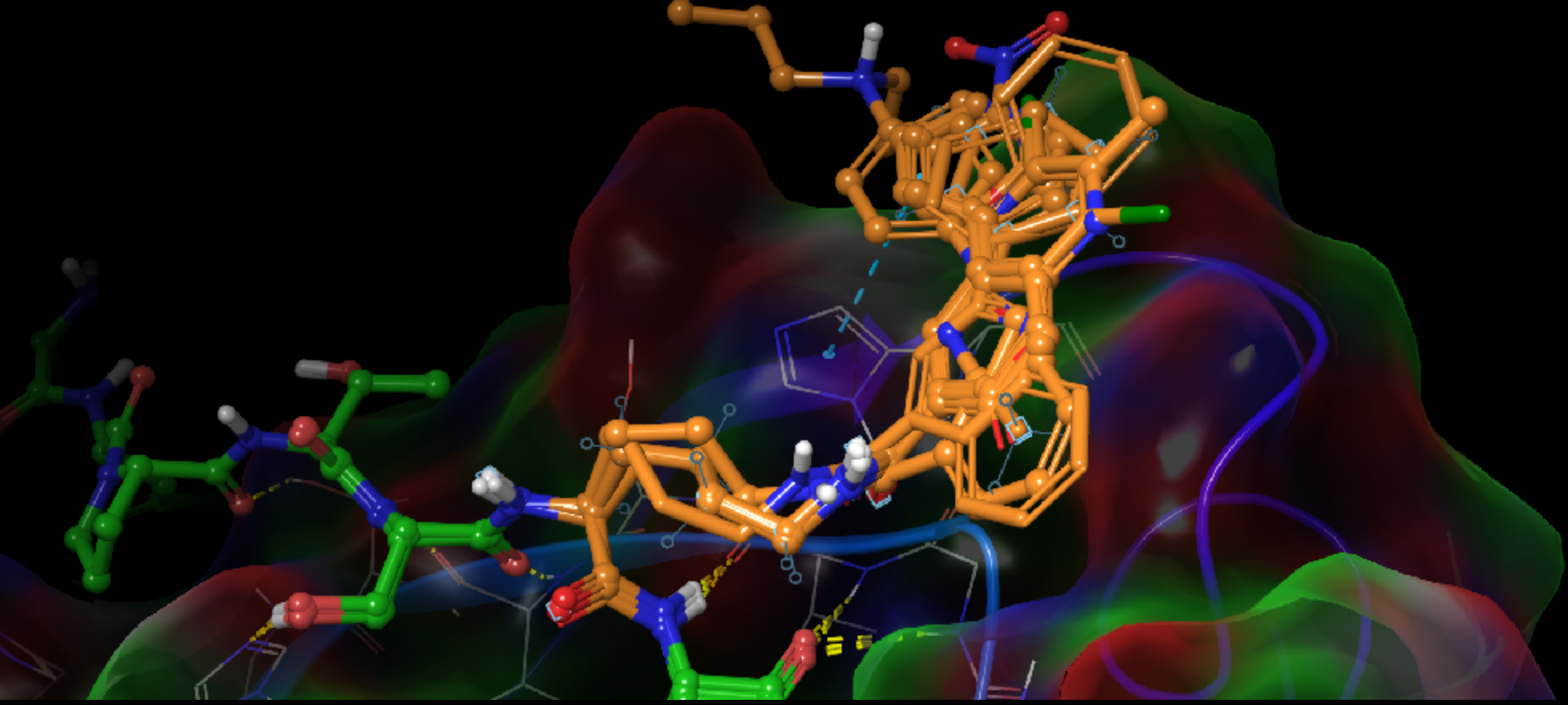


VIRTUAL SCREENING OF CAL-PDZ BINDING PEPTIDES

BRYAN CRAMPTON

BACKGROUND

- Synthesizing, purifying, and running a binding study on a peptide is **time consuming**

- Progress in enhancing binding affinity and selectivity of peptides to PDZ-domain targets has been **difficult**
    - Small molecules have also generally been unsuccessful "***undruggable target***"

- Computers can run simulations much more quickly
    - Parallelization further enhances speed

PREMISE

Chemically Modified Peptide Scaffolds Target the CFTR-Associated Ligand PDZ Domain

