CAEPIDR:
A COMPUTATIONAL APPROACH TO EFFICIENT PEPTIDE INFLUENCED DRUG REPURPOSING

by

Thomas Francis Long

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2014

© 2014
Thomas Francis Long
ALL RIGHTS RESERVED

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Thomas Francis Long

Thesis Title: CAEPIDR: A Computational Approach to Efficient Peptide Influenced Drug Repurposing

Date of Final Oral Examination: 31 October 2014

The following individuals read and discussed the thesis submitted by student Thomas Francis Long, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Timothy Andersen, Ph.D.	Chair of the Supervisory Committee
Owen McDougal, Ph.D.	Member, Supervisory Committee
Amit Jain, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Timothy Andersen, Ph.D., Chair of the Supervisory Committee. The thesis was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

ACKNOWLEDGMENTS

The author wishes to express gratitude to the professors who have helped him during his time at Boise State University. Most notably, Dr. Timothy Andersen, Dr. Amit Jain, and Dr. Owen McDougal. He would also like to thank his companion Kat Dunlevy, whose brilliant mind and dazzling smile have been of great assistance throughout his graduate career.

ABSTRACT

Since the discovery of the molecular basis of disease, numerous studies have reported a correlation between the activity of specific protein receptors and the progression of disease. As a result, drug development has become dependent on the study of protein receptor activities. The relative inexpense of computing hardware has made computational methods an important supplementary tool for receptor modeling. This work details an open source software tool that is capable of both efficiently screening large peptide mutant libraries and enabling 3-D conformer-based searches over local molecular databases.

A Computational Approach to Efficient Peptide Influenced Drug Repurposing (CAEPIDR) has been developed to explore the conformational ligand binding space of the $\alpha 3\beta 2$ nicotinic acetylcholine receptor (nAChR) isoform using a library of α-conotoxin (α-CTx) MII peptide mutants. The screen's top hits were used to identify small molecule drugs that might also bind to the receptor. The conformational ligand binding space of the nAChR was heuristically explored using a genetic algorithm, which managed a structure-based virtual screen of a 640,000 α-CTx MII peptide mutant library. A utility was developed to search the PubChem Compound database for small molecule drugs with a 3-D shape similar to the highest affinity peptides from the virtual screen.

CAEPIDR's genetic algorithm-based procedure was able to find 10 peptides with estimated free energies of binding (with the $\alpha 3\beta 2$-nAChR) below -20 kcal/mol, which

can be compared to α-CTx MII's -12.38 kcal/mol. These peptides were identified in spite of the genetic algorithm performing docking calculations for only 9,344 of the 640,000 α-CTx MII mutants. The PubChem Compound search yielded 2 small molecule drugs with estimated binding energies below -20 kcal/mol.

CAEPIDR has been integrated with DockoMatic to create DockoMatic 2.1, which can be used to create virtual peptide mutant libraries, virtually dock ligands to macromolecular receptors, and identify small molecule drugs for disease treatment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CID – Compound ID

HTS – High Throughput Screening

HTVS – High Throughput Virtual Screening

NIH – National Institute of Health

GA – Genetic Algorithm

α-CTx – α-conotoxin

nAChR – nicotinic acetycholine receptor

pdb – protein databank

sdf – structure data format

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 Drug Repurposing

The discovery of the molecular basis of disease has resulted in massive efforts to research and develop new medicines. So far, de novo drug development has followed the general pattern: lead candidate identification, prototype development, pre-clinical testing of the model systems, clinical trials, Food and Drug Administration (FDA) approval, clinical deployment, and continued monitoring of efficacy and any side effects associated with long-term use. This process, which has historically yielded a marketable drug for every 10 candidates that make it to clinical trials, has an average cost of $\approx$ \$1.2 billion per marketable drug and an average turn around time of 14 years [1, 2]. Reducing these costs is difficult since the majority of the expenses are a result of the clinical trials and cannot be circumvented due to federal safety regulations.

In an effort to reduce development costs, the National Institute of Health (NIH), universities, and pharmaceutical companies have made use of an alternative strategy for drug development: drug repurposing. As the name implies, drug repurposing focuses on the reuse of drugs or small molecules that have already undergone some level

of clinical testing. Repurposing allows researchers to build upon previous research and development efforts to expedite and reduce the costs of clinical trials, speeding the drug's review by the FDA and, if approved, integration into health care.

Drug repurposing is either receptor or ligand focused. Receptor-oriented repurposing is an attempt to identify drugs that can be reused to target another receptor or family of receptors (e.g., acetylcholine receptors). Drug databases are searched for small molecules that exhibit a set of characteristics (e.g., structural similarity to a known ligand or to a receptor's binding cavity) that makes them likely candidates for binding to the targeted receptor. Many chemical databases have been created to support this type of repurposing.

In contrast, ligand-oriented repurposing focuses on finding a new use for a specific drug. It is a popular approach for pharmaceutical companies since it has the potential to yield a new pharmaceutical without the majority of the expenses associated with drug development. Ligand-oriented repurposing is better known to the general public because of the success of repurposed drugs such as Viagra, Requip, and Colesevelam.

The drug repurposing paradigm has had immense success in recent years, accounting for nearly 30% of the newly (FDA) approved drugs between 1999 and 2008 [3]. This success has been fueled in part by resources, mostly in the form of large chemical databases, that have been made publicly available by both public and private institutions. NIH's PubChem database [4] is a notable example as it contains structural and bioactivity information for over 51 million small molecules. It also includes web-based tools for performing substructure, shape, and many other searches over any of their databases.

1.1.2 Molecular Docking

Ligand-gated ion channels are a class of membrane-bound receptor proteins that are often involved in cell signalling. These channels are activated or inhibited by ligands: the peptides and other small molecules that bind to the gate to form a complex with the receptor. The specialized role played by these receptors in the regulation of cellular activity has made them a central focus in the fields of biochemistry and pharmacology, where the goal is often to develop small molecule ligands to activate or inhibit biologically relevant receptors.

Many diseases have been correlated with abnormalities in specific cellular activities. For this reason, drug development is largely a search for small molecules that can activate or inhibit the receptor associated with the abnormal activity. As such, predicting the interaction of a small molecule and biological receptor is a central problem in biochemistry and pharmacology. While fast mathematical methods exist for predicting the interaction strength between two bound molecules of a given orientation, accurately predicting the binding orientation of the two molecules (molecular docking) is a complex procedure.

There are two common approaches to molecular docking: simulation and complementary surface modelling. In the complementary surface approach, both the surface area of the receptor's complex and the surface area of the ligand are represented as a set of descriptors, which are then compared to predict the binding orientation of the ligand. This assessment is relatively fast but limited in its ability to account for molecular flexibility. Complementary surface modelling lends itself to functional group-based comparisons, where the ligand's functional groups can serve as its descriptors for the model.

In comparison to the surface modelling approach, simulation is more accurate but has greater computational complexity. In simulations, molecular dynamics techniques are used in tandem with heuristic search methods to explore the orientation binding space for a ligand/receptor pair. All pairwise atomic interactions between the two molecules must be taken into account, subject to the additional degrees of freedom introduced by ligand and receptor flexibility. These degrees of freedom produce a huge orientation space, and simulation techniques often use a search heuristic, grid system, and scoring function to identify low energy[1] orientations.

Within the context of this thesis, the term molecular docking (or just docking) refers to the use of a simulation technique to determine a ligand/receptor pair's binding orientation and the application of a scoring function to assess the pair's binding affinity.

AutoDock

While molecular docking programs abound, AutoDock [5, 6] is the most popular amongst researchers and was cited in 2006 almost as often as the 4 next most cited docking programs combined [7]. AutoDock is a suite of automated docking tools that simplify the task of molecular docking. It consists of two main programs: *autogrid* for calculating coordinate grids to model the target receptor and *autodock* for simulating molecular docking within those grids. AutoDock is currently maintained by the Scripps Research Institute and is free under the GNU General Public License.

AutoDock uses a Lamarckian Genetic Algorithm to search for a ligand/receptor pair's lowest energy pose. A genetic algorithm (GA) performs an indeterministic search, so multiple GA runs will produce different affinity scores for the same lig-

[1] According to the Gibbs energy function

and/receptor pair. As such, researchers who use AutoDock will typically use multiple (100 is standard) GA simulations with different random seeds to more accurately assess a ligand's binding affinity. Since each simulation results in a predicted orientation (pose), groups of AutoDock simulations are also termed pose evaluations.

1.1.3 High Throughput Virtual Screening

Drug development typically involves screening large collections of compounds to identify a small set of compounds that can serve as the starting point for structure optimization and refinement. In high throughput screening (HTS), machine automation is leveraged to rapidly estimate the biochemical activity of large collections of drug-like molecules. A collection of assay plates, containing enclosed wells for storing and isolating chemical entities, is built and filled with a reactant/reagent pair. After allowing time for the reaction to occur, humans or machine optical devices measure and record the extent of the reaction. The expense of this procedure has made computational methods an important supplementary tool for drug development.

High Throughput Virtual Screening (HTVS) techniques may be categorized as either ligand-based or structure-based depending on the screening criteria. Ligand-based techniques are used to identify molecules that are similar to a set of ligands. The throughput of ligand-based methods is dependent on the similarity metric, but it is typically on the scale of thousands of chemicals per second. In contrast, structure-based HTVS techniques use molecular docking methods to identify molecules that have a high affinity for the target receptor. Structure-based HTVS techniques are consequently several orders of magnitude slower than their ligand-based counterparts, and a structure-based HTVS usually requires a parallel computing infrastructure to achieve an acceptable throughput. Working with parallel computing infrastructures

can require a large learning curve; fortunately, software programs such as WinDock [8], BDT [9], Glide [10], and DockoMatic [11, 12] have been built to facilitate HTVS and make HTVS tools more accessible to researchers who lack computer programming skills.

1.1.4 DockoMatic 2.0

DockoMatic 2.0 is open source software intended for use by researchers and educators. It consists of an intuitive graphical interface and a Perl script that uses the scripts from the AutoDockTools package to set up and run batches of docking jobs with AutoDock. In addition to single ligand/receptor docking, DockoMatic enables secondary ligand binding and structure-based HTVS. Ligands and receptors are input as protein database (pdb) files and the experimentally determined or predicted ligand binding domain on the receptor is specified with a grid coordinates file (gpf). DockoMatic is capable of submitting and subsequently monitoring hundreds of thousands of AutoDock jobs.

DockoMatic has the ability to generate a peptide's structure from an amino acid sequence and using the Obconformer program (part of the Open Babel tool suite [13]) for the structure's energy optimization. Dockomatic 2.0 includes the Treepack [14, 15] program, allowing *in silico* site-directed mutagenesis for complex peptide and protein structures using experimentally determined tertiary structure. As a result, a library of mutated peptides can be screened without manually generating the peptide's mutated structure; Treepack handles that on the fly. This greatly reduces the work associated with screening large collections of peptide mutants, an attribute of DockoMatic that has been leveraged in this thesis.

1.2 Towards a Drug for Parkinson's Disease

Epidemiological studies have demonstrated an inverse relationship between the development of neurodegenerative disorders and smoking [16, 17, 18], decreased levels of nicotinic receptors in postmortem studies of person's who had been afflicted with forms of dementia [19], and nicotine's ability to improve cognitive function in animals (including humans) [20]. These results have led many researchers to hypothesize that nicotinic receptors serve an important role in both neuronal survival and cognitive function [21]. This hypothesis has made nicotinic acetylcholine receptors (nAChRs), which are the primary targets for nicotine in the brain, a central focus for researchers interested in studying neurodegenerative disorders such as Parkinson's disease.

In addition to nicotine, α-conotoxins have been extensively studied as ligands for nAChRs. Conotoxins are one of a group of neurotoxin peptides that have been isolated from the venom of the marine cone snails of genus *Conus*. They are small peptides, consisting of 10-30 amino acid residues, and usually contain one or more disulfide bonds. α-conotoxins have received a lot of attention because, unlike nicotine, they have been found to selectively block specific nAChR subtypes. This has made them an important tool for probing the structure-function relationships of nAChRs.

1.2.1 Previous Work

Dr. Owen McDougal and his research team have been investigating α-CTx MII, a 16 amino acid peptide that exhibits an IC_{50} of 0.5 nM for the nAChR $\alpha 3\beta 2$-isoform [22]. α-CTx MII has a primary sequence of GCCSNPVCHLEHSNLC, contains two disulfide bonds between C2-C8 and C3-C16, and features an α-helix initiated at P6 and ending at S13 (see Figure 1.1). Site-directed mutagenesis studies on

nAChRs, investigations into the alteration of the primary sequence of α-CTx MII, and molecular modeling approaches have all been conducted to understand the selectivity and potency of α-CTx MII and its variants [23, 24, 25]. Notably, the results of a study by Bordia and colleagues showed the E11A mutant of the α-CTx MII peptide demonstrated a 50-fold binding preference for the $\alpha6\alpha4\beta2\beta3$-nAChR isoform [26].

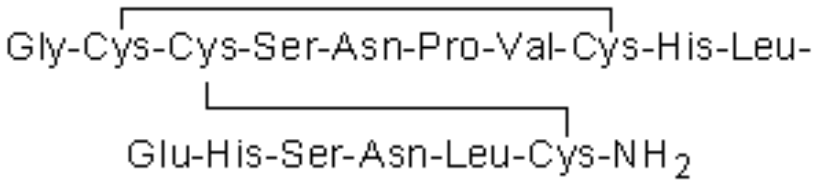


Figure 1.1: Primary sequence for α-CTx MII with disulfide linkages.

The findings of these studies motivated the team to develop a drug repurposing workflow that could be used to identify strong candidates for pharmaceuticals to treat Parkinson's disease. The workflow was:

Peptide Mutant Screening Perform a structure-based HTVS of an α-CTx MII mutant library to find peptides with high binding affinity for the $\alpha3\beta2$-nAChR.

Search PubChem Compound Use these peptide structures to perform a ligand-based HTVS of the PubChem Compound database and identify FDA approved drugs with 3-D conformers similar to the high affinity peptides.

