Virtual Screening of Combinatorially Modified PDZ-Domain Inhibiting Peptides Synopsis

Introduction

The initial goal of this project is to generate a set of novel peptides which are identified by the computational software as having a higher binding affinity to CAL than the original peptide. These peptides are then to be empirically tested in binding studies in order to validate the efficacy of the computational approach. Due to the multiple steps involved in this process, I chose to use the Schrödinger Drug Discovery suite since it contained a comprehensive set of molecular modeling and docking tools. Additionally, they recently released specific support for peptide docking, which is rare among open source docking software algorithms.

During the fall term I read the Schrödinger background theory and user manual and became familiar with the various software components and python API. I identified KNIME (https://www.knime.org)—an open source data mining tool included with the Schrödinger along with many plugins (nodes)—as the best way to manage this process. Maestro, Schrödinger's molecular visualization software, is used to view docking results and can also be used as a control center to ran various Schrödinger scripts. Below is a description of each software component used in the workflow for virtual screening of CAL-binding peptides.

Software

KNIME (https://www.knime.org): This software allows for visual control of data flow and integration of various pieces of Schrödinger software without having to use bash or python

scripts. It also automatically supports asynchronous execution of each node in the workflow, and distribution of a particular node's work over multiple cores; this would be virtually impossible to do in bash and significant work in python (scaling to distribute work over variable number of cores is not trivial). This software has been extremely intuitive to setup, it efficiently handles the hardest parts of writing computational software, and is very flexible with easily implemented python nodes loaded with Schrödinger libraries by default. This makes it extremely appealing for continued use—the future students who may pick up this project. KNIME also has a number of commercial extensions to their product, one of which is for easily distributing work over a computer cluster (described later).

Maestro (https://www.schrodinger.com/maestro): High performance molecular visualization and which integrates with rest of Schrödinger software. The python API is easily accessible for quick commands and all the software scripts below can manually be run through Maestro with a GUI— some of them are much easier to do this way.

CombiGlide (https://www.schrodinger.com/combiglide):

1. **CombiGlide Reagent Preparation**: Run reagent preparation script with (currently 550) pre-filtered carboxylic acids. The core molecule is ANSRWPTS[Ac-K]I from the previous study (known to bind CAL). This script is configured to find the carboxylate group on each reagent and clip the molecule at the C-C bond, designated the attachment point. Acids with two of such groups are ignored. Essentially, the process takes just the "R" group of each acid which will replace the terminal methyl group on the Ac-K residue of the core molecule in a later step. This step also

generates energetically favorable conformers for each reagent (if more than one exist).

2. **CombiGlide-KNIME-Maestro Connector**: Script to desiginate the attachment point(s) on the core molecule and reagents to couple at each point. Currently this script allows picking of a single attachment point / reagent library, and is launched in a GUI to manually pick the bond to be broken (currently, the N-CH_3 bond on the terminal amide of Ac-K is always broken). Eventually I would like to parameterize it with an input argument. For example, when running this workflow, pass in arguments {[Res1, Res2, Res3, …] and [ReagentLib1, ReagentLib2, ReagentLib3, …]} which defines multiple attachment points and corresponding reagent library. This adds additional complexity—only specific residues can be coupled with specific reagents and some residues may have multiple coupling locations. For the proof of concept this feature is unnecessary and can be addressed later if time permits.
3. **CombiGlide Reagent Enumeration**: Run script with outputs from #1 and #2. Iterates through entire library and creates the attachment bond between the Acetyl-K α-Carbon and the reagent—R group of organic acid. Output is a set of modified peptides structures, ready to be docked.

Glide (https://www.schrodinger.com/glide):

1. **Glide Receptor Grid Generation**: Run grid generation script with CAL protein complexed with ANSRWPTS[AC-K]I ligand to identify the binding site. Additionally specify 9 required H-bonds (see attached image) and one positional occupation

requirement. These constraints can be relaxed in the future if we feel it would yield better results. I favored a highly constrained system (10 constraints in the maximum) initially in order to keep most of the ligand in the same conformation (which we know binds), as well as to reduce computational expense. The grid generation also allows designating whether OH/SH R groups in CAL Ser, Thr, Tyr, and Cys residues allow rotation. In order to minimize computation time, I allowed rotation in residues close to the active site and disallowed it in those distal to it. This grid was also generated particularly for peptide docking—Glide somehow optimizes for field calculations for docking amino acid chains (they do not explain how). This script can be run through KNIME, but its difficult to parameterize all the constraints; since it only has to be done once per receptor protein I did this by hand once. The CAL crystal structure was also a dimer with two complexed copies of the peptide. I chose the peptide binding between both chains (rather than only binding to one) as the grid generation location.