Jeanine Amacher, Ruizhi Zhao, Mark Spaller, Dean Madden

Peptide	Sequence	K_D from FP (μM)	K_D from ITC (μM)[a]	K_D from SPR (μM)[a]
iCAL36	ANSRWPTSII	22.6±8.0[b]	42.2±4.0	44.2±0.3
			(1.9)[c]	(2.0)[c]
Ac-iCAL36	Ac-ANSRWPTSII[d]	22.6±1.7	73.2±12.9	74.0±2.6
			(3.2)	(3.3)
$iCAL36_{Ac\text{-}K^1}$	ANSRW/PTS[Ac-K]I[e]	14.9±5.8	26.4±4.5	39.5±3.2
			(1.8)	(2.7)
$iCAL36_{Ac\text{-}K^3}$	ANSRW/P[Ac-K]SII[e]	180±110	91.9[f]	122±10
			(0.5)	(0.7)
$iCAL36_{Ac\text{-}K^4}$	ANSRW/[Ac-K]TSII[e]	32.3±6.2	49.4[f]	64.6±4.0
			(1.5)	(2.0)
$iCAL36_{Ac\text{-}K^5}$	ANSR[Ac-K]PTSII[e]	550±400	ND	298±5
				(0.5)
$iCAL36_{BB\text{-}K^1}$	ANSRW/PTS[BB-K]I[g]	11.1±3.9	20.7±2.0	20.0±0.4
			(1.9)	(1.8)
$iCAL36_{FB\text{-}K^1}$	ANSRW/PTS[FB-K]I[h]	11.4±3.7	12.8±1.7	27.9±3.6
			(1.1)	(2.4)
$iCAL36_{Tfa\text{-}K^4}$	ANSRW/PTS[Tfa-K]I[i]	14.1±2.0	34.5±8.8	22.3±4.2
			(2.4)	(1.6)