Verification Screening Perform molecular docking calculations between the resulting small molecule drugs and the $\alpha3\beta2$-nAChR.

Synthesize Purchase or synthesize the highest affinity small molecules.

Wet Lab Use lab-based experiments, such as the two electrode voltage clamp experiment, to confirm that the predicted binding affinity translates to biological activity.

Extension Use the developed workflow to study the more biologically relevant nAChR $\alpha6\alpha4\beta2\beta3$-isoform in order to develop a drug to treat Parkinson's disease.

As the first step in the workflow, the team decided to perform an *in silico* combinatorial chemistry experiment-based on the α-CTx MII peptide. They constructed a library of α-CTx MII mutants and wanted to use structure-based HTVS methods to identify peptides with a high binding affinity to the $\alpha3\beta2$-nAChR. A pentameric homology model of the $\alpha3\beta2$ isoform of rat neuronal nAChR [27] was used for the initial experiment since it was the best available expression of a nAChR. Citing studies that identified the importance of the disulfide bonds in maintaining a rigid structure and the α-helix initiated by proline in restricting the peptide's length, the team decided to conserve the C2, C3, P6, C8, and C16 residues during their simulation. The plan was to allow the mutation of all other residues subject to the constraint that a residue's polarity and/or charge be conserved. That is, the polar and/or charged S4, N5, H9, E11, H12, S13, and N14 residues could mutate into polar and/or charged amino acids and the nonpolar G1, V7, L10, and L15 residues could mutate into nonpolar amino acids (excluding proline). However, the resulting mutation space consisted of an intractable 80 billion ($8^4 * 11^7$) peptides. To make the experiment tractable, the mutable residues were reduced to N5, H9, L10, E11, H12, and L15 and mutation into C was excluded. This reduced the mutation space to a collection of 640,000 ($8^2 * 10^4$) peptides. The HTVS capabilities of DockoMatic 2.0 were to be used to investigate the library.

Peptides are rarely used as pharmaceuticals since most are rapidly inactivated by gastrointestinal enzymes. As such, the second step in the workflow was using the top hits from the first step as the structural basis of a ligand-based HTVS of the PubChem Compound database. The PubChem project contains an online tool for searching any of the PubChem databases. The team planned to use the online tool to perform a 3-D shape-based similarity search against the top α-CTx MII mutants. Molecular docking calculations would be performed to predict the binding affinity (to the $\alpha 3\beta 2$-nAChR) of the identified small molecules. Lab-based techniques could then be used to verify the computational predictions, and the entire workflow could be used to probe the more biologically relevant nAChR $\alpha 6\alpha 4\beta 2\beta 3$-isoform in order to identify drugs for treating Parkinson's disease.

1.3 CAEPIDR

CAEPIDR was developed to facilitate the experiment described in Section 1.2. Following experimental validation, the software that was developed for the $\alpha 3\beta 2$-nAChR experiment was generalized and incorporated into DockoMatic. The resulting program, DockoMatic 2.1, is a powerful tool for exploring a receptor's conformational binding space with peptide mutation and identifying small molecule drugs for disease treatment.

In total, this thesis represents 4 major contributions to the field of biochemistry. First, the identification of small molecule drugs with a high binding affinity to the nAChR $\alpha 3\beta 2$-isoform. Second, GAMPMS, a model for efficiently and accurately searching a peptide's mutation space for receptor binding affinity. Third, a procedure for performing fast 3-D shape-based similarity searches over molecular databases.

Last, an extension to DockoMatic that includes an implementation of the GAMPMS model and the 3-D shape-based search procedure.

The body of this thesis has been divided into 2 chapters. Chapter 2 details the Genetic Algorithm Managed Peptide Mutant Screening (GAMPMS) model, which was developed to increase the throughput of DockoMatic's virtual screening process when screening peptide mutant libraries. The general model, as well as an implementation of it, are explained and experimental results are shown that demonstrate its soundness. A GAMPMS of the 640,000 α-CTx MII peptide mutant library was performed, in fulfillment of the workflow from Section 1.2, and is described. The results of the search are also presented. The chapter ends with a usage case of DockoMatic 2.1, demonstrating how to use the integrated GAMPMS program.

PubChem's online search tool does not work with small peptides such as the α-CTx MII peptide mutants, and an alternative tool was needed to search the database. Chapter 3 details the procedure used to search a local copy of the PubChem Compound database. The procedure is presented in the form of several pseudo-code algorithms with accompanying textual explanations. The PubChem Compound database search, for small molecules that were structurally similar to the α-CTx MII mutants, is described and the identified small molecules are presented. The chapter ends with a usage case of DockoMatic 2.1, demonstrating the use of the SimSearcher tool. SimSearcher can be used to quickly search local molecular databases (sdf format) for structurally similar 3-D conformers.

CHAPTER 2

GENETIC ALGORITHM MANAGED PEPTIDE MUTANT SCREENING

2.1 Introduction

2.1.1 Motivation

DockoMatic 2.0 facilitates the screening of large peptide mutant libraries by enabling the following workflow: enumerate the peptide ligands in a file, enter the receptor and specify its binding domain as a set of grid coordinates, select the number of GA runs to be used in AutoDock simulations, and submit the jobs to run on a server. However, the simultaneous submission of hundreds of thousands of jobs is likely to overload the cluster's scheduler and upset its administrators. In practice, screening a large library with DockoMatic 2.0 requires the user to partition the library and submit each partition separately, usually after some of the previously submitted jobs have completed. Since most users of DockoMatic 2.0 are not expected to have scripting skills, this procedure is not viable for large libraries.

Even with a script to monitor job submissions, performing molecular docking calculations for hundreds of thousands of ligands is computationally expensive. For example, using AutoDock 4.0 to simulate the binding of each of the 640,000 α-CTx

MII mutants on a 122 core cluster[1] (using 100 GA runs) would require:

1. $\approx$ 2 years to complete
2. $\approx$ 73.7 Terabytes of storage
3. 5246 partitions (for user submission)

assuming that each pose evaluation would requires 2 minutes to complete.

For many reasons, perhaps the most important of which is the inability of most researchers to access a large quantity of computing power for a long period of time, these requirements made the experiment intractable. An alternative method was needed to reduce the time and computational requirements. To address these problems, we asked the following question:

How can the α-CTx MII mutant library be screened in less time, with less computational resources, and without the need for human supervision?

One solution that immediately presented itself was to reduce the number of pose evaluations per ligand. This would result in a proportionate reduction in the time and storage requirements for the screening. However, using only a few pose evaluations would still require 5,246 submissions and would undermine the accuracy of the study.

2.1.2 Related Work

Generally applicable HTVS techniques exist that can greatly increase the throughput of virtual screenings. Typically, these approaches use machine learning algorithms to predict the binding affinity of new compounds. The affinity of docked ligands is used

[1]The size of the Computer Science Department's in house (beowulf) cluster at Boise State University

to train the algorithms. Once trained, the algorithms are several orders of magnitude faster than molecular docking techniques. As such, they can be used to screen huge libraries that would be intractable to screen with traditional docking methods.

In one such approach, molecular docking techniques are used to assess the binding affinity of a subset of a compound library. The docking results are then used to train a random forest algorithm which is used to predict the binding affinity of the library's remaining compounds [28]. Using this approach, Plewczynski was able to find 60% of the active compounds for a protein target when docking only 10% of the library. Random forests have been shown to compare favorably to other methods for screening molecular databases [29, 30]. Cherkasov et al. took a similar approach but built a QSAR regression model instead of a random forest. The QSAR was then used to predict the binding affinity for new compounds based on their 2D descriptors [31, 32]. The approach was able to reduce the time required for the screening of a 90,000 compound library (of potential human sex hormone-binding globulin (SHBG) binders) by 40% while finding 4 structures that were SHBG binders with high micromolar affinities. Different regression models, decision trees, neural networks, and many other classifiers from machine learning have been used to a similar effect. Ma [33] and Melville et al. [34] have both surveyed applications of machine learning classifiers within the field of HTVS, and the interested reader is encouraged to review their work for a more detailed treatment of the subject.

A genetic algorithm (GA) is another popular machine learning technique that has found application in computational and combinatorial chemistry. Sheridan et al. [35, 36] have used a GA to construct a library of synthesized oligomers from large sets of simple chemical fragments. The GA was used to optimize the diversity of the library's members with the assumption that maximum structural diversity would

result in the broadest coverage of the bioactivity space. The SELECT program [37] searches for optimally diverse libraries using a product-based approach to library design. With SELECT, a fully enumerated virtual library can be searched with a GA for combinatorial subsets that are optimally diverse. Studies demonstrating low hit rates yielded by maximally diverse libraries [38, 39] have resulted in the extension of SELECT to support multiobjective optimization. Fonseca and Fleming give a detailed survey of various approaches to multiobjective optimization [40], and several programs have been created to perform multiobjective optimization [41, 42].

2.1.3 Genetic Algorithm-Based Searching

While machine learning algorithms can be used in place of traditional docking methods in order to increase the throughput of a screen, they are less accurate. Additionally, the score that is assigned by a custom implementation (using a unique training set) of a machine learning algorithm to a ligand/receptor binding is rather meaningless in the context of a scientific community. In contrast, reporting an AutoDock score is useful since anyone can use AutoDock and attempt to replicate the reported results.

Ultimately, an unsupervised algorithm was needed that could use the results of previous docking jobs to make an informed decision as to which mutations would increase the peptide's binding affinity and which would decrease it. Algorithm 1 illustrates the essential characteristics of the desired model, which can be implemented effectively with a GA.

A GA is a non-deterministic, iterative search heuristic that uses techniques derived from natural evolution to search for an optimal solution to a given problem. A GA searches by first randomly generating a collection (population) of hypotheses (individuals) and then iteratively deriving a new population from the current one. A

Algorithm 1 Heuristic Screen

```
Dock random peptides from the library
while improvement in top affinity ligands do
    Infer goodness of mutations using data from docked peptides
    Generate peptide mutants which could test inference model
    Dock those mutants
end while
```

new population is built by applying a set of operators, known as genetic operators, to the members of the current population. Common genetic operators include: mutation, where individuals experience slight alterations; crossover, where pieces of the best known individuals are recombined to form new individuals; and selection, where individuals meeting some criteria are transferred to the next population. Central to GAs is the ability to assess the goodness (fitness) of an individual. In the case of molecular screening, the fitness of a ligand is likely to be directly proportional to its binding affinity with a target receptor. A GA continually derives new populations until some stopping criteria is met: typically when a set number of iterations have occurred or there is stagnation in the fitness of the top individuals.

A genetic algorithm can be used to perform a heuristic HTVS of a peptide mutant library, reducing the time and computing resources required for the screening without the need for algorithm training.

The Genetic Algorithm Managed Peptide Mutant Screening (GAMPMS) model was built to demonstrate the soundness of this hypothesis. In the GAMPMS model, a GA manages a structure-based HTVS of a collection of peptide mutants. The GA iteratively docks small collections of compounds and uses the binding affinities of previously docked compounds, in tandem with techniques derived from natural evolution, to select compounds for subsequent docking calculations. This approach

compares favorably to other machine learning methods, when screening large peptide mutant libraries, since it does not require training. The GA handles job submission and monitoring, allowing researchers to focus instead on setting up the next phase of the study.

2.2 GAMPMS

Algorithm 2 GAMPMS

```
stagnant_rounds, i ← 0
population_i ← generate random peptides
while stagnant_rounds < MAX do
    fitness evaluation of population_i
    if change in highest affinity ligands then
        stagnant_rounds = 0
    else
        stagnant_rounds++
    end if
    population_{i+1} ← genetic_operators(population_i)
    i = i + 1
end while
```

The GAMPMS model is shown in Algorithm 2, and a possible iteration is illustrated in Figure 2.1. As mentioned previously, the GAMPMS model uses a genetic algorithm to iteratively dock small collections of compounds (populations). Every population has the same number of peptides whose binding affinity with the target receptor is unknown, so a constant number of docking jobs are submitted at the beginning of each iteration. The GAMPMS model is therefore self-throttling, requiring no submission management on behalf of the user.

In GAMPMS, peptide mutant libraries are defined as a base peptide and a set of mutation constraints. Mutation constraints specify which residues are subject to mutation (mutable) and which amino acids can be substituted for each mutable

residue. As an example, the 640,000 $\alpha - CTx$ MII mutant ligand library from Section 1.2 is defined in Table 2.1.

Base Peptide: α-CTx MII	
Mutable Residue	**Substitutable Amino Acids**
N5	R,N,D,E,Q,H,K,S,T,Y
H9	R,N,D,E,Q,H,K,S,T,Y
L10	A,G,I,L,M,F,W,V
E11	R,N,D,E,Q,H,K,S,T,Y
H12	R,N,D,E,Q,H,K,S,T,Y
L15	A,G,I,L,M,F,W,V

Table 2.1: The 640,000 $\alpha - CTx$ MII mutant ligand library defined as a base peptide and a set of mutation constraints.

To be consistent with the GA terminology from before, peptide mutant screening is envisioned as an optimization problem. An individual is a sequence of amino acids such that there is one amino acid associated with each mutable residue in the peptide mutant library's definition. The goal of a peptide mutant screening is to find an individual that, when the individual's amino acids are substituted for the base peptide's mutable residues, results in a peptide with an optimal binding affinity with the target receptor.

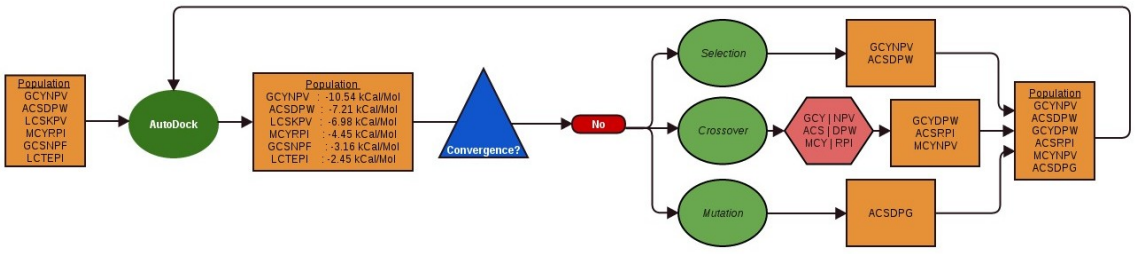


Figure 2.1: A possible iteration of GAMPMS

2.2.1 Implementation of the Genetic Algorithm

Representation of an Individual

In the GA that was integrated into DockoMatic 2.1, each individual (peptide mutant) is represented as the sequence of amino acids that are substituted for the base peptide's mutable residues. Since the base peptide is assumed to be the same for all mutants, it is not necessary to incorporate it into the representation. Using the single letter amino acid symbols, each individual can be represented as a character array.

