    file3=open(outfile,"w")
    file4=open(dockout,"w")

    # Save the ligand properties file in a dictionary indexed by the ligand name

    properties={}
    file_h=open(property,'r').readlines()
    for line in file_h:
        if line.find('ligand')== -1:
            properties[line.split(',')[0]]=line.strip('\n').split(',')

    headers=file_h[0].split(',')
    lh=len(headers)

    # Find the number of properties n

    n=int(lh/2)

    # Save the infile in a dictionary indexed by the ligand name of ligands not
    # docked yet

    file_h = (open(infile,'r').readlines())

    # Define and create energy bins for all ligands docked so far

    no_of_bins=5
    energy_all=numpy.loadtxt(dockin, delimiter=",", skiprows=1, usecols=(2,))
    hist_energy, bins=numpy.histogram(energy_all, no_of_bins)
    len_h=len(hist_energy)

    # Find the probability of each energy bin

    prob_energy=[]
    for i in range(no_of_bins):
        prob=(float(hist_energy[i])+float(0.1))/(float(sum(hist_energy))+float(
            no_of_bins*0.1))
        prob_energy.append(prob)

    # define keys for docked ligands in order of docking of ligands already docked

    ligands_d=[]
    file_h=open(dockin,'r').readlines()
    for line in file_h:
        if (line.find('dock_id')== -1):
            ligands_d.append(line.split(',')[1].strip())

    # Save the information of ligands docked with their energies and energy bins

    docked={}
    file_h=open(dockin,'r').readlines()
    d=numpy.digitize(energy_all, bins) #array of bins for all the energy values

```

```

for i,line in enumerate(file_h):
    if i!=0:
        dock_line=line.strip('\n').strip(',').split(',')
        if d[i-1]==no_of_bins+1:
            dock_line.append(d[i-1]-2)
        else:
            dock_line.append(d[i-1]-1)
        docked[line.split(',')[1].strip()]=dock_line

len_cd=len(docked)

# Function to return the energy bin of a ligand from a dictionary of compounds
# docked

def return_bin(compound):
    bin=docked[compound][4]
    return bin

# Function to return the probability of a property in each energy bin

def property_prob(len,energy_bin,compound_prop,e,len_h,i,n):
    count=0
    for k in ligands_d:
        e_bin=return_bin(k)
        if int(e_bin)==int(energy_bin):
            d_p=properties[k][n+i] #property for docked
            c_p=compound_prop[n+i] #property for compound
            if (int(d_p)==int(c_p)):
                count=count+1
            property_prob=(float(count)+0.1)/(float(e)+len_h*0.1)
    return property_prob

# Calculate the probability of each compound
# keys for properties of ligands in order of infile of ligands not docked yet

prob=[]
file_h = (open(infile,'r').readlines())
new_ligands={}
ligands_n=[]
for line in file_h: #for each new compound
    if (line.find('l_pdb')==1):
        if (line.find('L_NDP')==1):
            if (line.find('L_MTX')==1):
                new_ligands[line.split(',')[0].strip()]=line.strip('\n')
                ligands_n.append(line.split(',')[0].strip())
            else:
                print >> file3, "%s" %line.strip('\n')
        else:
            print >> file3, "%s" %line.strip('\n')
    else:
        print >> file3, "%s" %line.strip('\n')

# Calculate the probability of falling in each energy bin for each new ligand

for ligand in ligands_n:
    ligand_prop=properties[ligand]
    maxX=float('-inf')
    for energy_bin in range(no_of_bins): #for each energy bin

```

```

    ligand_prob=1
    energy_prob=prob_energy[energy_bin]
    e_count=hist_energy[energy_bin]
    for i in range(1,n+1): #for each property of the compound and in the bin
        ligand_prob=ligand_prob*float(property_prob(len_cd,energy_bin,
            ligand_prop,e_count,len_h,i,n))
    x=float(ligand_prob)*float(energy_prob)
    if x>maxX:
        bestE=energy_bin
        maxX=x
    prob.append((ligand,bestE)) #Compound and its most likely energy bin

# Sort the ligands on ascending order of the predicted binding energy bins.

    prob_sorted=sorted(prob, key=operator.itemgetter(1))
    size=len(prob_sorted)

# Select the ligands to be docked in the next round of dockings.

N=500
if size <= N:
    file_h=open(infile,'r').readlines()
    n=0
    for line in file_h:
        print >> file4, "DOCK%s_%s, %s" %(int(node),n,line.strip('\n'))
        n=n+1

if size > N:
    for i in range(N):
        out=new_ligands[prob_sorted[i][0]]
        print >> file4, "DOCK%s_%s, %s" %(int(node),i,out)
    for i in range(N,len(prob_sorted)):
        out=new_ligands[prob_sorted[i][0]]
        print >> file3, "%s" %out

```