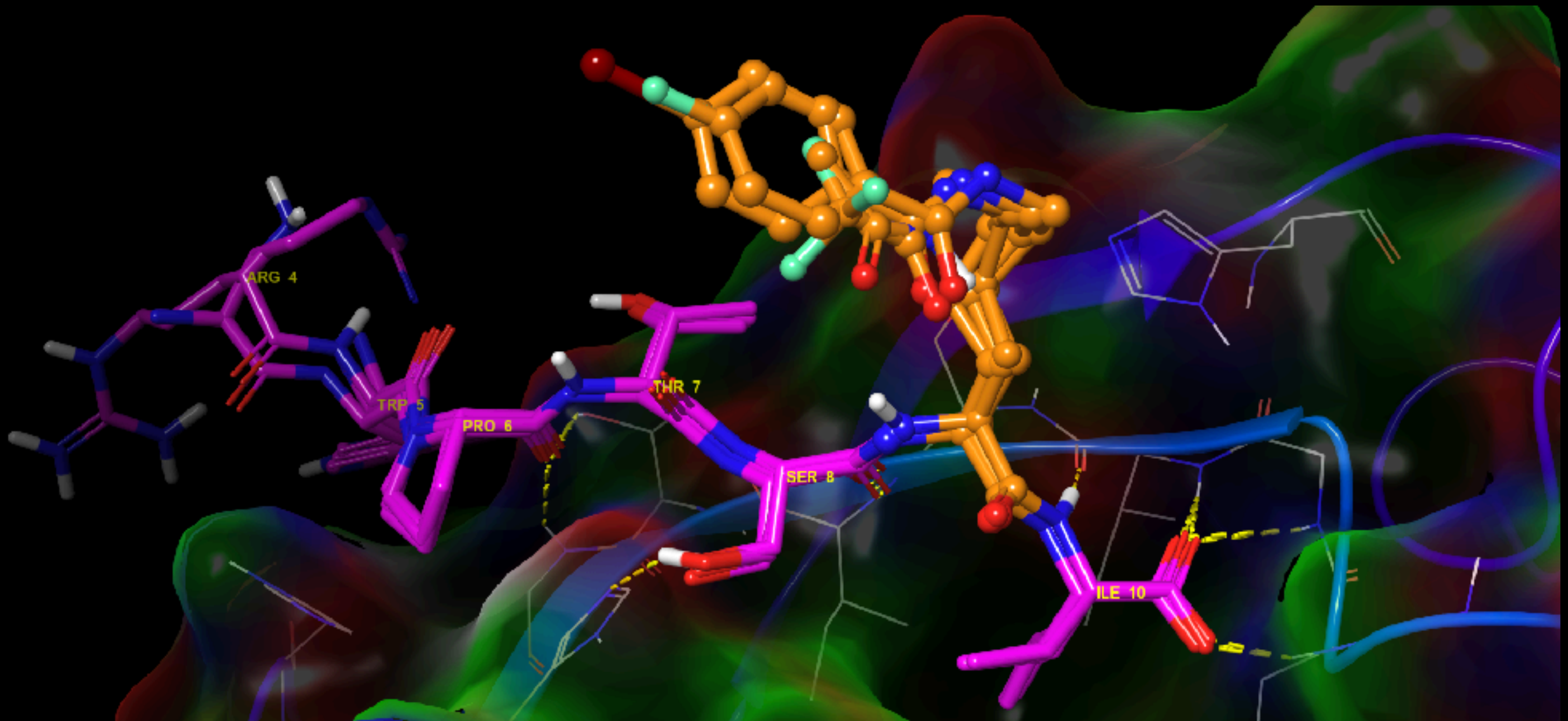
DOCKED CRYSTAL STRUCTURE

RWPTS[Ac-K]I

WPTS[4-bromobenzoic-acyl-K]I

RWPTS[Tfa-acyl-K]I

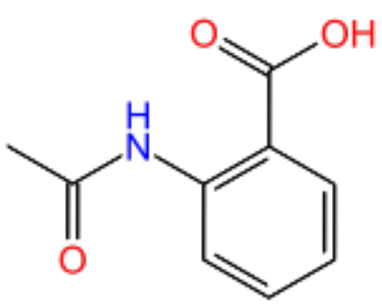
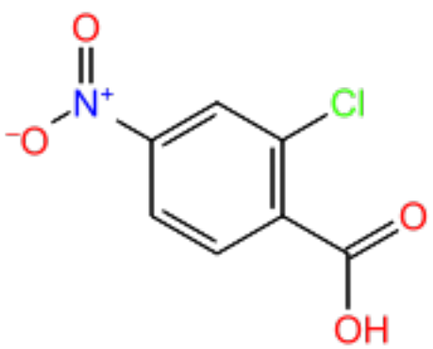
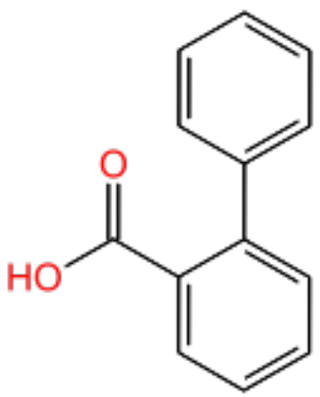
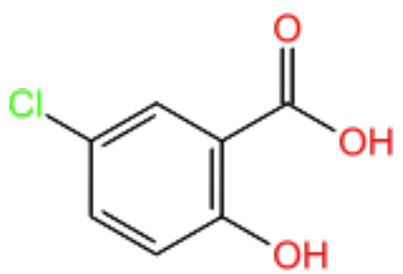
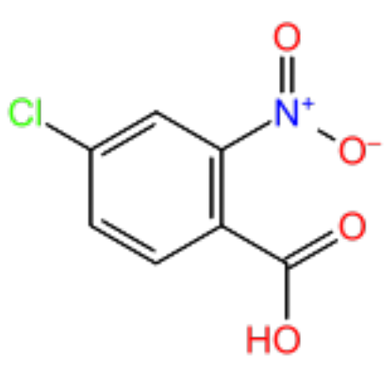
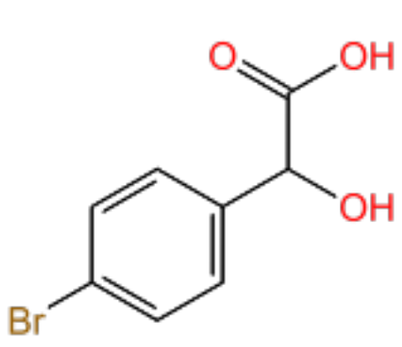
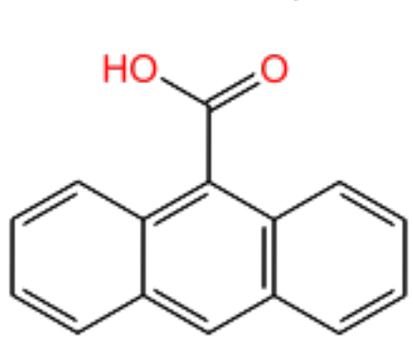
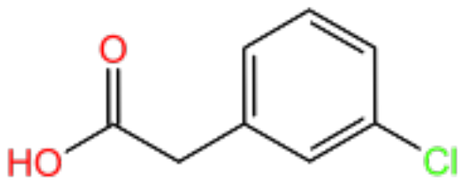
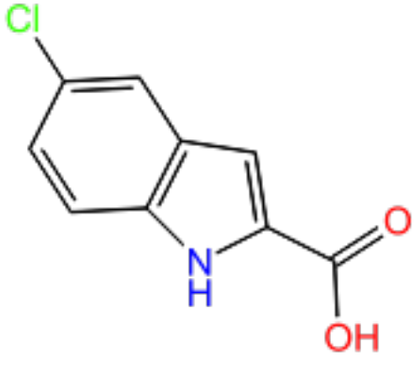
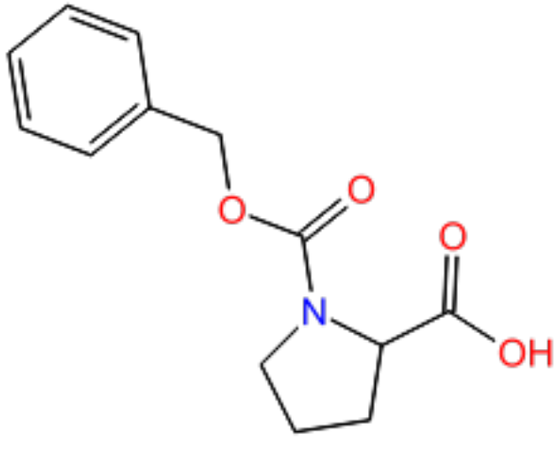
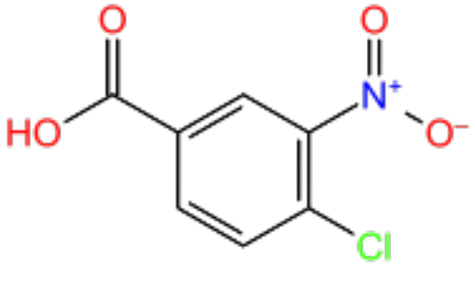
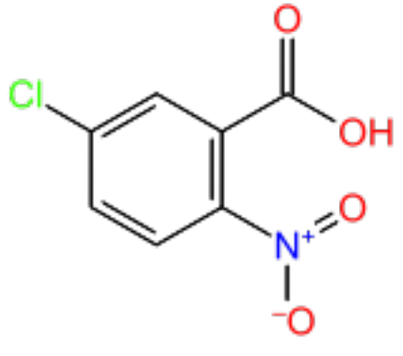
WPTS[4-fluorobenzoic-acyl-K]I