Fitness Evaluation

In DockoMatic's GAMPMS implementation, evaluating the fitness of an individual is a two-step process. In the first step, a pdb file is built for each individual by submitting the individual's set of mutations, along with the base peptide, to the Treepack program. In the second step, the pdb file is used as the ligand for a molecular docking simulation against the target receptor. AutoDock is used to dock the ligand against the target receptor, and the Gibbs energy of the highest affinity pose is considered the fitness value for the individual. The number of pose evaluations to be used in each AutoDock simulation is configurable.

Genetic Operators

Three genetic operators are used in DockoMatic 2.1's GA.

Elitism The first operator is elitism: a special case of the more general selection operator. As previously mentioned, the selection operator selects individuals, based on

some user-defined criteria, to be carried over to the successive population. In the case of elitism, that criteria is fitness ranking within the population. In DockoMatic 2.1, the elitism operator selects the top $elite\_factor * 100\%$ of a population's individuals and adds them to the successive population.

Crossover The second genetic operator is two parent, two offspring N-point crossover using a fitness proportionate selection scheme. The implemented two parent, two offspring N-point crossover occurs in three steps. First, a selection scheme is used to chose two individuals (the parents) from the current population. Since a fitness proportionate selection scheme is used, the probability of an individual being chosen is directly proportional to its fitness ranking within the population. In the second step, a set of N indices are chosen from within a parent's range ($[1, individual\_length]$) and both parents are split into $N + 1$ pieces according to the indices. Finally, the pieces from both parents are combined, in an alternating fashion, to make two different offspring that share features of both parents. Figure 2.2 shows an example of two parent, two offspring, 2-point crossover.

In DockoMatic's implementation, the indices defining the splits are chosen randomly for each pair of parents. The number of indices (N) is configurable but defaults to $N = \lfloor \frac{individual\_length}{2} \rfloor$. Recall that $individual\_length$ is the number of mutable residues in the peptide mutant library's definition.

Mutation The final genetic operator is mutation, a process whereby each amino acid in an individual's sequence has a small chance of being replaced by a different amino acid from the substitution set of the associated mutable residue. Two versions of the mutation operator were implemented. In the first version, the probability of

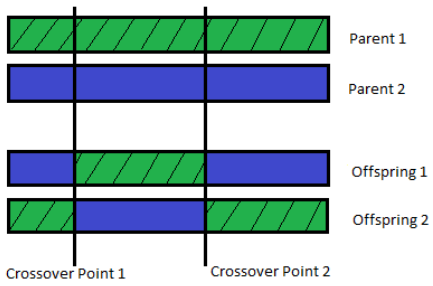


Figure 2.2: A pictorial representation of 2 parent, 2 offspring, 2-point crossover. Two individuals from the current population are split into pieces and those pieces are reassembled, in an alternating fashion, into 2 new individuals.

an amino acid being replaced by another amino acid was proportional to the historic probability of such a mutation occurring. The historic probabilities were derived from the PAM 1 mutation matrix (see Figure 2.3). In the second version, an amino acid had an equal chance of changing into any of the other amino acids in the associated residue's substitution set. The different versions were used in multiple screens of a known affinity peptide mutant library (see Section 2.3). It was found that the uniform mutation technique had a (statistically insignificant) lower average number of docking calculations for a screening that found similar affinity ligands. Due to its relative simplicity, uniform mutation is used in DockoMatic 2.1's GA.

In DockoMatic 2.1, the user can specify the *mutation_rate*, which is the probability that any amino acid is altered during the mutation operation.

Terminating Condition

A genetic algorithm iteratively builds new populations, using a set of genetic operations, until some criteria is met (see Algorithm 2). In DockoMatic 2.1, the genetic algorithm stops generating new populations when there has been no change in the

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
Ala A	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
Arg R	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
Asn N	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
Asp D	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
Cys C	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Gln Q	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
Glu E	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
Gly G	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
His H	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
Ile I	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
Leu L	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
Lys K	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
Met M	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
Phe F	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
Pro P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
Ser S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
Thr T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
Trp W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Tyr Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
Val V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

Figure 2.3: The PAM 1 mutation matrix. The number at coordinates row, col represents the average number of times that the amino acid *row* has mutated into the amino acid *col* for every 10,000 mutations that have been identified.

top_X highest affinity peptides over the last λ iterations. Intuitively, using larger λ values will increase the probability of finding the highest affinity ligands in the library, whereas using lower λ values will reduce the screening's run time. The values of both top_X and λ are configurable.

Parameters

In review, the behavior of the GA is parameterized with 5 numeric values, representing:

$elitism\_factor$: the percent of the population's top individuals that are carried over to the successive population.

$mutation\_rate$: the probability of a mutation occurring within an individual.

top_X : the number of optimal peptide mutants to identify.

$|\rho|$: the population size.

Name	Size	Residues	Amino acids
$library_{64}$	64000	N5	-
$library_{46}$	46656	N5	D
$library_{32}$	32768	N5	D,H
$library_{25}$	25088	N5	D,H,V
$library_{16}$	16807	N5	D,H,V,Q
$library_{8}$	8000	N5,L15	-
$library_{5}$	5382	N5,L15	D

Table 2.2: Additional constraints were added to the 640,000 $\alpha - CTx$ MII mutant library in order to create smaller libraries. This table lists, for a specific library, the residues that were no longer deemed mutable and the amino acids that were no longer substitutable.

λ : the GA stops after λ iterations of no change in the top top_X individuals.

$|\rho| = 50$ and $top\_X = 10$ were used as the default values in the experiments in section 2.3. The following notation is used to list the additional parameter values for a GAMPMS experiment: $\{\ elitism\_factor, mutation\_rate, \lambda\ \}$.

2.3 Validation

For proof of concept, the GAMPMS implementation was used to screen libraries of $\alpha - CTx\,MII$ mutants for binding affinity towards the $\alpha 3\beta 2$ isoform of rat neuronal nAChR. Each library was a subset of the 640,000 $\alpha - CTx\,MII$ mutant library from Section 1.2; the libraries are defined in Table 2.2.

The most important subset was the 64,000 mutant library ($library_{64}$) resulting from the removal of N5 from the set of mutable residues. Since it was small enough to screen exhaustively but large enough to yield interesting results, it was used for most of the experiments in this section.

2.3.1 Experimental Setup and Evaluation Metrics

The soundness of GAMPMS was assessed by comparing its results to those of an exhaustive, structure-based HTVS (i.e., molecular docking calculations were performed for each compound) of the same library. The binding energy of the top hits and the number of docking calculations performed were the focus of the assessment. Comparison of the number of docking calculations was straightforward; every peptide was docked with the exhaustive approach whereas only a fraction of them were docked with GAMPMS. The number of docking calculations performed by a GAMPMS is reported as a percent of the library's cardinality, denoted $\%Docked$.

Comparison of the binding affinity of the top hits is a little more complicated. We define $\Sigma_{x,r,s}$ as the summation of the binding energy, against receptor r, of the x best peptides identified by a HTVS s. The similarity in binding affinity between a GAMPMS and an exhaustive screening is measured as:

$$\%Affinity = \frac{\Sigma_{x,r,GAMPMS}}{\Sigma_{x,r,exhaustive}}.$$

All $\%Affinity$ scores in the current study use $x = 10$ and $r = \alpha 3\beta 2$ nAChR.

AutoDock 4.0 was used, with 30 pose evaluations per docking (i.e., ga_runs = 30), for the exhaustive, structure-based HTVS of $library_{64}$ against the $\alpha 3\beta 2$ nAChR model. A fitness map, which associated a peptide to its highest affinity score from AutoDock, was constructed from the HTVS results and used as the fitness function for each GAMPMS mentioned below. This map allowed a GAMPMS to run in a fraction of a second and removed the inconsistency from AutoDock scoring.[2]

The results of a GA are dependent on the values of its parameters (e.g., $mutation\_rate$, $elite\_factor$, λ, etc.) and the pseudo-random numbers generated during the crossover

[2]If a ligand is given a different fitness score in each screening, it can be difficult to compare the results of those screenings.

and mutation operators. As such, a meaningful assessment of GAMPMS required multiple screenings of each library. A given GAMPMS, identified by the library screened and the parameter values used, was repeated multiple times to isolate the variance resulting from the random number generator. We report the results of 1910 GAMPMS's of $library_{64}$ and 10 GAMPMS's of each of the smaller libraries. The fitness map from above served as the basis of the comparison between a GAMPMS's identified peptides and those that would have been identified by an exhaustive screening.

2.3.2 GAMPMS Consistency

The consistency of GAMPMS, as it relates to the $\%Affinity$ and $\%Docked$ scores, was assessed with 300 independent GAMPMS's of $library_{64}$. For 100 of the experiments, the set of parameter values $\{\ .2, .02, 2\}$ was used to produce low $\%Docked$ scores. The more general set of parameter values $\{\ .2, .05, 12\ \}$ was used for an additional 100 experiments. The last 100 experiments were run with a set of parameter values, $\{\ .3, .08, 25\ \}$, selected to yield a high $\%Affinity$ score. The results are captured by the histograms in Figures 2.4 and 2.5. Figure 2.4 shows the frequency of $\%Affinity$ scores across the 300 experiments, whereas Figure 2.5 shows the frequency of $\%Docked$ scores across the same experiments. In both of these figures, the height of $bar[i]$ represents the fraction of GAMPMSs that resulted in a score $x : bar[i] \geq x \geq bar[i-1]$. The exception to this rule is $bar[0]$, whose height represents the fraction of GAMPMSs that resulted in a score $x : x \leq bar[0]$. For perspective, 100 screens were performed by docking 5,700 (the largest number of peptides docked during the 300 experiments from above) random members of $library_{64}$. This random sampling resulted in a mean $\%Affinity$ score of 91.7 ($\sigma = 1.3$).

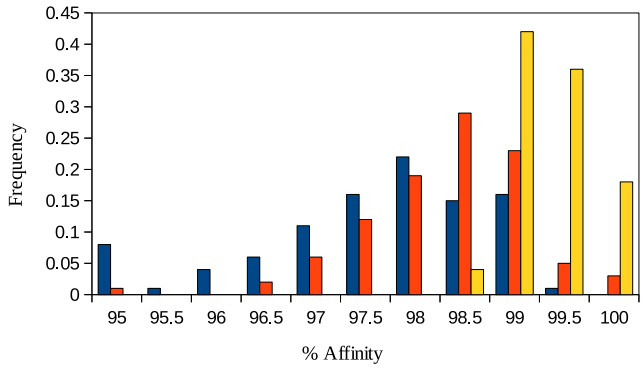


Figure 2.4: A histogram representing GAMPMS's soundness when screening $library_{64}$. The height of a bar represents the frequency of the range of $\%Affinity$ scores. The blue bars summarize the data from 100 experiments with "low % Docked" parameter values, the yellow bars summarize the data from 100 experiments with "high %Affinity" parameter values, and the orange bars summarize the data from 100 experiments with more neutral parameter values. The lowest recorded $\%Affinity$ score across all 300 screens was 94.05.

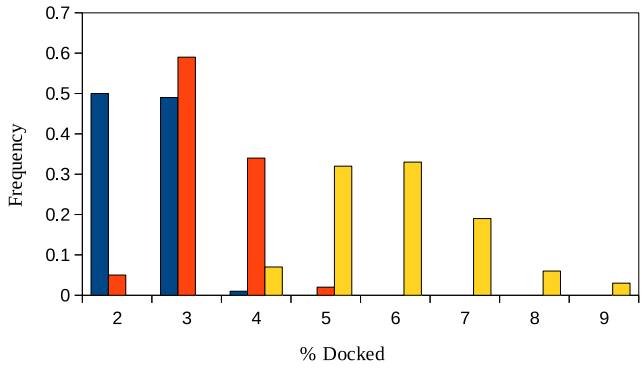


Figure 2.5: GAMPMS's efficiency when screening $library_{64}$. The x-axis shows the percent of the library that was docked and the y-axis shows the associated frequency. The color scheme is the same as detailed in Figure 2.4.

Figures 2.4 and 2.5 demonstrate that GAMPMS can consistently find some of the highest affinity ligands while performing docking calculations for a relatively small number of the peptides. Moreover, the results indicate that the variance in performance can be tuned with a single parameter. Specifically, increasing λ had the effect of decreasing the variance in the $\%Affinity$ scores and decreasing λ had the effect of decreasing the variance in the $\%Docked$ scores.

2.3.3 GAMPMS Scalability

$|\rho|$ vs Performance

To assess how the GAMPMS implementation scaled with population size, $|\rho|$, 1,600 independent GAMPMSs of $library_{64}$ were performed. The value of $|\rho|$ was changed for every 100 screens. The parameter values $\{\ .25, .05, \lceil \frac{900}{|\rho|} \rceil\ \}$ were used for all 1,600 GAMPMSs. The $\%Docked$ score and the number of iterations required for convergence were recorded and are displayed in Figure 2.6.

The time required to apply the genetic operators, submit docking jobs, and parse the fitness values is insignificant compared to the time required for the molecular docking simulations. As a result, the approximate run time of a GAMPMS can be expressed with Rquation 2.1.

$$time_{total} = iterations * \left\lceil \frac{time_{job} * |\rho|}{available\_cores} \right\rceil \tag{2.1}$$

where $time_{job}$ is the amount of time required to dock a ligand to a receptor with AutoDock.[3]

[3]This time is dependent on the number of pose evaluations being used; 1-3 minutes per pose evaluation depending on your system's speed.