2. **Glide Ligand Docking**: This script is the heart of entire workflow: it runs docking software on each ligand from #3, using the receptor grid from #4. There are three levels of resolution, HTVS (high throughput virtual screening), SP (standard precision), and XP (extra precision). Each level significantly increases the accuracy of the scoring calculation due to: finer grained energy calculation "grid" in 3D space, slightly more complex force field model, and more extensive search through ligand's conformational space. I chose to dock the ligands with the "flexible" strategy as opposed to "rigid" because it was recommended for peptide docking. Currently the

workflow works (see image below) by running the all ligands through HTVS and filtering out all compounds with docking scores lower than the unmodified peptide—more negative value indicates higher binding affinity. The remaining targets are run through SP (again against the unmodified peptide) and filtered again. Finally, the remaining ligands are docked with XP and filtered compared to the unmodified peptide. I ran HTVS on my computer with the 686 ligands (more than the initial number of acids because of multiple conformers) and 155 of them scored less than -7.26 (core molecule docking score). The SP is not yet finished running on my computer (two days and counting), and licensing limitations have prevented from running it on the computer cluster—discussed later.

Protein Preparation Wizard: Simple python script (and GUI) to prepare protein for computational simulation. Adds missing hydrogens, any missing heavy chain atoms (known from sequence but not structure), removes irrelevant or incorrectly placed water molecules, performs slight energy minimizations (with the protein highly constrained), and more. Can be done as a node in the workflow with inputs, but easier to do once per receptor in Maestro and save interactively.

MAE Converter: Simple python script to convert SDF files (acid library) or PDB files (protein / ligand) into a maestro format to be compatible for other scripts.

Computer Cluster

I have setup and installed the Schrodinger software on my account on the Discover computer cluster. Unfortunately, the trial license was already expired—it started on the day issued in December. I have still been able to use a VPN to connect to UConn license on my own computer, but for a number of reasons it is not possible to do this on the research cluster. Most problematic is that use of a VPN is highly discouraged and requires administrative permission to install. I reached out to the computing team about the possibility of installing Glide and CombiGlide on the cluster (they currently only have Schrödinger's Jaguar), but but have not received a response yet. It is unlikely they will be able to get a cluster-wide license (pretty expensive I imagine), but we do have the ability to install licenses on just our account. However, Schrödinger licenses for multi-node computers are still expensive; each Glide jobs costs five "tokens". Based on the email from you with their pricing (16 Tokens - $4,800 or 25 Tokens - $7,500), we would only be able to run 3-5 concurrent jobs at once (one job per core). My computer has four cores (albeit not quite as fast as the cluster, but reasonably close), and a our free Discovery account gives us access to 8 cores. Thus, with a $7,500 license we could not even utilize the full power of our free Discovery account. On the other hand, for $4,900 we could invest in the Discovery cluster and own a single node (16 cores) with access to up to 64 cores when available (cluster has not hit capacity in months). In light of this, the Schrödinger software seems extremely expensive in comparison: investing in the cluster would give enormous computing power resources for the next 4 years for less than one

year of Schrödinger subscription. The UConn license I am connected to has essentially unlimited (200,000) tokens available for use at any time, and there are other open source docking softwares which may yield comparable results to Glide. In light of these problems, I have been playing numerous alternative approaches to getting this running.

One solution is to use Schrödinger to generate ligand libraries locally (not too computationally expensive), export the structures to the computer cluster, and have a simple python script feed each ligand into AutoDock (built into Discovery). This is the cheapest solution, but definitely requires the most work and is the least flexible in terms of future development—we cannot utilize KNIME to control data flow or ever use the awesome features of CombiGlide's multi-attachment point docking algorithms which could be very beneficial in further refining leads down the road (as well as QSite (mixed MM/QM) or CoreHopping software. Additionally AutoDock has generally been considered less accurate for peptide-binding scoring than Glide, does not easily allow for the filtering and increasing precision of docking.

A second solution, as mentioned before, would be to just purchase a Schrödinger license and install it just on our account (or convince Discovery team to add Glide for everyone). This would be reasonably expensive, but would very easily solve our problem. If we had an active license, everything else is setup so that I could begin running the workflow immediately.

A final solution that I have explored is using KNIME cluster executor to run the workflow across the cluster. This summarizing description is what interested me: "KNIME Cluster Executor simplifies interaction with the cluster and reduces costs by moving compute-intensive or license-restricted computations from the data scientist's machine to the distributed computing environment." From my understanding, this means that we can install KNIME software on the cluster which will execute tasks from the master computer with the licensed software (my computer). Thus, we could theoretically connect to UConn license and run up to 40,000 Glide jobs from my computer with all of the computational "work" distributed over the Discovery cluster (without actually installing/licensing Schrödinger on the cluster). I am in contact with KNIME about whether I could utilize their product this way and what the cost would. If we ended up taking this approach, we could put the savings from Schrödinger license fees towards investing in Discovery and be able to run up to 64 jobs simultaneous, assuming availability of cores.