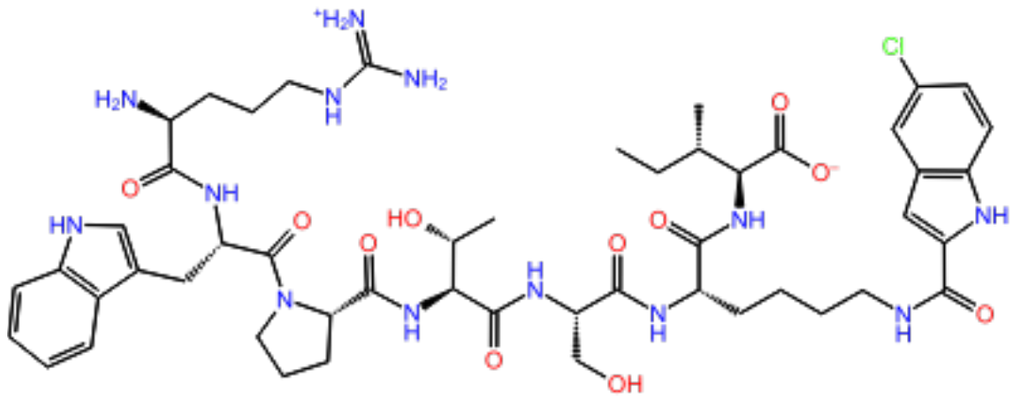
PREMISE

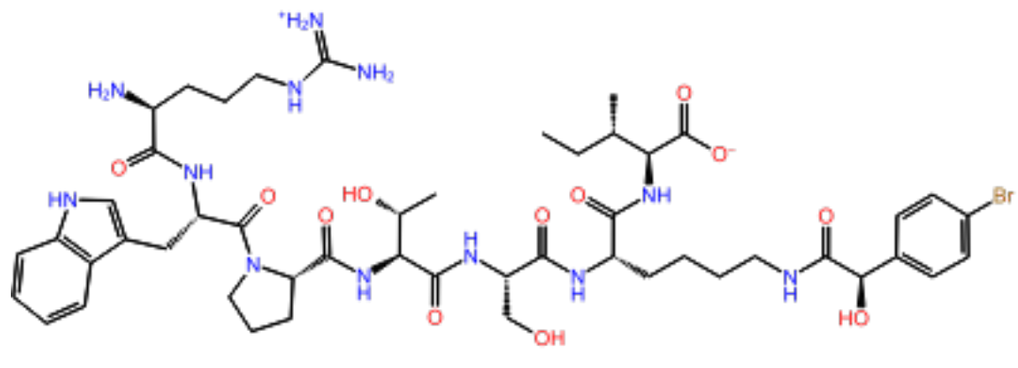
- The P(-1) position of CAL-PDZ binding peptides tolerates modifications
    - Particularly a Lys residue coupled to organic acid has shown good results
- Screen a library of ~700 purchasable carboxylic acids for binding affinity, synthesize top hits and run ITC binding study to verify