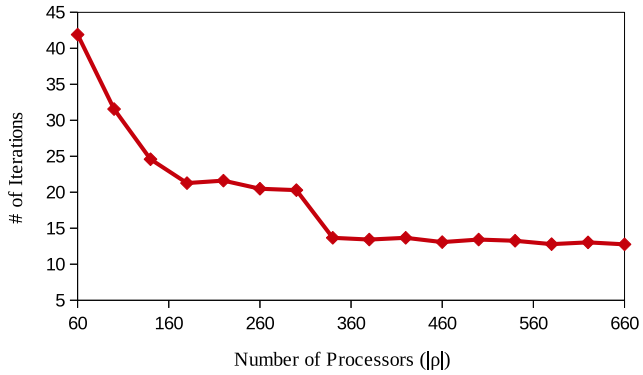


Figure 2.6: GAMPMS's performance measures based on the number of processing cores used, $|\rho|$. The y-axis shows the number of iterations (submissions) needed for a GAMPMS to converge. Each diamond represents the mean value, for the associated ρ, from 100 GAMPMSs of $library_{64}$.

It can been seen from Equation 2.1 that the lowest run time can be achieved by choosing the population size, $|\rho|$, to be less than or equal to the number of available processing cores. In this case, all docking jobs can be run in parallel. Selecting a larger value of $|\rho|$ will at least double the run time of the screen.

Library Size

To approximate how the $\%Docked$ score scales with larger libraries, $library_{64}$, $library_{46}$, $library_{32}$, $library_{25}$, $library_{16}$, $library_{8}$, and $library_{5}$ were each screened 10 times with the GAMPMS implementation. Each screening used the same set of parameter values: $|\rho| = 30, topX = 5$ and $\{\ .2, .04, 5\ \}$. The $\%Docked$ score from each GAMPMS was plotted and is shown in Figure 2.7.

The inverse relationship between the size of a library and the $\%Docked$ score indicates that the number of docking jobs required for a GAMPMS increases more

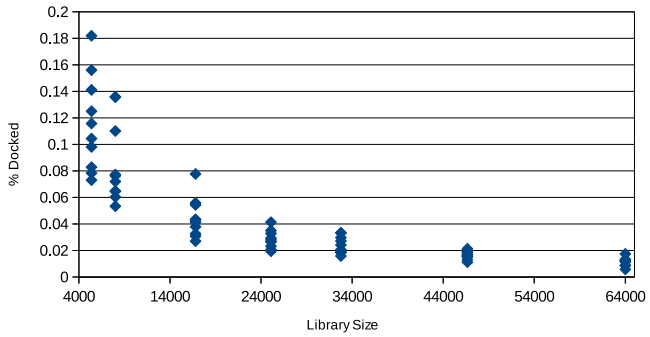


Figure 2.7: The %*Docked* scores from 10 GAMPMSs of different sized libraries. Each diamond represents a %*Docked* score.

slowly than does the cardinality of the compound library. So the average %*Docked* score for a GAMPMS scales very well with library size. It is also interesting to note the negative correlation between the variance in the %*Docked* scores and the size of the library. If the trend holds for larger libraries, it might be possible to accurately predict the number of iterations required for the screening to complete.

2.4 A GAMPMS of the 640,000 α-CTx MII Mutant Ligand Library

The GAMPMS implementation was used to screen the 640,000 α-CTx MII mutant library for binding affinity to the $\alpha 3\beta 2$ nAChR. The results from Section 2.3 were used to identify a set of parameter values that would yield peptides with a relatively high binding affinity without requiring more than a week to run. The GA was configured to mutate one in every 50 amino acids, carryover the top 40% of each population, use two parent, two offspring, 3-point crossover, and attempt to identify a set of 50

Peptide	Energy (kcal/mol)
MII:N5Y:H9Y:L10W:E11T:H12N	-21.07
MII:N5Y:L10W:E11Q:H12S:L15F	-20.91
MII:N5Y:H9Y:L10W:E11Q:H12S:L15V	-20.91
MII:N5Y:L10W:E11S:H12S:L15F	-20.88
MII:N5Y:L10W:E11S:H12S:L15W	-20.79
MII:N5Y:H9S:L10W:E11K:H12S:L15F	-20.74
MII:N5Y:L10W:E11Y:H12S:L15V	-20.73
MII:N5Y:H9K:L10W:E11S:H12N:L15G	-20.71
MII:N5Y:H9N:L10W:E11S:H12S:L15W	-20.68
MII:N5Y:L10W:E11K:H12S:L15G	-20.66
MII	-12.38

Table 2.3: The 10 highest affinity peptides found with a GAMPMS, the base peptide (α-CTx MII), and their associated Gibbs free energy of binding.

optimal peptides. The GA terminated after it went 5 rounds without an improvement in the binding energy of the 50 top peptides.

The screening was performed using 128 cores on Idaho National Laboratory's (INL's) Fission cluster.[4] 40 pose evaluations were used in the AutoDock docking simulation for a ligand/receptor binding. Had DockoMatic 2.0 been used for the HTVS, it would have required 5,000 submissions (of 128 AutoDock jobs) and roughly 292 days. Instead, the GAMPMS implementation submitted 73 groups of 128 jobs (for a total of 9,344 molecular docking jobs) and finished in 4 days and 6 hours. The 10 highest affinity mutants identified by GAMPMS, as well as the α-CTx MII peptide (for reference), are shown in Table 2.3.

[4]The screening was funded as part of the Idaho University Consortium

2.5 Usage: Screening Peptide Mutants with DockoMatic 2.1

A workflow has been included in DockoMatic 2.1 to facilitate the screening of large peptide mutant libraries. The workflow is illustrated here by walking through the setup of a GAMPMS of the 640,000 α-CTx MII library from Section 1.2.

To screen a peptide mutant library, DockoMatic 2.1 must be loaded and the receptor's pdb and gpf file, the output directory, and the ligand's (base peptide's) pdb file must be entered in the associated fields. The number of AutoDock cycles (pose evaluations) must be specified, as well as any special options for the swarm utility. When these steps are completed, the "Mutation Screening" check box can be marked to bring up the peptide mutant screening wizard (*the wizard*). Figure 2.8 shows what DockoMatic's main window looks like when setting up a GAMPMS of an α-CTx MII (MII) mutant library against the $\alpha 3\beta 2$ nAChR (A3B2_2br8).

the wizard was designed to simplify the process of defining the peptide mutant library and, if desired, to configure the genetic algorithm to be used for a GAMPMS of the defined library.

2.5.1 Defining the Peptide Mutant Library

Recall that a peptide mutant library can be defined as a base peptide and a set of mutation constraints, which specify the mutable residues and the amino acids that can be substituted for each mutable residue. The base peptide is input as the ligand in DockoMatic's main screen, and the mutation constraints are specified in the first two steps of *the wizard*. In the first step, residues belonging to the base peptide are selected for mutation. In the second step, a set of amino acids is associated with each of the mutable residues.

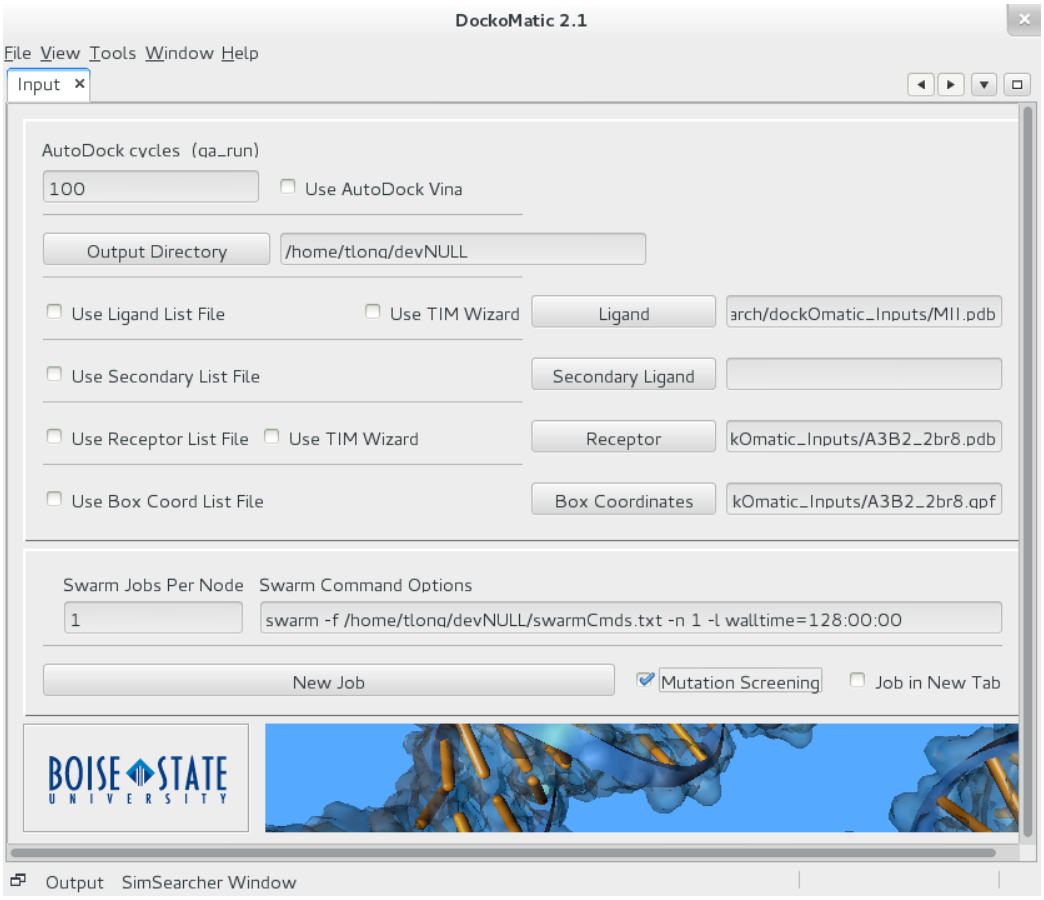


Figure 2.8: The main window component of DockoMatic 2.1. To screen a peptide mutant library, the user needs to input the necessary data files and then select the "Mutation Screening" option, which has been highlighted in red in the picture.

Mutable Residues

When *the wizard* is first loaded, the base peptide's amino acid sequence is parsed from the pdb file and displayed as a list of residues. Residues are deemed mutable if they are selected in the list. One can use the standard multiselect and range select features (respectively, CTRL + click and select first, hold SHIFT, select last) to select residues for mutation. The "Go To" button is useful when dealing with a larger base peptide (more residues). The button can be used to center the list on a certain residue (enter its index and click "Go To") or on the next type of residue (enter the single letter representation for the amino acid and hit "Go To"). For convenience, the currently selected residues will be displayed at the bottom of the screen.

To setup the GAMPMS for the the 640,000 α-CTx MII mutant library from Section 1.2, hold down the CTRL button, click the N5, H9, L10, E11, H12, and L15 residues, and then click Next. Figure 2.9 shows the completed step before clicking Next.

Mutation Constraints

Once the mutable residues have been selected, a peptide mutant library can be defined by constraining the mutation of each mutable residue. This is achieved by entering a substitution set for each mutable residue. One specifies a mutable residue's substitution set by selecting the mutable residue from the top list, selecting the substitution set from the lower list (using the same list features described above), and then hitting the "Bind Constraints" button. The same substitution set can be easily assigned to multiple residues by multiselecting from the residue list. Figure 2.10 shows the nonpolar residue substitution set selection for the 640,000 α-CTx MII

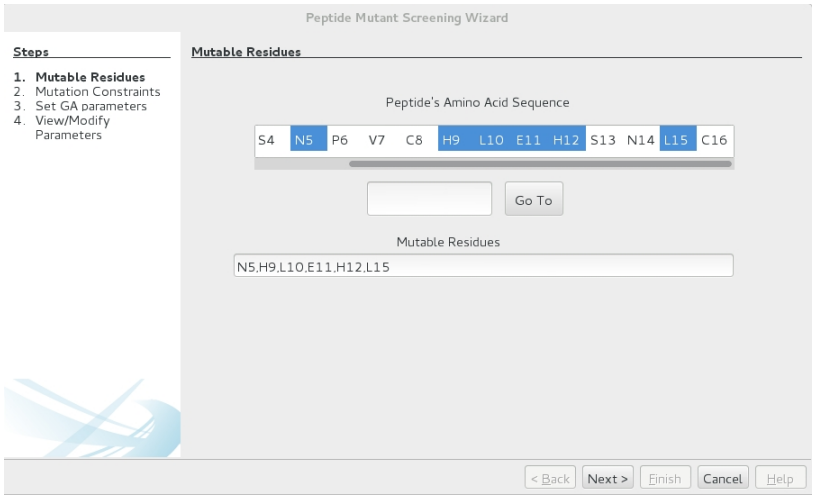


Figure 2.9: The first step in the Peptide Mutant Screening wizard. The user selects residues from the base peptide that can be mutated during the experiment. The Go To button can be used to focus the list on an acid type or index. Clicking the "Next" button goes to the Mutation Constraints screen.

mutant library.

A peptide mutant library has been fully defined when a substitution set has been selected for each mutable residue. Clicking "Finish" returns the user to the main DockoMatic screen, where they can perform an exhaustive structure-based screen of the library by clicking the "New Job" button. Alternatively, clicking the "Next" button allows the user to configure a GA for a GAMPMS of the library.