Conclusion And Next Steps

Below I attached the top ten scoring peptides generated by HTVS. Based on visual inspection, many of them look promising with favorable intermolecular interactions away between the added acid and the receptor protein (away from the active site). Some of the halogenated acids in particular have multiple halogen bonds and pi-cation interactions, as shown

in the image below. Some of leads have nitrogens conjugated in a pi-system as well as ethers; I don't know if this impacts the synthetic viability/ease.

Going forward, I think we should quickly pick an approach to get this computation workflow up and running on the cluster. If I get any updates from the people at Discovery or KNIME I will keep you updated. Additionally, I think we should really expand the size of the acid library we are using once we get the cluster running. I think ~600 acids is far too small a library to have a high probability of getting a good lead. I've been in contact with Enamine and they have much larger libraries which I think we should utilize. Before doing that we need to develop a strategy to filter the library, or see if they can filter it for us.

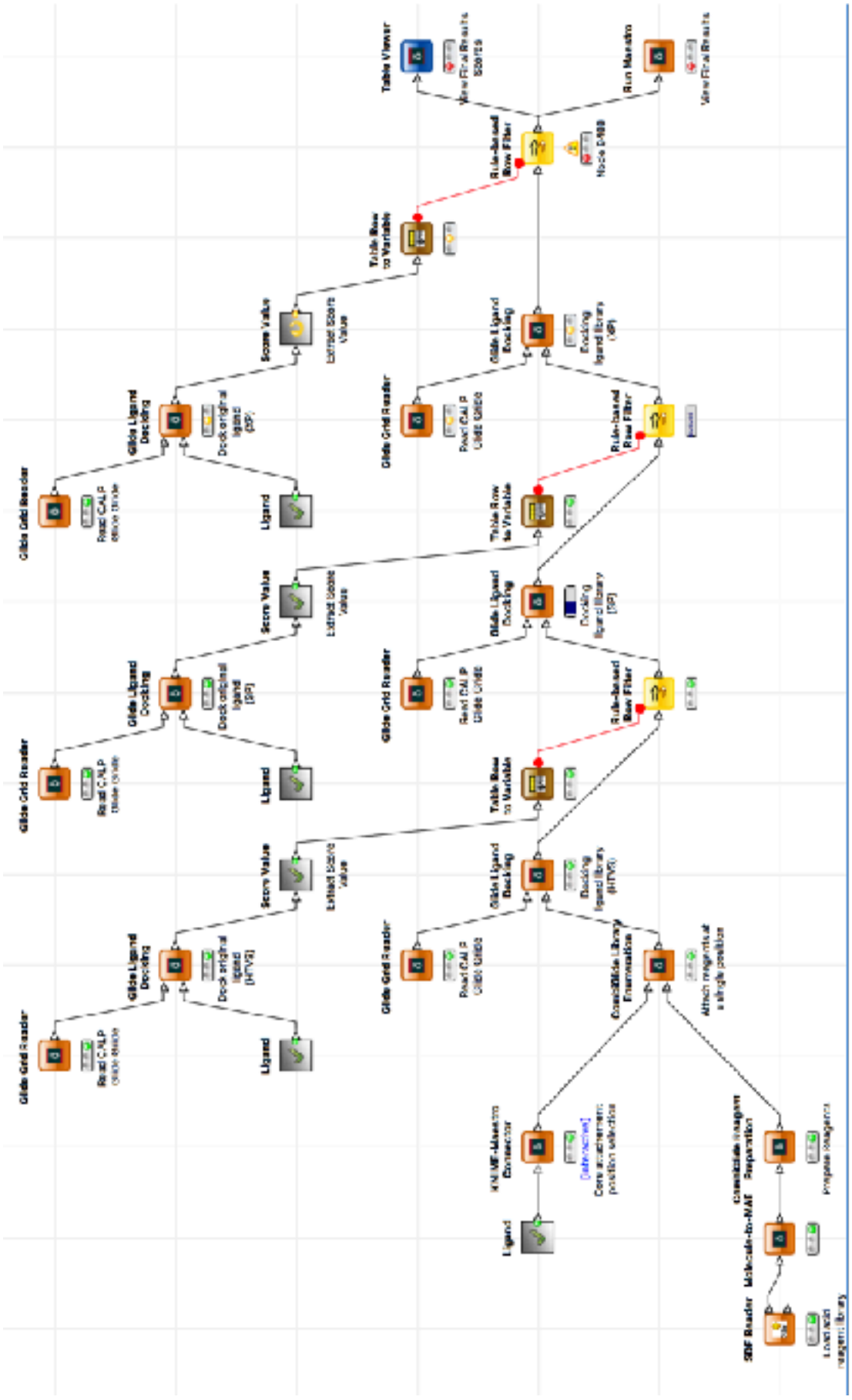
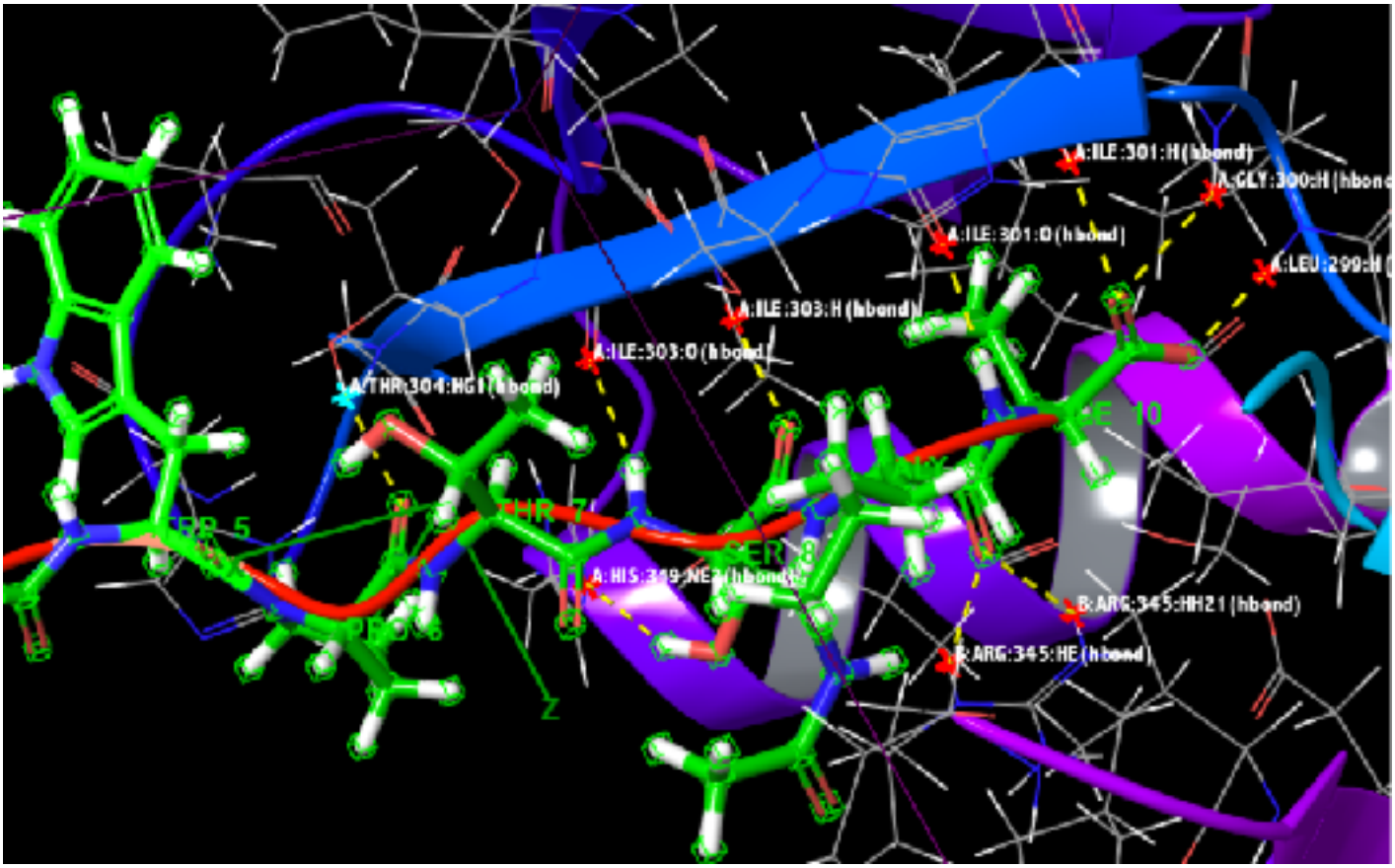