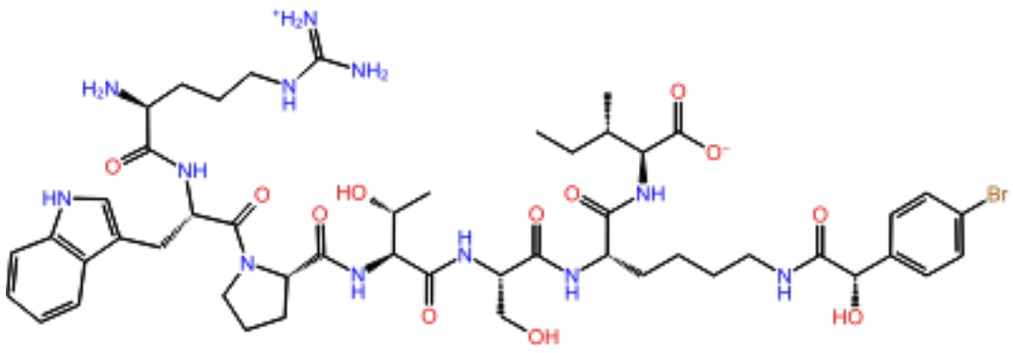
FILTERED ACID LIBRARY SAMPLE

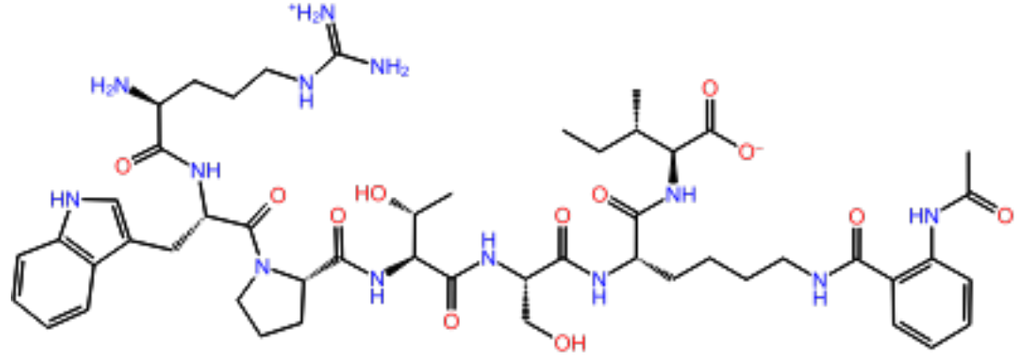













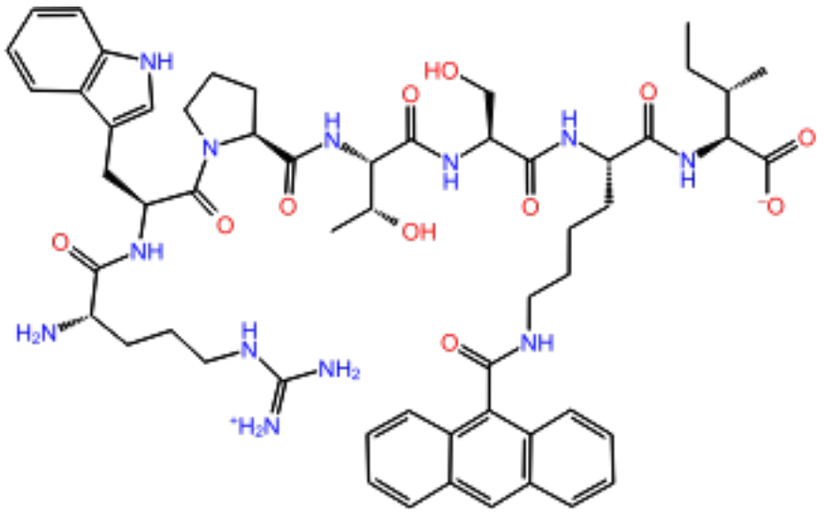
ENUMERATED PEPTIDES SAMPLE

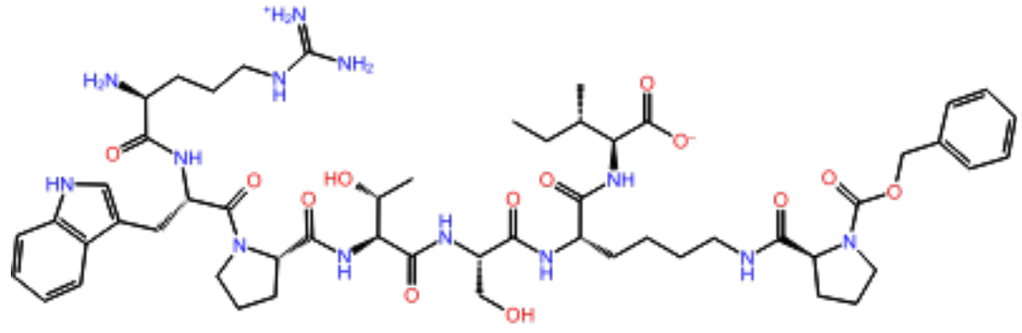




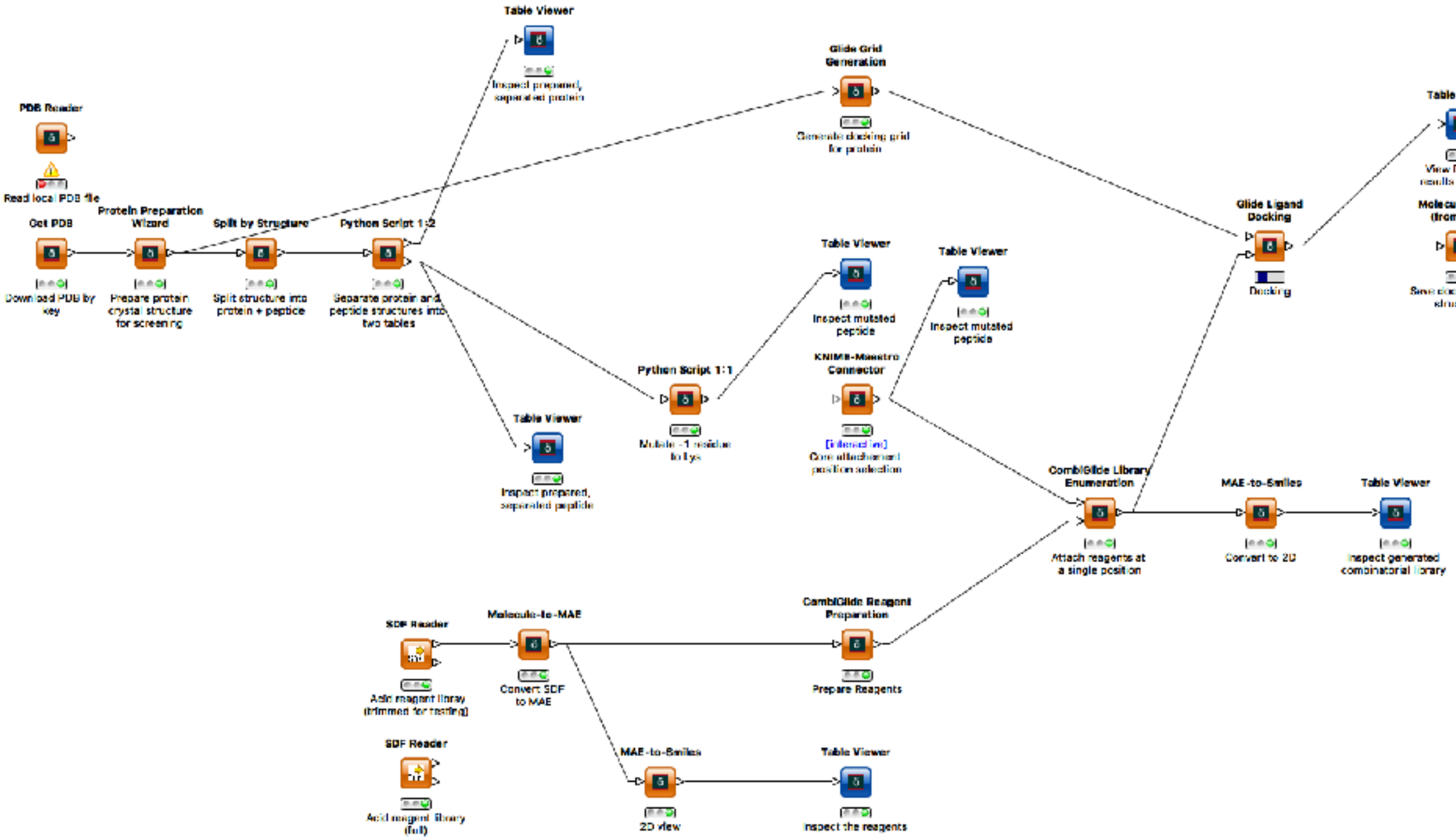








PREVIOUS WORKFLOW



- Turns out most of those steps only need to be done once