2.5.2 Setting the GA's Parameters

Configuring a GAMPMS requires specifying $|\rho|$, top_X, and the desired balance between the opposing goals of minimizing the number of docking calculations required for the screening and maximizing the estimated binding strength of the resulting ligands. The population size, $|\rho|$, determines the number of molecular docking jobs

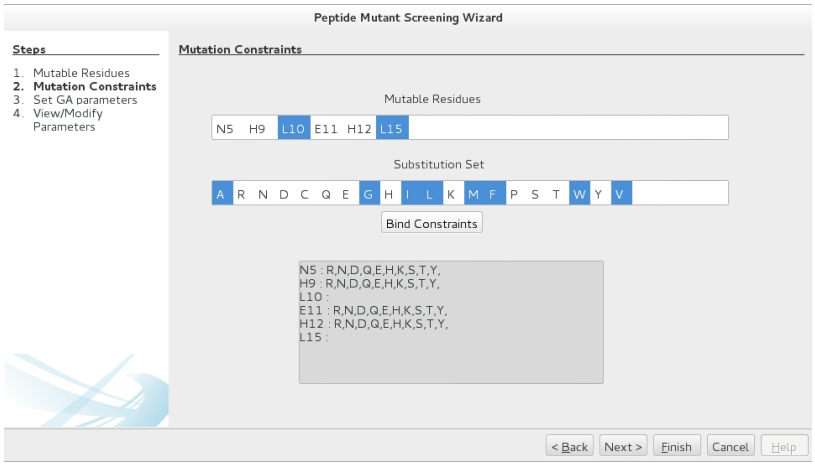


Figure 2.10: The second step in the Peptide Mutant Screening wizard. The user assigns a substitution set to each mutable residues by selecting the relevant list elements and hitting the Bind Constraints button. The resulting library can be screened exhaustively ("Finish") or with a GAMPMS implementation ("Next").

that will be submitted at each iteration. To minimize the time required for a GAMPMS, $|\rho|$ should not exceed the number of processors that can be dedicated to the screening. As mentioned in Section 2.2.1, the GAMPMS process ends when there has been no change in the *top_X* highest affinity ligands in the last λ rounds. Thus, the value of *top_X* affects both the number of iterations that will be generated and the cardinality of the "optimized" result set.

In Section 2.2.1, it was shown that a GA's performance can be configured by specifying values for genetic operators' parameters and the termination condition. However, part of DockoMatic's purpose is to make molecular docking tools accessible to students. It is assumed that DockoMatic 2.1 users will be unfamiliar with GAs and thus unable to configure a GA's parameters to meet their screening goals. So instead of inputting numeric values for unfamiliar parameters, a novice user tells *the wizard*

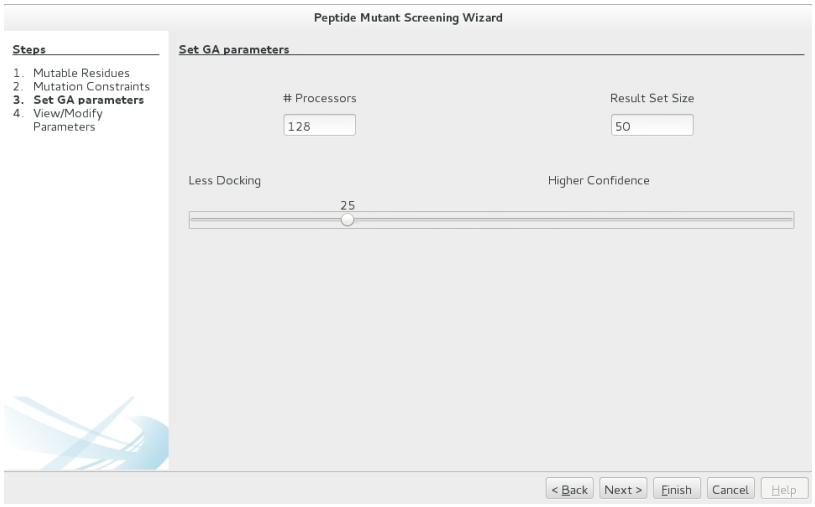


Figure 2.11: Configuring a GAMPMS requires setting the population size, *top_X* (Result Set Size), and the sliding bar's position. The remaining parameters are generated automatically based on these 3 values. The generated values can be viewed/modified by clicking *Next*, or the GAMPMS can be instantiated by clicking *Finish*.

how to weigh the opposing goals of minimizing the number of docking calculations required for the screening and maximizing the estimated binding strength of the resulting ligands. This is achieved by adjusting *the wizard*'s sliding bar (see Figure 2.11). Appropriate values for $elitism\_factor, mutation\_rate$, and λ are generated automatically.

When all 3 values have been input, the generated parameter values can be viewed and/or modified by clicking *Next*. Alternatively, the GA can be instantiated by clicking *Finish*. Clicking the "New Job" button from the main DockoMatic screen will start a GAMPMS of the defined library.

2.6 Conclusion

GAMPMS uses a genetic algorithm to manage a structure-based high throughput virtual screening of a peptide mutant library. It is capable of reducing the number of molecular docking calculations required to screen a 640,000 peptide library by roughly 98.5% while finding ligands with high binding affinity with the target receptor. Its performance is easily configurable, allowing the user to prioritize the number of docking calculations performed or the binding affinity of the identified peptides. GAMPMS scales well with the size of the peptide mutant library, exhibiting an inverse relationship between the size of the library being screened and the percent of the library's compounds that need to be docked.

GAMPMS has the advantage of significantly reducing the number of docking calculations required for structure-based HTVS. In comparison to other approaches, GAMPMS does not require training or any form of human supervision. Instead, GAMPMS iteratively docks populations of mutants using evolutionary techniques and the binding energies of previously docked mutants to select compounds for subsequent docking simulations. The types of libraries that can be screened with GAMPMS are limited to combinatorial libraries such as those that result from mutating a molecule. As with most other non-trivial heuristic searches, there is no guarantee that a GAMPMS will find the highest affinity ligand from a library. The integration of the GAMPMS model into DockoMatic 2.1 represents an important extension to the suite's investigative potential.

CHAPTER 3

SIMILARITY SEARCH

3.1 Introduction

3.1.1 Motivation

Receptor-driven drug repurposing typically involves a ligand-based HTVS of small molecule databases. Many small molecule databases exist, including DrugBank, PubChem, BindingDB, ChemSpider, and the Beilstein and Gmelin databases (available through Reaxys), which provide web-based tools for performing substructure and molecular similarity searches. However, each database uses a different interface and different algorithms for their search. This can be problematic when one wishes to perform a specific type of search over a database that contains the necessary data but not the tools.

Having identified a set of peptides with a high-binding affinity to the target receptor, the next step in the workflow (described in Section 1.2) was the identification of small molecule drugs that closely resembled the 3-D shape of the α-CTx MII mutant ligands. This was to be achieved using PubChem's online search tools. The tools are accessible free of charge, courtesy of the NIH, and can be used to search any of the PubChem databases. The compound structure similarity search was identified as the most appropriate for our experiment. The 200 highest affinity α-CTx MII mutants

from the GAMPMS were used as input for the search. However, it was discovered that PubChem's 3D search, which is geared towards small, drug-like molecules, can only generate a model for compounds containing less than 16 rotatable bonds. The α-CTx MII mutants had at least 20 rotatable bonds, and as a result the tool failed to generate a model for any of them. An alternative means of searching PubChem was needed.

While there are many chemical databases, PubChem is arguably the most useful for drug repurposing. It contains patent information, bioassay results, bioactivity data, and structural information for over 51 million small molecules, including drugs that have already been approved by the FDA.[1] For these reasons, searching PubChem was a vital part of the research project.

PubChem provides an FTP service for downloading their Compound database, but the question remained:

How can a local copy of the PubChem database be quickly searched for molecules that are "similar" to our small peptides?

First, a means of assessing similarity was needed. PubChem 3D uses volume overlay techniques to compare each molecule in the database to a set of (a few thousand) structurally diverse reference molecules. A binary fingerprint is then constructed for each molecule by associating a bit with every reference molecule. If a reference molecule is similar to a molecule, the corresponding bit is set. When all the molecules have a fingerprint, the similarity between molecules is quickly assessed using the Tanimoto coefficient (shown in Equation 3.1) to compare their binary fingerprints.

[1] Although all 4 types of information are not available for all molecules

$$Tanimoto = \frac{AB}{A + B - AB} \tag{3.1}$$

where A and B are the counts of fingerprint set bits in the molecule pair (respectively) and AB is the count of bits in common.

Generating the fingerprint for every molecule is a computationally demanding endeavor, but they can be precomputed in an completely parallel manner. PubChem 3D used this technique to reduce the search time by two orders of magnitude [43]. However, we did not want to spend months determining reference shapes and generating fingerprints in order to search the database a few times. A different technique was needed to assess similarity within the database.

3.1.2 Similarity Metrics

Ligand-based HTVS must use a similarity metric that is easy to compute (or at least precomputable) and discriminative. Since ligand-based HTVSs are often used to discover ligands with similar bioactivity, without needing to perform expensive molecular docking calculations, similarity metrics account for ligand characteristics that are thought to be important in determining binding properties. Metrics are broadly classified as structure or (pharmacophore) feature based, although some metrics handle both. Structure-based metrics assess the shape of a molecule using the molecule's 2-D or 3-D coordinates. Graph comparison techniques such as Maximum Common Subgraphs [44, 45] and graph kernels [46, 47], regression models such as the Quantitative Structure-Activity relationship (QSAR) model [48], and volume overlay techniques are commonly used to assess molecular similarity [49]. Feature-based metrics focus on the presence and location of chemical features (e.g., hydrogen donors

and acceptors, ring centers, and charged atoms) that are needed to trigger a biological response from a specific biological target. QSAR and graph techniques, which work well with a descriptor-based representation, can also be used to assess feature-based similarity.

The size of chemical databases (terabytes) often necessitates the use of molecular signatures: compact, trivially comparable entities that encapsulate a molecule's relevant features. Using signatures divides the work of searching into two steps: signature generation and signature comparison. Signatures are typically designed to transfer complexity from the comparison aspect and into the generation aspect. Since signatures can be precomputed, search times are typically much faster when signatures are used. PubChem 3D uses binary fingerprints, which require computationally demanding volume overlay techniques to generate, as a molecule's signature.

3.2 Search Model

It was hypothesized that a shape distribution technique [50] could be used to assess 3-D shape similarity between molecules. With a shape distribution technique, a shape sampling function is used to construct a distribution of measurements. The distribution serves as the molecule's signature and a distribution difference measure, such as the χ^2 test, is used to quickly compare the signatures.

While distribution tests are fast, performing 51 million of them can take a substantial amount of time. Multilevel K-means clustering provided a sound method for decreasing search time since it would allow a recursive search operation to compare the target molecule with a subset of the clusters. This would reduce the number of comparisons required and therefore the search's run time.

A model was devised to allow quick similarity searches, with any target molecule, over local molecular databases. For clarity, using a molecule M as the basis of a similarity search (i.e., searching with a target molecule M) over a database D is equivalent to searching D for items that are similar to M. The model consisted of three steps:

Map Map all molecules to signatures.

Cluster Cluster the signatures for quicker searching.

Search Map the target molecule to a signature and search the (clustered) database for similiar signatures.

Map must occur first to make a search tractable. The Cluster step is optional but highly recommended because of its ability to reduce search time by a few orders of magnitude. The Map and Cluster steps are computationally expensive but only need to be performed once per database and can be pre-computed. Search is the end product of the process, allowing users to quickly perform molecular similarity searches over the database.

3.2.1 Map

Generating signatures is an embarrassingly parallel problem, which is made even simpler by the fact that molecular databases are typically downloaded as a collection of data files. To quickly generate signatures, it is necessary to first partition the database files to create a partition for each available processing core. Then, using a function ($map()$) to generate a signature for a molecule, an instance of the Map_DB algorithm (Algorithm 3) can be run on each processor in order to generate signatures

for the associated partition. The important work in the Map step occurs in the function $map()$.

Algorithm 3 Map_DB

```
for each DB_file ∈ partition do
  for each molecule ∈ DB_file do
    mMolecule ← map(molecule)
    write(mMolecule)
  end for
end for
```

$map()$ is responsible for generating a molecule's signature (mapping a molecule to a signature). The signature needs to be both descriptive and easily comparable so that a similarity metric can be discriminative and efficient, respectively. Signatures can be pre-computed, making the computational complexity of their generation less important than that of the similarity metric.

Shape Similarity

The shape distribution approach described in Section 3.3 was used to gauge the 3-D shape similarity of two molecules. In this approach, a shape sampling function is applied to a 3-D shape in order to attain a set of measurements. The distribution of these measurements is used as the shape's signature. Any distribution difference test (e.g., χ^2) can be applied to the two signatures to quickly judge the similarity of the associated molecules. This approach has been successfully applied to the comparison of 3-D protein structures [51]. The implemented shape sampling function measures the euclidean distance between unique pairs of atoms within a molecule. The amount of computation needed for sampling is configured by defining the number of samples to take. Since most of the molecules within PubChem Compound are small (less

than 50 atoms), it was feasible to generate a distribution using all $\frac{N(N+1)}{2}$ unique measurements (N is the number of atoms in the molecule).

A distribution is represented as a histogram containing a constant number of bins and a maximum measurement threshold. Algorithms 4 and 5 demonstrate the process used to create a molecule's shape signature. Algorithm 5 is used as *map*() in Algorithm 3 to generate shape signatures for a group of data files. Four similarity metrics were implemented for signature comparison: Chi Square, L1 norm, L2 norm, and the Root of Products test. These distribution tests are described in Appendix A.

Algorithm 4 Shape_Sample(*molecule*)

```
for each atom_i ∈ molecule do
  for each atom_j ∈ molecule do
    if i ≠ j then
      if NOT sampledList.contains((i, j)) then
        measurements.add(L2Norm(atom_i, atom_j))
        sampledList.add((i, j), (j, i))
      end if
    end if
  end for
end for
return measurements
```

3.2.2 Cluster and Search

Clustering is an optional step, although it is highly recommended for shape-based similarity searches. Without clustering, searching a database with molecule q requires comparing the signature of q to every signature in the database. For the PubChem database, this would mean performing 51 million calculations. Clustering the signatures can reduce the number of similarity calculations by a few orders of magnitude.