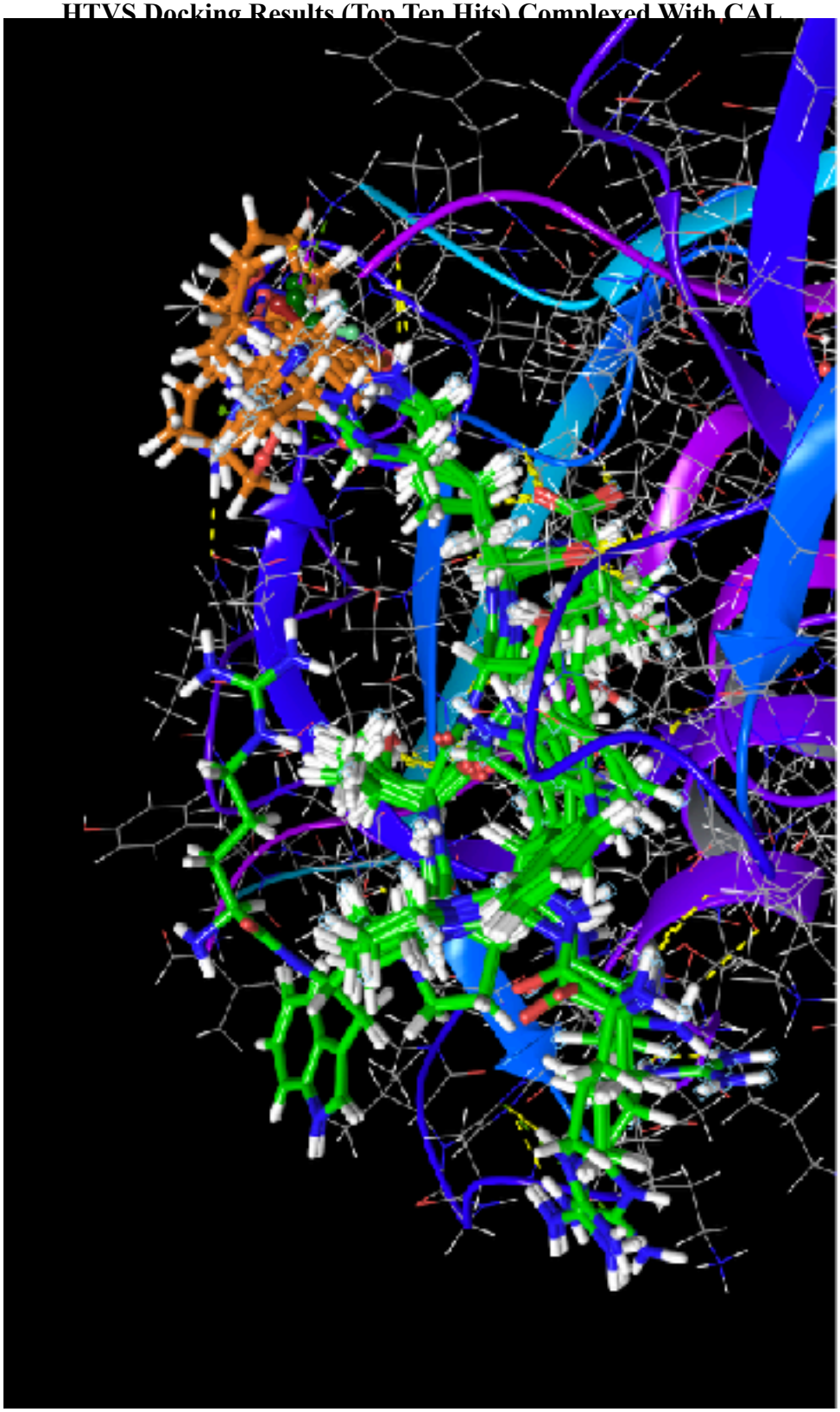
Table Viewer
View Final Results
Run Maestro
View Final Results
Rule-based Row Filter
Table Row to Variable
Score Value
Extract Score Value
Glide Ligand Docking
Dock original ligand (SP)
Glide Grid Reader
Read GALP
Ligand
Glide Ligand Docking
Docking ligand library (SP)
Glide Grid Reader
Read GALP
Rule-based Row Filter
Table Row to Variable
Score Value
Extract Score Value
Glide Ligand Docking
Dock original ligand (SP)
Glide Grid Reader
Read GALP
Ligand
Glide Ligand Docking
Docking ligand library (SP)
Glide Grid Reader
Read GALP
Rule-based Row Filter
Table Row to Variable
Score Value
Extract Score Value
Glide Ligand Docking
Dock original ligand (HTVS)
Glide Grid Reader
Read GALP
Ligand
Glide Ligand Docking
Docking ligand library (HTVS)
Glide Grid Reader
Read GALP
Core2Glide Library Enumeration
Attach reagents at a single position
Core enumeration: position selection
Ligand
SDF Reader
Molecule-to-NAME
Concatenate reagent lists
Prepare reagents