- The time consuming part of the workflow is docking and simulation with MM-GBSA
- Easier to run these from terminal when parallelizing rather than KNIME
- Jobs fail, need to be recovered, etc.

UPDATES

1. New approach to docking and estimating binding affinity of peptides

2. Parallelization of jobs over Google Compute Engine

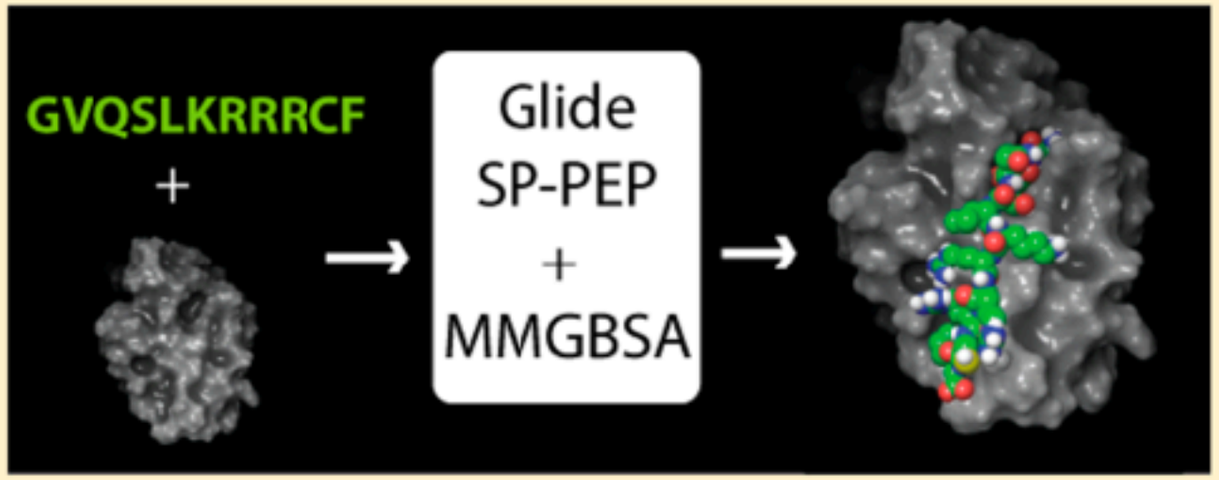
3. Initial Results!