Algorithm 5 map(*molecule*)

```
ID ← molecule.getID()
measurements ← Shape_Sample(molecule)
bins ← Integer[num_bins]
for each measurement ∈ measurements do
  i ← 1
  for 1 to num_bins do
    if measurement < (i * bin_width) then
      bins[i] + +
      BREAK
    end if
    i + +
  end for
  if i > num_bins then
    bins[num_bins] + +
  end if
end for
for each bin ∈ bins do
  bin ← bin/measurements.size()
end for
return ID, bins
```

Let us imagine dealing with a database containing $|DB|$ signatures. If the database is clustered with the K-means algorithm, where $K = k_1 * k_2 * \ldots * k_n$, then an effective search could be performed with

$$\approx K + \frac{|DB|}{K} \tag{3.2}$$

similarity calculations by comparing the target molecule to each of the K cluster centers and then to each of the $\approx \frac{|DB|}{K}$ signatures within the cluster whose signature was most similar to the target molecule. If $|DB| \gg K$, a single K-means clustering would reduce the number of comparisons by a factor of K.

Nested (multilevel) clustering can be used to further reduce search time. In multilevel clustering, most clusters contain subclusters. Algorithm 6 gives a pseudo code algorithm for the idea, with a user calling $NlevelCluster(N, DB)$ to perform N level clustering with the K-Means clustering algorithm. A "Big Data" implementation of the K-means clustering algorithm was used for generating the two outermost clusters, whereas an in-memory implementation was used for subsequent clusters. Both of these implementations are discussed in Appendix B.

Algorithm 6 $NlevelCluster(level, DB)$

```
KMeans_Cluster(DB)
if level > 1 then
  for each cluster ∈ DB do
    NlevelCluster(level − 1, cluster)
  end for
end if
```

If the DB database is clustered with n-level clustering, where level i has k_i clusters (recall $K = k_1 * k_2 * \ldots * k_n$ from above), then the approximate number of similarity calculations required for an effective search is given by:

$$\approx \sum_{i=1}^{n} k_i + \frac{|DB|}{K} \tag{3.3}$$

As a result, the difference in the number of required signature calculations between the n-level clustering and the single clustering is given by:

$$\prod_{i=1}^{n} k_i - \sum_{i=1}^{n} k_i \tag{3.4}$$

So if $|DB| = 50$ million and $K = 20 * 20 * 20 = 8000$, then multilevel clustering can reduce the search time by $\approx 65\%$, compared to a single K-means clustering.

The idea used in the single level cluster search can be easily extended to handle nested clusters. Algorithm 7 shows a recursive technique that can search a collection of signatures that have been subjected to N-level clustering.[2] To search with the target molecule q, one would call $Search(q, DB)$.

Algorithm 7 $Search(q, DB)$

```
if DB contains clusters then
  for each cluster ∈ DB do
    sim ← similarity(q, cluster.getCenter())
    if sim < CLUSTER_SIMILARITY_THRESHOLD then
      Search(q, cluster)
    end if
  end for
else
  for each signature ∈ DB do
    sim ← similarity(q, signature)
    if sim < SIGNATURE_SIMILARITY_THRESHOLD then
      write(ID_signature, sim)
    end if
  end for
end if
```

[2] including 0-level & 1-level clustering

3.3 Searching PubChem Compound

PubChem's FTP tool[3] was used to download the most diverse conformer for each molecule in the PubChem Compound database. The SDF directory contained 2,864 sdf files, with each covering a range of 25,000 Compound ID numbers (CIDs). For various reasons, many hundreds of thousands of compounds have been removed from PubChem since its creation, so there are gaps in the CID sequence and each sdf file contains data for less than 25,000 molecules. The 2,864 sdf files, which required 300 GB of storage, were stored on INL's servers.

Shape distributions (signatures) were created for the downloaded molecules. The Euclidean distance between all the unique atomic pairings within a molecule was used to sample the 3-D shape of the molecules. The distances were binned to create a histogram distribution. Each histogram contained 10 bins and each bin had a width of 1.5 units. Distances greater than 15 units were placed in the last bin. The 2,864 sdf files were divided into 16 groups of 179 files and signatures were generated for each group in parallel. This required 3 hours and produced a signature file corresponding to each sdf file.

Searching the signature database with a single peptide required 24 minutes. To reduce the search time, the signatures were clustered using multilevel K-means clustering. During clustering, the χ^2 test was used to assess the distance between signatures. As a result of the clustering, the signatures were divided into 50 clusters, where each of those clusters contained 20 subclusters and each of those clusters contained 5 subsubclusters.

The clustered signature database was queried with the top 200 peptides from the

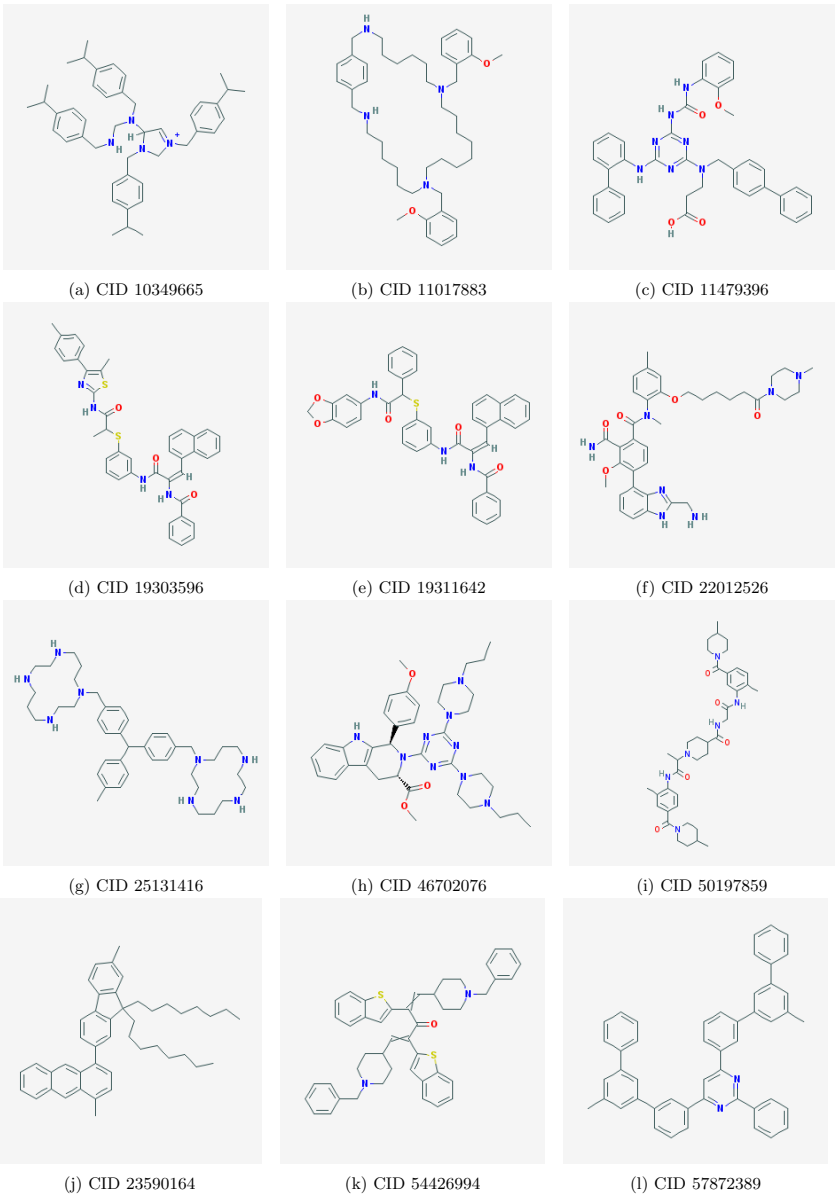
[3]get ftp://ftp.ncbi.nih.gov/pubchem/Compound_3D/01_conf_per_cmpd/SDF/*

GAMPMS described in Section 2.4. When querying the database, the χ^2 test was used to compare the peptide to the center of each of the 50 outermost clusters. The cluster whose center was closest to the peptide, and all clusters whose centers had a similarity score (with the peptide) within .01 of that score, were searched. The same process occurred with the 20 subclusters contained in each searched cluster, and again with the 5 subsubclusters contained in each searched subcluster. When a subsubcluster was searched, the peptide was compared to each signature in that cluster. In this manner, the 100 molecules that were most similar to a peptide were identified in a few seconds with only a few thousand distribution tests.

Duplicate molecules and those containing inorganic elements such as silicon were removed from the 20,000 small molecule collection, leaving 1,320 small molecules. Each of these small molecules was docked against the $\alpha 3\beta 2$ nAChR model using AutoDock with 40 pose evaluations. To identify only distinct scaffolds, the 1,320 molecules were clustered and the molecule with the highest binding affinity with the $\alpha 3\beta 2$ nAChR was selected from each cluster. In this manner, 128 molecules were identified. The 12 highest affinity molecules from the set of 128 are shown in Figure 3.1.

3.4 Usage: Searching PubChem with SimSearcher

The SimSearcher tool has been implemented in DockoMatic 2.1 to perform quick similarity searches over local molecular databases. In this section, SimSearcher is demonstrated by walking through the mapping, clustering, and searching of the PubChem Compound database. The PubChem Compound database, which was already downloaded for the experiment in Section 3.3, was stored in the *PubChemData* folder.



(a) CID 10349665 (b) CID 11017883 (c) CID 11479396

(d) CID 19303596 (e) CID 19311642 (f) CID 22012526

(g) CID 25131416 (h) CID 46702076 (i) CID 50197859

(j) CID 23590164 (k) CID 54426994 (l) CID 57872389

Figure 1: 2 dimensional structures

1

Figure 3.1: Small molecule drugs with predicted high binding affinity for the $\alpha 3\beta 2$-nAChR.

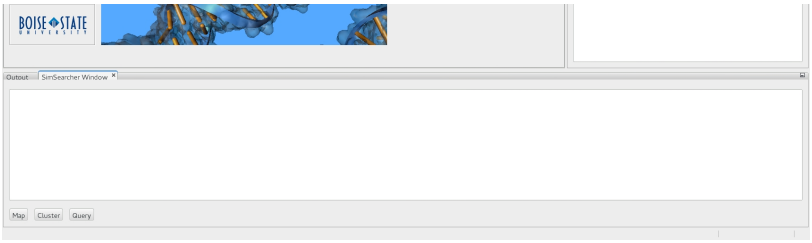


Figure 3.2: The user interface for the SimSearcher program. To run any step of the program (i.e., map, cluster, search), a user clicks the associated button. Output is displayed in the area above the buttons

3.4.1 Map

The Map wizard is used to map the database's molecules to a database of shape signatures. A user clicks the *Map* button at the bottom of the SimSearcher window (see Figure 3.2) to start the wizard. In the Mapping wizard, one must specify the molecular database, the output folder (where to put the signature database), the type of signature to generate, and the submission parameters.

For the demonstration, the *PubChemData* folder is used as the database directory and the *PubChemSignatures* folder as the output directory. Shape distributions are selected for use as the signatures (the pharmacophore signature is discussed in Appendix C) and the default values are used. These values were chosen by building histograms for a subset of the PubChem Compound database with the goal of minimizing the number of empty or almost empty bins. Finally, one tells the program to divide the work across X processes and to run Y processes per node. Once again, the swarm utility is used to submit jobs to the cluster scheduler. Figure 3.3 shows the completed wizard for the example, using $X = 12$ and $Y = 4$. Clicking the *Finish* button will submit the jobs to the scheduler.

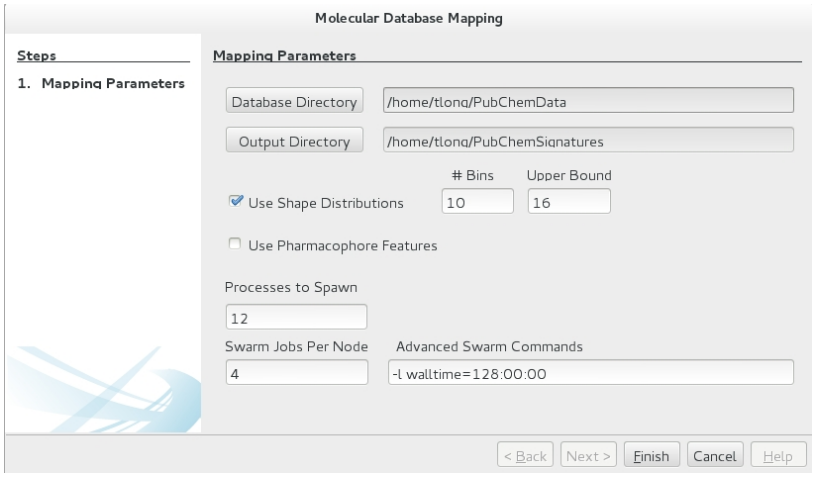


Figure 3.3: The interface for the Map program. A user points the program to the molecular database and specifies the type of signature to generate as well as the submission parameters (swarm parameters and the number of processes to spawn).

3.4.2 Cluster

The *Cluster* button can be clicked, from the SimSearcher window, to load the Cluster wizard, which allows up to 3 levels of clustering to be performed. With the Cluster wizard, one specifies the mapped (signature) directory, the output directory, the distribution test to use when assessing signature similarity (during K-means clustering), the number of clusters to generate at each level, and the submission parameters.

For the demonstration, the *PubChemSignatures* (mapped) database is used and the output is sent to the *ClusteredDB* directory. The χ^2 distribution test is selected to be used when comparing signatures. The number of clusters (the K in K-means clustering) to generate at each level is then entered. To use 2 (or 1) level clustering, one can specify 0 for the number of clusters at level 3 (and 2). Finally, one tells the program to divide the work across X processes and run Y processes per node. Figure

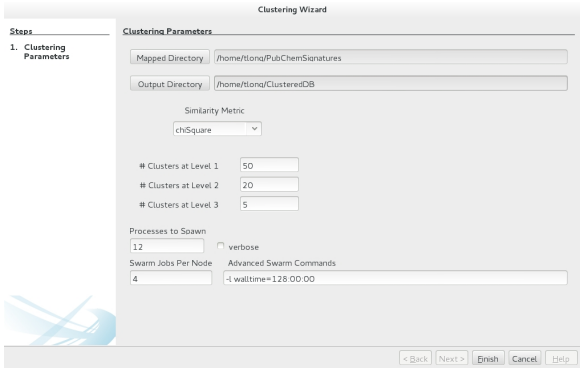


Figure 3.4: The interface for the Cluster program. A user points the program to the signature database and specifies what similarity metric to use, how many clusters to generate at each level, and what parameters to use for submission (swarm parameters and the number of processes to spawn).