KNIME Workflow



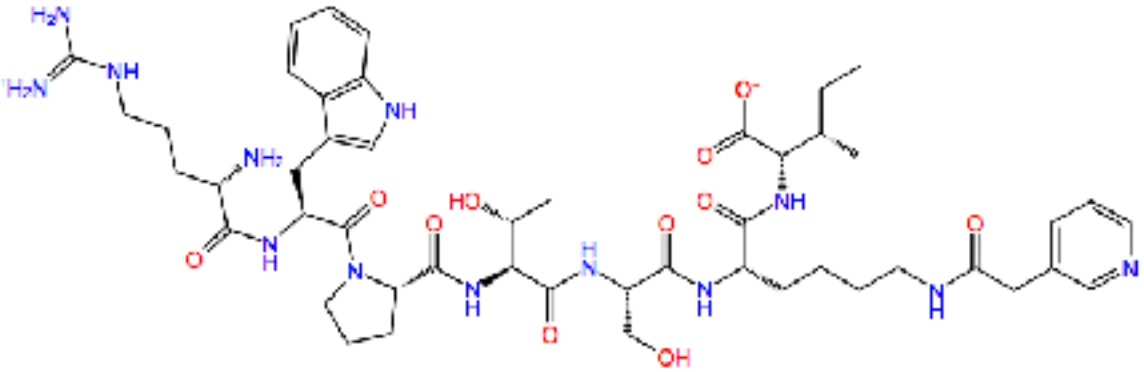
Glide Grid Generation (H-Bond Constraints)