NEW APPROACH TO DOCKING PEPTIDES

Improved Docking of Polypeptides with Glide

Ivan Tubert-Brohman, Woody Sherman, Matt Repasky, and Thijs Beuming*

ABSTRACT: Predicting the binding mode of flexible polypeptides to proteins is an important task that falls outside the domain of applicability of most small molecule and protein−protein docking tools. Here, we test the small molecule flexible ligand docking program Glide on a set of 19 non-α-helical peptides and systematically improve pose prediction accuracy by enhancing Glide sampling for flexible polypeptides. In addition, scoring of the poses was improved by post-processing with physics-based implicit solvent MM-GBSA calculations. Using the best RMSD among the top 10 scoring poses as a metric, the success rate (RMSD $\leq$ 2.0 Å for the interface backbone atoms) increased from 21% with default Glide SP settings to 58% with the enhanced peptide sampling and scoring protocol in the case of redocking to the native protein structure. This approaches the accuracy of the recently developed Rosetta FlexPepDock method (63% success for these 19 peptides) while being over 100 times faster. Cross-docking was performed for a subset of cases where an unbound receptor structure was available, and in that case, 40% of peptides were docked successfully. We analyze the results and find that the optimized polypeptide protocol is most accurate for extended peptides of limited size and number of formal charges, defining a domain of applicability for this approach.