3.4 shows the completed wizard from the example, with $X = 12$ and $Y = 4$. Clicking the *Finish* button will submit the clustering jobs to the scheduler.

3.4.3 Search

Searching the molecular database is the end goal of the process. The *Search* button is used to load the Search wizard, which allows a search of a (clustered) signature database. With the Search wizard, one specifies the signature database to be searched, the target molecule (as an sdf file), the similarity test to use, and the number of similar molecules to identify. The search is run on the local machine.

For the demonstration, the *ClusteredDB* is searched for the 10 molecules most similar to the first compound in PubChem 3D (the compound with CID = 1). The χ^2 test is selected to be used to assess signature similarity. The completed wizard for this step is shown in Figure 3.5. Upon clicking *Finish*, DockoMatic 2.1 will parse the target molecule (CID_1) from the sdf file and generate a signature for it. It will

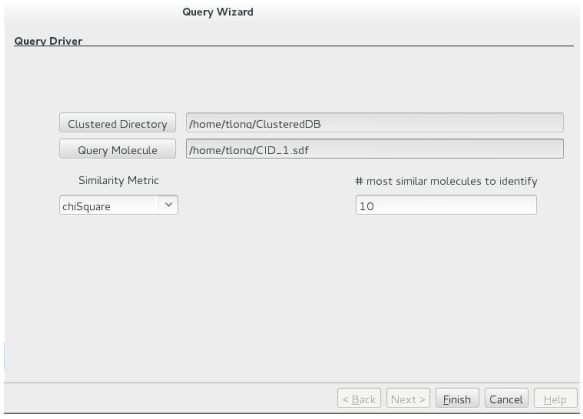


Figure 3.5: The interface for the Search program. A user points the program to a database of molecular signatures and specifies the target molecule and the number of similar molecules to identify.

then use Algorithm 7 to locate the most similar signatures and output the ID and similarity score associated with the 10 most similar signatures.

One can search the database with a set of target molecules by submitting a .txt file as the Query Target. The text file should contain the path to each query target's sdf file, with one path per line.

For comparison, both *PubChemSignatures* and *ClusteredDB* were searched with a single target molecule (CID = 1) for the 10 most similar molecules. The *PubChemSignatures* search took a little more than 24 minutes to complete and performed $\approx$ 51 million similarity calculations. In comparison, the *ClusteredDB* search required a few seconds, performed $\approx 15,000$ similarity calculations, and found the same 10 molecules.

3.5 Conclusion

The development of additional signature types and corresponding similarity metrics could increase SimSearcher's utility. A pharmacophore-based signature and corresponding similarity metric have been developed (see Appendix C) and are included in DockoMatic 2.1, but pharmacophore clustering is not supported. Work remains to be done to improve the speed and test the accuracy of the developed pharmacophore-based similarity search.

Once a molecular database has been mapped to a signature database, the molecular database can be deleted. If a researcher wishes to learn more about a molecule (e.g., the one that was most similar to a target molecule), he or she can use the ID to locate its information from the online database. This prevents researchers from having to allocate hundreds of gigabytes of memory to store redundant molecular information.

In summary, a model has been devised to allow quick similarity searches over local molecular databases with any target molecule. The model consists of 3 steps: Map, Cluster, and Search. In the Map step, signatures are generated for the database's molecules. In the Cluster step, N-level K-means clustering of the signature database is performed to reduce the number of comparisons needed for a search. Each of these steps needs to be performed once per database per signature type. In the Search step, a recursive search algorithm is used to locate signatures similar to the target molecule(s). The ID for the identified molecules are output, allowing the user to search the original web database for additional information on the molecule. The model has been implemented and integrated with DockoMatic 2.1.

CHAPTER 4

CONCLUSIONS

4.1 Overview of Work

CAEPIDR was developed to explore the conformational ligand-binding space of the $\alpha3\beta2$ nAChR isoform and use the results to identify small molecule drugs that target the receptor. A GA-based search procedure (GAMPMS) was used to heuristically explore the ligand-binding domain of the $\alpha3\beta2$ nAChR isoform using a 640,000 α-CTx MII mutant library. The GAMPMS required only 9,344 docking calculations and identified peptides with estimated binding affinities 70% higher than the original α-CTx MII peptide.

In CAEPIDR's repurposing step, the PubChem Compound database was searched for molecules bearing a shape similar to the highest affinity α-CTx MII mutants. To perform the search with small peptides, the database was downloaded and the shape distribution based signatures were generated for each molecule. The signatures were clustered using multilevel K-means clustering and searched with the peptide mutants. The estimated binding affinities of the identified small molecules varied, but the top molecule's predicted affinity was 70% higher than that of the α-CTx MII peptide. Some of the top identified small molecules, which are shown in Figure 3.1, are being purchased from a vendor to provide additional validation for the approach.

CAEPIDR has been generalized and integrated with DockoMatic. DockoMatic

2.1 contains an intuitive graphical interface for a peptide mutant screening workflow, allowing a researcher to quickly create virtual peptide mutant libraries. The user has the option to screen the peptide mutant library exhaustively or with an implementation of GAMPMS. DockoMatic 2.1 also contains the SimSearcher module, which facilitates the mapping, clustering, and searching of local molecular databases. Searching a clustered database with SimSearcher requires a few seconds per target molecule, and a list of target molecules can be submitted to facilitate larger searches. As a result, DockoMatic is a powerful tool for researchers interested in the drug repurposing model.

DockoMatic is an open source software tool and is available for download on sourceforge.net.

4.2 Future Work

At this point, the first 3 steps of the workflow from Section 1.2 had been completed. The 128 identified small molecules are promising candidates for repurposing to target the $\alpha 3\beta 2$-nAChR and treat Parkinson's disease. However, much work remains to be done before any of these drugs can be tested for treating Parkinson's disease. For example, Lipinski's rule of 5, that can be used to remove candidates which have a high chance of failing clinical trails, can be applied to filter the set of molecules. Once a narrowed set is specified, members of that set must be purchased or synthesized and tested in a lab setting to confirm the predictions of the computational methods. However, these procedures are beyond the scope of this thesis; the 128 molecules represent this thesis' main contribution to the project.

The SimSearcher utility can be improved through additional experimentation

to discover better default values for the shape distribution based signatures. Additionally, pharmacophore features can be integrated into the signatures in order to improve SimSearcher's ability to identify molecules with similar bioactivity. A pharmacophore-based metric has been implemented (see Appendix C) and has been tested for effectiveness. Its performance (see Table A.1 from Appendix A) was equivalent to that of the Chi-Square test. Unfortunately, this made it useless as a complementary metric. Work remains to be done to modify the metric so that it is capable of complimenting the shape distribution approach, which would give researchers a better approximation for bioactivity similarity.

REFERENCES

[1] Therapeutic Development Process.(2014). Retrieved, August 6, 2014 from http://www.ncats.nih.gov/research/reengineering/process.html

[2] DiMasi, J.A., Hansen, R.W., Grabowski, H.G. "The price of innovation: new estimates of drug development costs." Journal of Health Economics 22 (2003) 151-185.

[3] Jin, G., Wong, T.C. "Toward better drug repositioning: prioritizing and integrating existing methods into efficient pipelines." Drug Discovery Today, Volume 00, Number 00, January 2014.

[4] Bolton, E., Wang, Y., Thiessen, P.A., Bryant, S.H. "PubChem: Integrated Platform of Small Molecules and Biological Activities." Chapter 12 IN Annual Reports in Computational Chemistry, Volume 4, American Chemical Society, Washington, DC, 2008 Apr.

[5] Morris, G.M., Goodsell, D.S., Halliday, R.S., Huey, R., Hart, W.E., et al. (1998) Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. J Comput Chem 19: 1639.

[6] Goodsell, D.S., Olson, A.J. (1990) Automated docking of substrates to proteins by simulated annealing. Proteins 8: 195-202.

[7] Sousa, Fernandes & Ramos (2006) Protein-Ligand Docking: Current Status and Future Challenges Proteins, 65:15-26

[8] Hu, Z., Southerland, W. (2007) "WinDock: structure-based drug discovery on Windowsbased PCs". J Comput Chem 28: 2347-2351.

[9] Vaqu, M., Arola, A., Aliagas, C., Pujadas, G. (2006) "BDT: an easy-to-use front-end application for automation of massive docking tasks and complex docking strategies with AutoDock". Bioinformatics 22: 1803-1804.

[10] Friesner, R.A., Banks, J.L., Murphy, R.B., Halgren, T.A., Klicic, J.J., et al. (2004) "Glide: A New Approach for Rapid, Accurate Docking and Scoring Method and Assessment of Docking Accuracy". J Med Chem 47: 1739.

[11] Bullock, C.W., Jacob, R.B., McDougal, O.M., Hampikian, G., Andersen, T. (2010) Dockomatic - automated ligand creation and docking. BMC Research Notes 3.

[12] Jacob, R.B., Bullock, C.W., Andersen, T., McDougal, O.M. (2011) "DockoMatic: Automated peptide analog creation for high throughput virtual screening". J Comput Chem 32: 2936- 2941.

[13] The Open Babel Package. (2011) Version 2.2.3. Available: http://www.openbabel.org. Accessed: December 2011.

[14] Xu, J. Research in Computational Molecular Biology. Miyano S, Mesirov J, Kasif S, Istrail S, Pevzner P, Waterman M, editors. Springer Berlin; Heidelberg: 2005. pp. 423-439.

[15] Xu, J., Berger B. J ACM. 2006;53(4):533-557.

[16] Fratiglioni, L. and Wang, H.X. "Smoking and Parkinson's and Alzheimer's disease: review of the epidemiological studies." Beh Brain Res 113(1-2), 117-120 (2000) doi:10.1016/S0166-4328(00)00206-0

[17] Morens, D.M., Davis, J.W., Grandinetti, A., Ross, G.W., Popper, J.S., and White, L.R. "Epidemiologic observations on Parkinson's disease: incidence and mortality in a prospective study of middle-aged men." Neurology 46(4), 1044-1050 (1996)

[18] Allam, M.F., Campbell, M.J., Hofman, A., Del Castillo, A.S., Fernandez-Crehuet Navajas, R. "Smoking and Parkinson's disease: systematic review of prospective studies." Movement Disorders 19(6), 614-621 (2004)

[19] Perry, E.K., Martin-Ruiz, C.M., Court, J.A. "Nicotinic receptor subtypes in human brain related to aging and dementia". (Review) (52 refs). Alcohol 24(2), 63-68 (2001)

[20] Levin, E.D., McClernon, F.J., Rezvani, A.H. "Nicotinic effects on cognitive function: behavioral characterization, pharmacological specification, and anatomic localization". Psychopharmacology 184(3-4), 523-539 (2006)

[21] Picciotto, M.R, Zoli, M. "Neuroprotection via nAChRs: the role of nAChRs in neurodegenerative disorders such as Alzheimer's and Parkinson's disease". Frontiers in Bioscience 13, 492-504, January 1, 2008

[22] Cartier, G.E., Yoshikami, D., Gray, W.R., Luo, S., Olivera, B.M., McIntosh, J.M. (1996) "A new -conotoxin which targets $\alpha 3\beta 2$ nicotinic acetylcholine receptors." J Biol Chem 271:7522-7528

[23] McIntosh, J.M., Azam, L., Staheli, S., Dowell, C., Lindstrom. J.M., Kuryatov, A., Garrett, J.E., Marks, M.J., Whiteaker, P. "Analogs of alpha-conotoxin MII are selective for alpha6-containing nicotinic acetylcholine receptors". Mol Pharmacol. 2004 Apr;65(4):944-52.

[24] Salminen, O., Drapeau, J.A, McIntosh, J.M., Collins, A.C., Marks, M.J., Grady, S.R. "Pharmacology of -Conotoxin MII-Sensitive Subtypes of Nicotinic Acetylcholine Receptors Isolated by Breeding of Null Mutant Mice". Molecular Pharmacology June 2007 vol. 71 no. 6 1563-1571

[25] Whiteaker P., McIntosh J.M., Luo S., Collins A.C., Marks M.J. "125I-alpha-conotoxin MII identifies a novel nicotinic acetylcholine receptor population in mouse brain." Mol Pharmacol. 2000 May;57(5):913-25.

[26] Bordia, T., Grady, S.R., McIntosh, J.M., Quik, M. (2007) "Nigrostriatal damage preferentially decreases a subpopulation of $\alpha 6\beta 2$ nAChRs in mouse, monkey, and Parkinson's disease striatum." Mol Pharmacol 72:52-61

[27] Somisetti V.S, Roberts, J., Bharadwaj, V.S., Slingsby, J.G., Rohleder, C., Mallory, C., Groome, J.R., McDougal, O.M., Maupin, C.M. ChemBioChem, "Acetycholine Promotes Binding of alpha-Conotoxin MII for a3b2 Nicotinic Acetylcholine Receptors" 15, 413-424 (2014).

[28] Plewczynski, D. "Virtual High Throughput Screening Using Combined Random Forest and Flexible Docking". Combinatorial chemistry & High Throughput Screening, 2009, (1386-2073), 12 (5), p. 484.

[29] Svetnik, V., Liaw, A., Tong, C., Culberson, J.C., Sheridan, R.P., Feuston, B.P. J. Chem. Inf. Comput. Sci., 2003, 43, 1947-1958.

[30] Svetnik, V., Liaw, A., Tong, C., Wang, T. Multiple Classifier Systems; Proceedings 2004; pp. 334-343.

[31] Cherkasov, A., Ban, F.Q., Li, Y., Fallahi, M., Hammond, G.L. J. Med. Chem., 2006, 49, 7466-7478.

[32] Cherkasov, A., Curr. Comput-Aided Drug Des., 2005, 1, 21-42.

[33] Ma, X.H. "Comparative Analysis of Machine Learning Methods in Ligand-Based Virtual Screening of Large Compound Libraries". Combinatorial chemistry & High Throughput Screening, 2009, (1386-2073), 12 (4), p. 344.