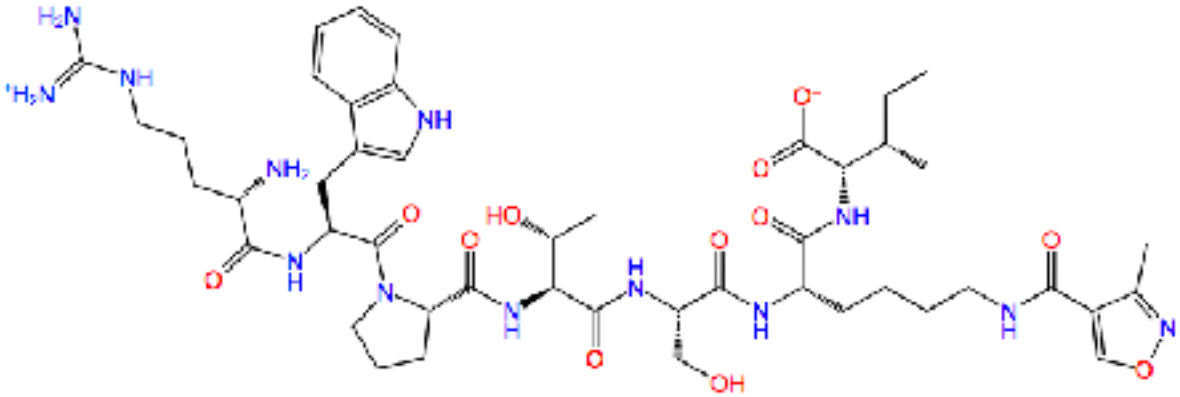
HTVS Docking Results (Top Ten Hits) Complexed With CAL

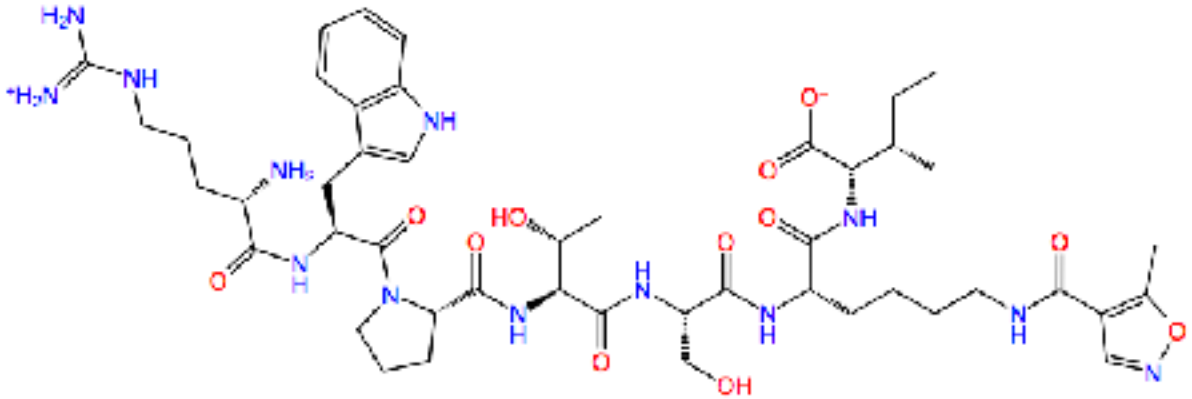


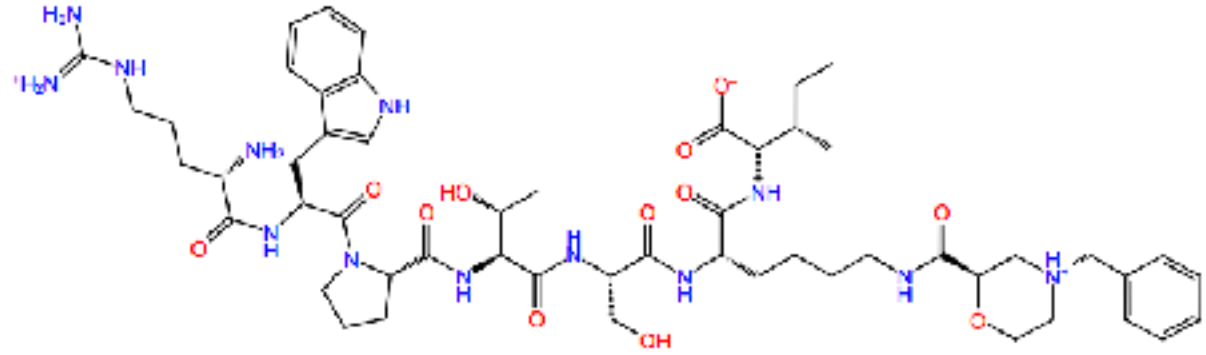
HTVS Docking Results (Top Ten Hits) Ligand 2D Structures