[34] Melville, J.L., Burke, E.K., Hirst, J.D. "Machine Learning in Virtual Screening". Combinatorial Chemistry & High Throughput Screening, 2009, I2, 332-343.

[35] Sheridan, P., Robert and Kearsley, K. Simon. "Using a Genetic Algorithm to Suggest Combinatorial Libraries". 1994. J. Chem. Inf. Comput. Sci, 1995, 35, 310-320.

[36] Sheridan, P., Robert, S.F., Sonia, P., Kearsley, K.S. "Designing Targeted Libraries with Genetic Algorithms". 2000. Journal of Molecular Graphics and Modelling 2000, 18, 320-334.

[37] Gillet, V.J., Willett, P., Bradshaw, J. "Selecting combinatorial libraries to optimize diversity and physical properties". J. Chem. Inf. Comput. Sci. 1999, 39, 167-177.

[38] Valler, M.J., Green, D. "Diversity screening versus focused screening in drug discovery". Drug Discovery Today 2000, 5, 286-293.

[39] Martin, E.J., Crichlow, R.W. "Beyond mere diversity: tailoring combinatorial libraries for drug discovery". J. Comb. Chem. 1999, 1, 32-45.

[40] Fonseca, C.M., Fleming, P.J. "An overview of evolutionary algorithms in multiobjective optimization". In EVolutionary Computation; De Jong, K., Ed.; Massachusetts Institute of Technology: Cambridge, 1995; Vol. 3, No. 1, pp 1-16.

[41] Gillet, V.J., Khatib, W., Willett, P., Fleming, P.J., Green, D.V.S. "Combinatorial Library Design Using a Multiobjective Genetic Algorithm". 2001. J. Chem. Inf. Comput. Sci. 2002, 42, 375-385.

[42] Gillet V.J., Willett P., Fleming P.J., Green D.V. "Designing focused libraries using MoSELECT". J Mol Graph Model. 2002 Jun;20(6):491-8.

[43] Fontaine F., Bolton E., Borodina Y., Bryant S.H. "Fast 3D shape screening of large chemical databases through alignment-recycling". Chem Cent J. 2007;1:12. doi: 10.1186/1752-153X-1-12

[44] Inokuchi, A., Washio, T., Nishimura, K., Motoda, H. "A Fast Algorithm for Mining Frequent Connected Subgraphs". IBM Research, Tokyo Research Laboratory, 2002.

[45] Cao, Y., Jiang, T., Girke, T., "A maximum common substructure-based algorithm for searching and predicting drug-like compounds." Bioinformatics (2008) 24 (13): 366-374

[46] Ralaivolaa, L., Swamidassa, S.J., Saigoa, H., Baldi, P. "Graph kernels for chemical informatics". Neural Networks,Volume 18, Issue 8, October 2005, Pages 1093-1110

[47] Friehlich, H., Wegner, J.K., Zell, A. “Assignment Kernels For Chemical Compounds”. International Joint Conference on Neural Networks 2005 (IJCNN’05), 2005, 913-918.

[48] Freyhult, E.K., Andersson, K., Gustafsson, M.G. (April 2003). ”Structural Modeling Extends QSAR Analysis of Antibody-Lysozyme Interactions to 3D-QSAR”. Biophysical Journal 84 (4): 2264-72.

[49] Dearden, J.C. (2003). “In silico prediction of drug toxicity.” Journal of Computer-aided Molecular Design 17 (2-4): 119-27.

[50] Osada, R., Funkhouser, T., Chazelle, B., DobKin, D. “Shape Distributions.” ACM Transactions on Graphics, Vol. 21, No. 4, October 2002, Pages 807-832.

[51] Zhou, Y., Zhang, K., Ma, Y. “3D protein structure similarity comparison using a shape distribution method”. Information Technology and Applications in Biomedicine, 2008. May 2008, 233-236.

[52] Bolton, E., Kim, S., Bryant, S., “PubChem3D: Similar conformers”. Journal of Cheminformatics, 2011. 3:13.

APPENDIX A

DISTRIBUTION TESTS

The calcDistTest() function (below) was used to assess the similarity of shape-based signatures. Since each signature was a fixed-size histogram, signatures were represented as an array of doubles. The calcDistTest() function was built to work with one of four metrics, and the implementations of those metrics follow the code for calcDistTest().

An experiment was performed to compare the performance of the shape distribution approach, with each of the 4 distribution tests, to the results reported by Bolton et al. on a 3-D conformer search of PubChem Compound [52]. To set up the experiment, a set of 16 molecules was selected from the PubChem Compound database. 8 of the molecules, set Pos, were ligands of prostaglandin synthase and therefore had established similar bioactivity. These were the ligands reported on by Bolton et al. While shape Tanimotos have an established threshold for similarity (.8 - .85), our approach did not. As a result, we also included a set of 8 molecules, Neg, which were selected to be dissimilar to each other. Its use is explained below.

The pairwise similarity between all 16 molecules was assessed using each of the 4 implemented distribution tests. A threshold was used so that the tests could act as classifiers. That is, if $test(mol_a, mol_b) > THRESHOLD$, then mol_a would be considered similar to mol_b. The ideal classification for the 16 molecules is given by

Equation A.1. The actual classification for all 5 tests (the 4 implemented tests as well as the results reported by Bolton et al. for the Shape Tanimoto) are given in Table A.1. In the experiments, the $THRESHOLD$ was set so that there was no more than 1 false positive.

$$test(x, y) = \begin{cases} similar & \text{if } x \in Pos \,\&\, y \in Pos \\ not\, similar & \text{if } x \in Neg \,\&\, y \in Neg \end{cases} \tag{A.1}$$

Test	**True Positives**	**False Positives**
chiSquare	18	1
pdfL1	14	1
pdfL1	14	1
rootOfProduct	20	1
Bolton et al.	14	NA

Table A.1: Distribution Test results.

As can be seen, the shape distribution approach compared favorably to the techniques used by PubChem 3D.

```
public static double calcDistTest(double[] dist1, double[] dist2, String dtest) {
    int x;
    double sum = 0.0;

    if (dist1.length != dist2.length){
        sum = 0.0;
    } else if (dtest.equals("chiSquare")) {
        for (x = 0; x < dist1.length; x++){
            sum += chiSquare(dist1[x], dist2[x]);
        }
        sum /= 2.0;
    } else if (dtest.equals("pdfL1")) {
        for (x = 0; x < dist1.length; x++){
            sum += pdfL1(dist1[x], dist2[x]);
        }
        sum /= 2.0;
    } else if (dtest.equals("pdfL2")) {
        for (x = 0; x < dist1.length; x++){
            sum += pdfL2(dist1[x], dist2[x]);
        }
        sum = Math.sqrt(sum);
    } else if (dtest.equals("rootOfProduct")) {
        for (x = 0; x < dist1.length; x++){
            sum += rootOfProduct(dist1[x], dist2[x]);
        }
        sum = 1 - sum;
    }
    return sum;
}
```

Figure A.1: Function to compare distributions using the test arguments

```
public static double chiSquare(double a, double b) {
    double result = 0.0;
    if ((a + b) != 0.0){
        result = Math.pow(a - b,2) / (a + b);
    }
    return result;
}

public static double pdfL1(double a, double b) {
    return Math.abs(a-b);
}

public static double pdfL2(double a, double b) {
    return Math.pow(a - b,2);
}

public static double rootOfProduct(double a, double b) {
    return Math.sqrt(a * b);
}
```

Figure A.2: Code for distribution tests

APPENDIX B

K-MEANS CLUSTERING

The SimSearcher module uses N-level, K-means clustering to reduce the time required for database searches. K-means clustering is an iterative clustering algorithm that partitions a set of data points by placing K center points and adjusting their locations until an optimal arrangement is found. An arrangement is considered optimal if the sum of the distances between each data point and the cluster center closest to it is minimized. However, our version of K-means clustering iterates for a set number of cycles before stopping. At each iteration, a cluster center is moved to the average coordinates of all the points that were in the cluster during the previous round.

B.1 In-Memory Clustering

When all the data points can be placed into main memory, algorithm 6 can be used to perform N-level, K-means clustering by using Algorithm 8 as $KMeans\_Cluster(DB)$.

B.2 Big Data Style

When all the data points cannot be placed into main memory, a different algorithm needs to be used. In this case, algorithm 9 needs to be used. This algorithm assumes a parallel computing infrastructure, and the parameter *cores* is used to specify the number of processes to spawn during each iteration.

Algorithm 8 In Memory $KMeans\_Cluster(DB, K)$

```
clusters ← choose K random points as cluster centers
for i ∈ [0, X] do
  for each point ∈ DB do
    distance ← MAX_VALUE
    for each cluster ∈ clusters do
      tmp ← getDistanceBetween(cluster.center, point)
      if tmp < distance then
        closest ← cluster
        distance ← tmp
      end if
    end for
    closest.addPoint(point)
  end for
  for each cluster ∈ clusters do
    distance ← MAX_VALUE
    for each point ∈ cluster.getPoints() do
      run_total+ = point.coordinates
    end for
    cluster.setCenter(run_total/cluster.points.size)
  end for
end for
```

Algorithm 9 Big Data $KMeans\_Cluster(DB, cores, K)$

```
clusters ← choose K random points as cluster centers
write clusters.getCenters() to centers_file
partitions ← partition(DB, cores)
for i ∈ [0, X) do
  for each partition ∈ partitions do
    spawn worker(partition, false) process
  end for
  wait for all workers to finish
  if i ¡ X then
    for each file ∈ worker totals file do
      for j ∈ [1, K] do
        running_total[j]+ = running_total_cluster[j]
        points_in_cluster[j]+ = count[j]
      end for
    end for
    for j ∈ [1, K] do
      cluster[j].setCenter(running_total[j]/count[j])
    end for
    write clusters.getCenters() to centers_file
  end if
end for
for each partition ∈ partitions do
  spawn worker(partition, true) process
end for
```

Algorithm 10 $worker(partition, final)$

```
clusters ← read from centers_file
for file ∈ partition do
  for each point ∈ file do
    distance ← MAX_VALUE
    for each cluster ∈ clusters do
      tmp ← getDistanceBetween(cluster.center, point)
      if tmp < distance then
        closest ← cluster
        distance ← tmp
      end if
    end for
    if final then
      closest.addPoint(point)
    else
      closest.add_running_total(point.coordinates)
      closest.increment_count()
    end if
  end for
  if final then
    append_points_to_cluster_file(clusters)
    clusters.remove_all_points()
  end if
end for
if ∄final then
  write clusters.running_totals and clusters.counts to worker's totals file
end if
```

APPENDIX C

PHARMACOPHORE-BASED SIMILARITY METRIC

Pharmacophore (feature) based similarity is another popular estimator of bioactivity similarity. As such, a rotationally invariant function was developed to generate a feature-based signature for a molecule, assuming a rigid structure.

A molecule's data file contains information on atom locations, charge, and connectivity, so simple graph traversal techniques can be used to determine the existence of features within a molecule. For the algorithms in this section, we assume that a function $parseFeatures()$ exists that returns the features of the argument molecule.

Each feature has a type (e.g., ring, cation) and a location (set of (x,y,z) coordinates), and the location of a composite entity (e.g., the molecule, a ring) is considered to be the average location of its atoms. The feature-based signature generation function is shown in Algorithm 11. The ID (its CID in the case of PubChem Compound) and its features are parsed from a data file. Next, every feature is cast as a typed vector originating at the molecule's center. Finally, a 4-tuple (see Algorithm 12) is created for every pair of typed vectors in the molecule. The 4-tuple contains the type of both of the typed vectors as well as their magnitude difference and the angle between them.

A custom function was designed to map two feature-based signatures, s_1, s_2 to a number $sim \in [0, 1]$. Before describing the function, we note that a signature is a

Algorithm 11 map($molecule$)

$ID \leftarrow molecule.getID()$
$features \leftarrow parseFeatures(molecule)$
for each $feature \in features$ **do**
  $feature.coordinates \leftarrow feature.coordinates - molecule.getCenter()$
end for
for each $feat_i \in features$ **do**
  for each $feat_j \in features$ **do**
    if $i \neq j$ **then**
      if $NOT\ sampledList.contains((i,j))$ **then**
        $FeaturePairs.add(generateFeaturePair(feat_i, feat_j))$
        $sampledList.add((i,j),(j,i))$
      end if
    end if
  end for
end for
return $ID, FeaturePairs$

Algorithm 12 $generateFeaturePair(vector_1, vector_2)$

$type_1 \leftarrow vector_1.getType()$
$type_2 \leftarrow vector_2.getType()$
$\Delta \leftarrow L2Norm(vector_1, vector_2)$
$\theta \leftarrow \arccos \frac{vector_1 \cdot vector_2}{|vector_1||vector_2|}$
return $< type_1, type_2, \Delta, \theta >$

collection of FeaturePairs. Now, let μ and ρ be any positive integer. The metric is:

$$h(s_1, s_2) = \frac{\sum_{\forall x \in s_1} f(x, s_2)}{|s_2|\mu} \tag{C.1}$$

where

$$f(x, s_2) = MAX(g(x, y)) \tag{C.2}$$

and where $y \in s_2$. We also note that each 4-tuple in s_2 is used at most once at the in calculating the similarity between a 4-tuple:

$$g(x, y) = \begin{cases} 0 & \text{if } types\, don't\, match \\ \frac{\mu\rho - |\theta_x - \theta_y|}{\rho}(1 - |\Delta_x - \Delta_y|) & \text{otherwise} \end{cases} \tag{C.3}$$

In other words, the function computes the similarity between two signatures by summing the similarity between the most similar FeaturePairs in each signature. FeaturePair similarity is 0 if the feature types differ; otherwise, it is inversely proportional to the difference in the angles and magnitudes.

The technique has the following shortcomings:

1. There is no defined average value for a collection of pharmacophore signatures, so K-means clustering is unusable and queries require too many similarity computations.

2. The signature generation assumes a rigid structure, which is inaccurate.

3. The method seems to classify small molecules in a manner similar to shape-based similarity, so the methods do not complement each other well.