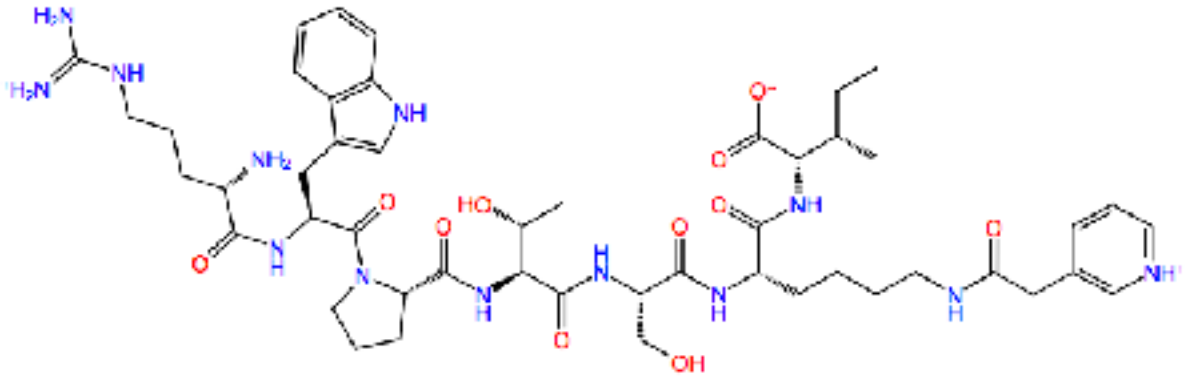
Ordered By Decreasing Affinity

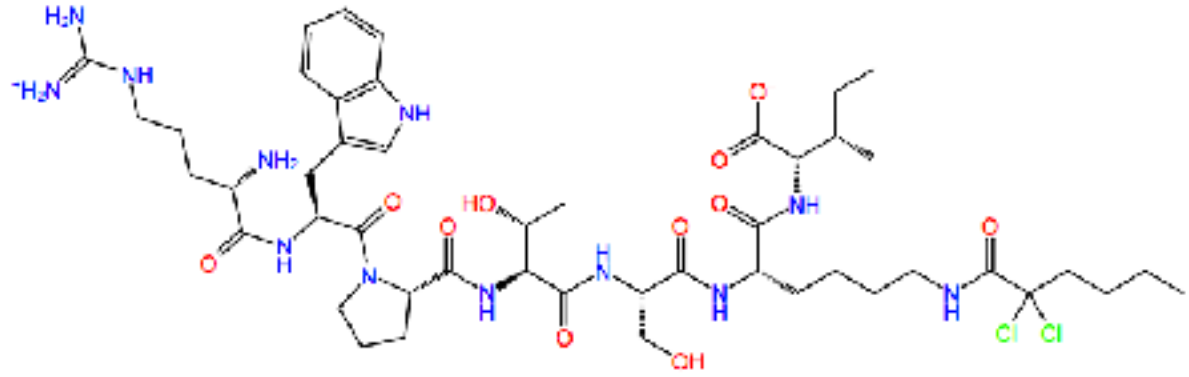


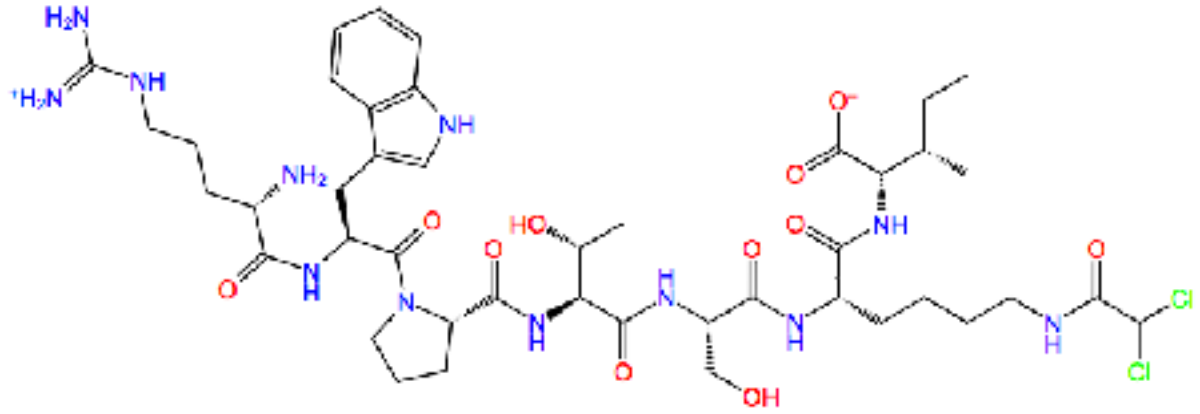


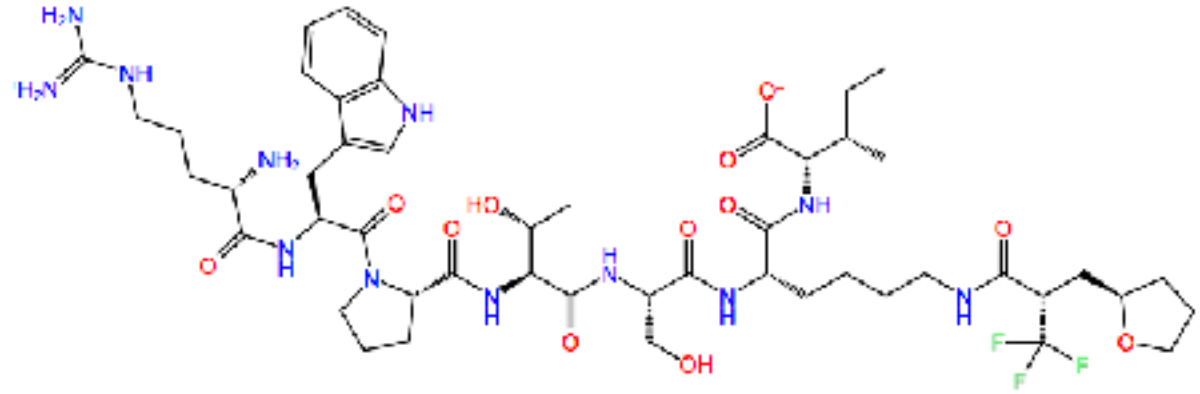


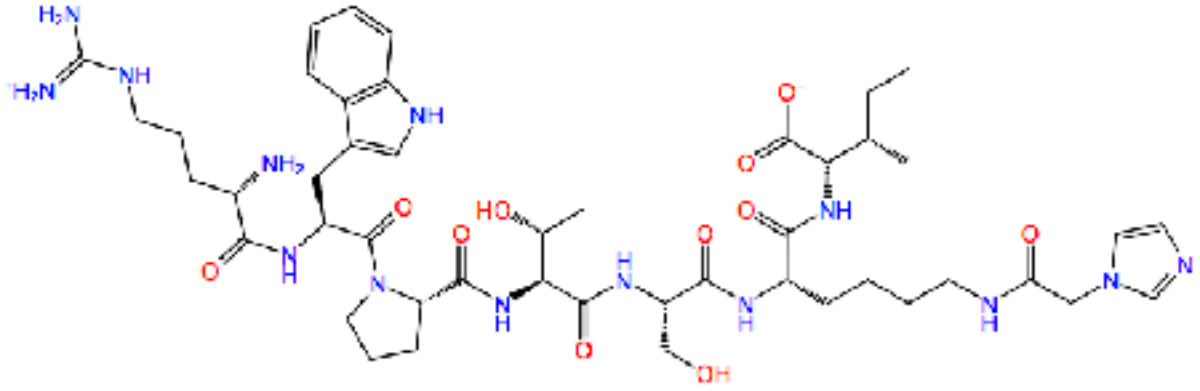


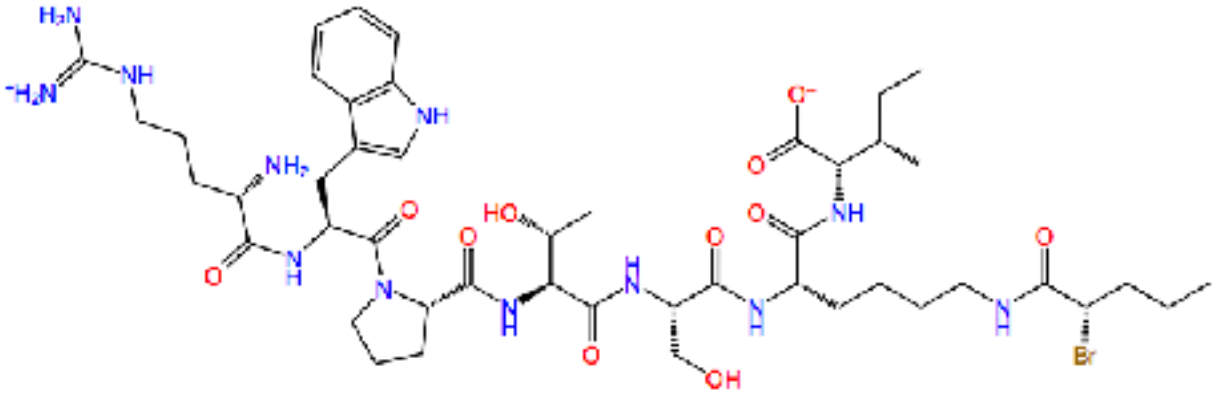











H2N
NH
H2N
NH2
NH
O
O
N
H
O
N
O
HO
H
N
O
H
N
O
OH
O
NH
O
O
H
N
N
H
O
Br