NEW APPROACH TO DOCKING PEPTIDES

1. Generate library of (707) compounds from bound peptide in crystal structure **RWPTS[Ac-K]I**

2. Dock each compound with new *SP-Peptide* docking protocol in **Glide** *using heavy constraints*

3. Simulate docked poses with **Prime MMGBSA** to get predicted ΔG_{bind} , instead of using docking score

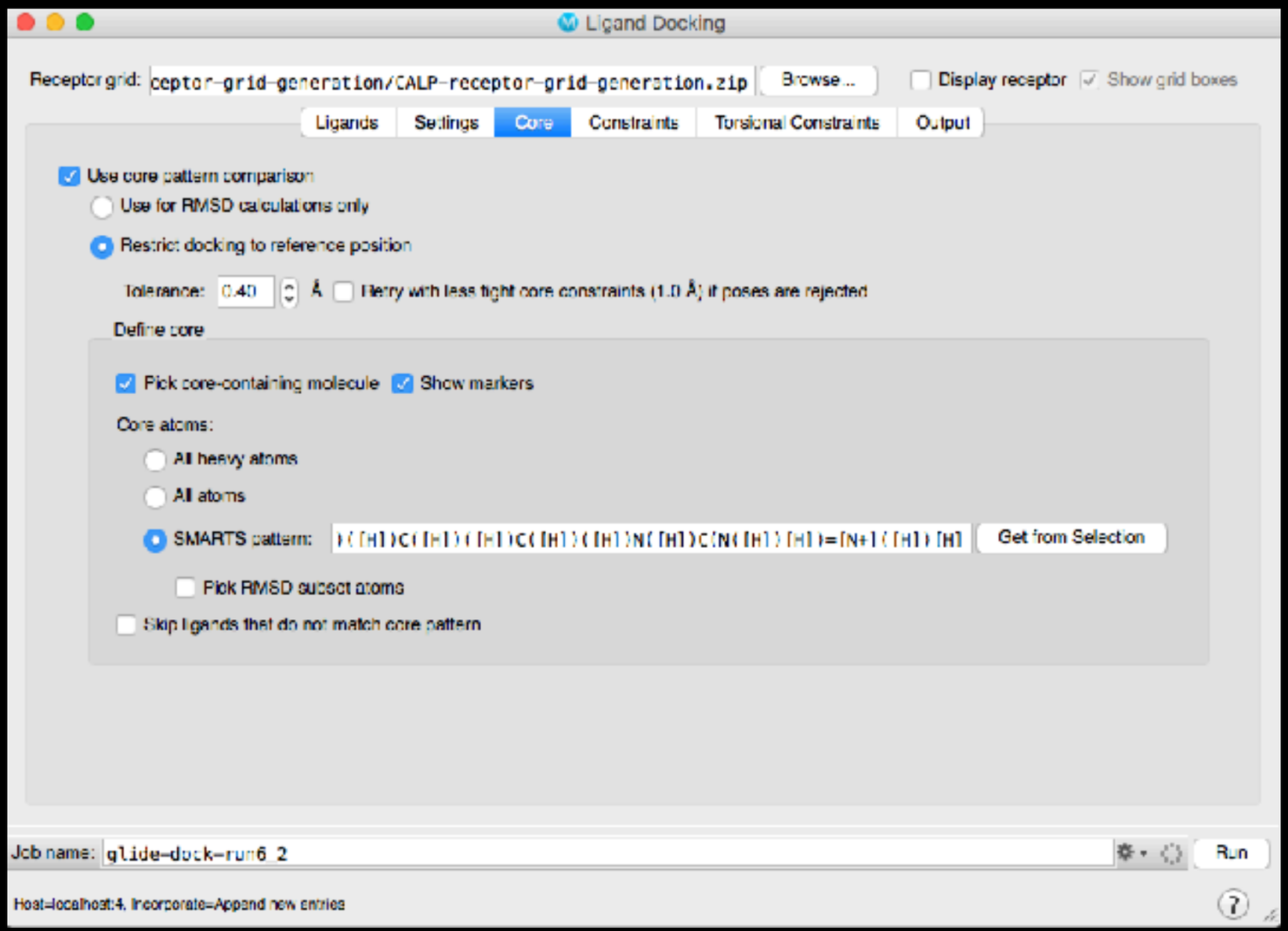
GLIDE SP-PEPTIDE DOCKING

- Previous approach was to use required H-bonds on the receptor grid to constrain motion of peptide backbone
  - Caused processing to take too long—each pose generated then checked for constraints
- New approach is to use **SMART pattern** to identify atoms to constrain in each ligand
- **SP-Peptide** protocol from recent publication, rather than the **HTVS —> SP —> XP** funnel

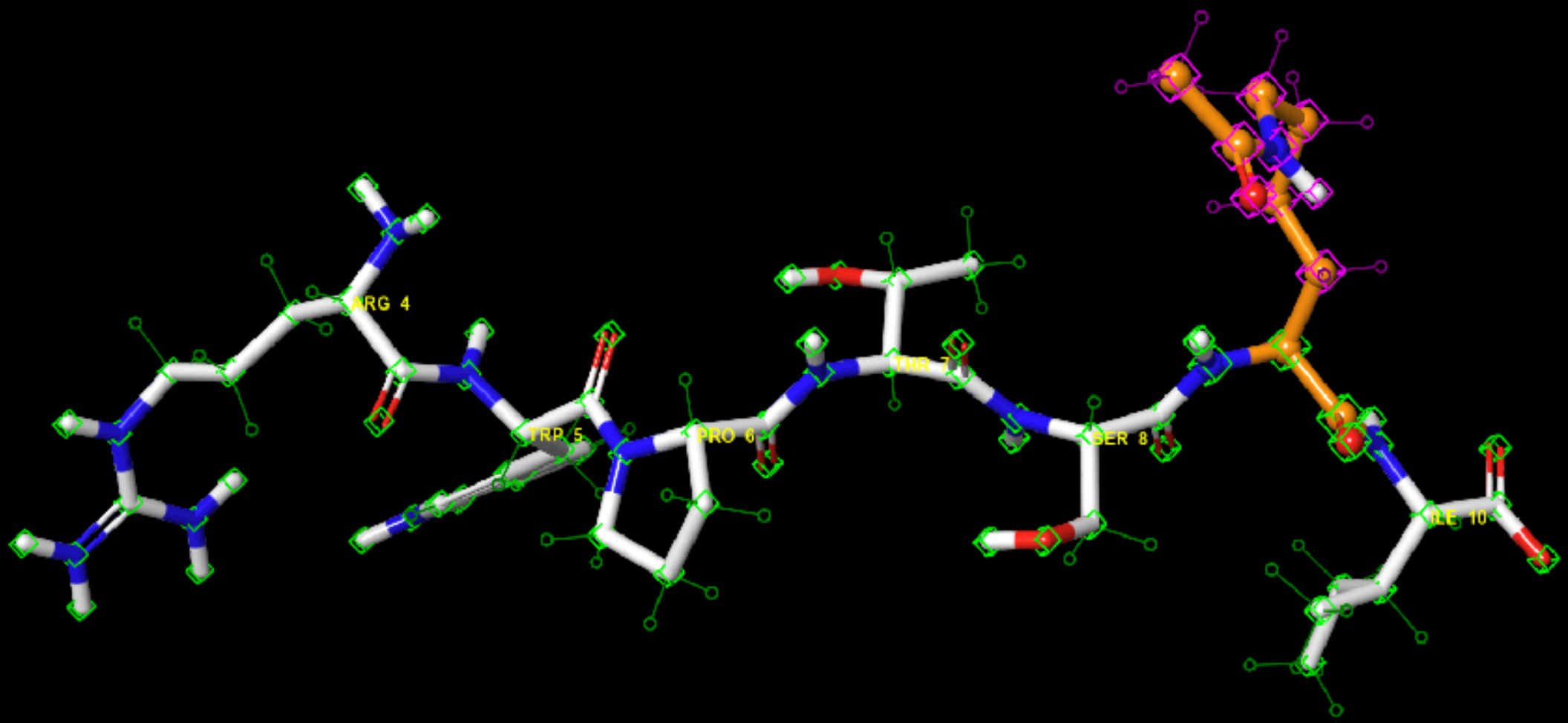
SMART PATTERN CONSTRAINTS

- **SMART pattern:**

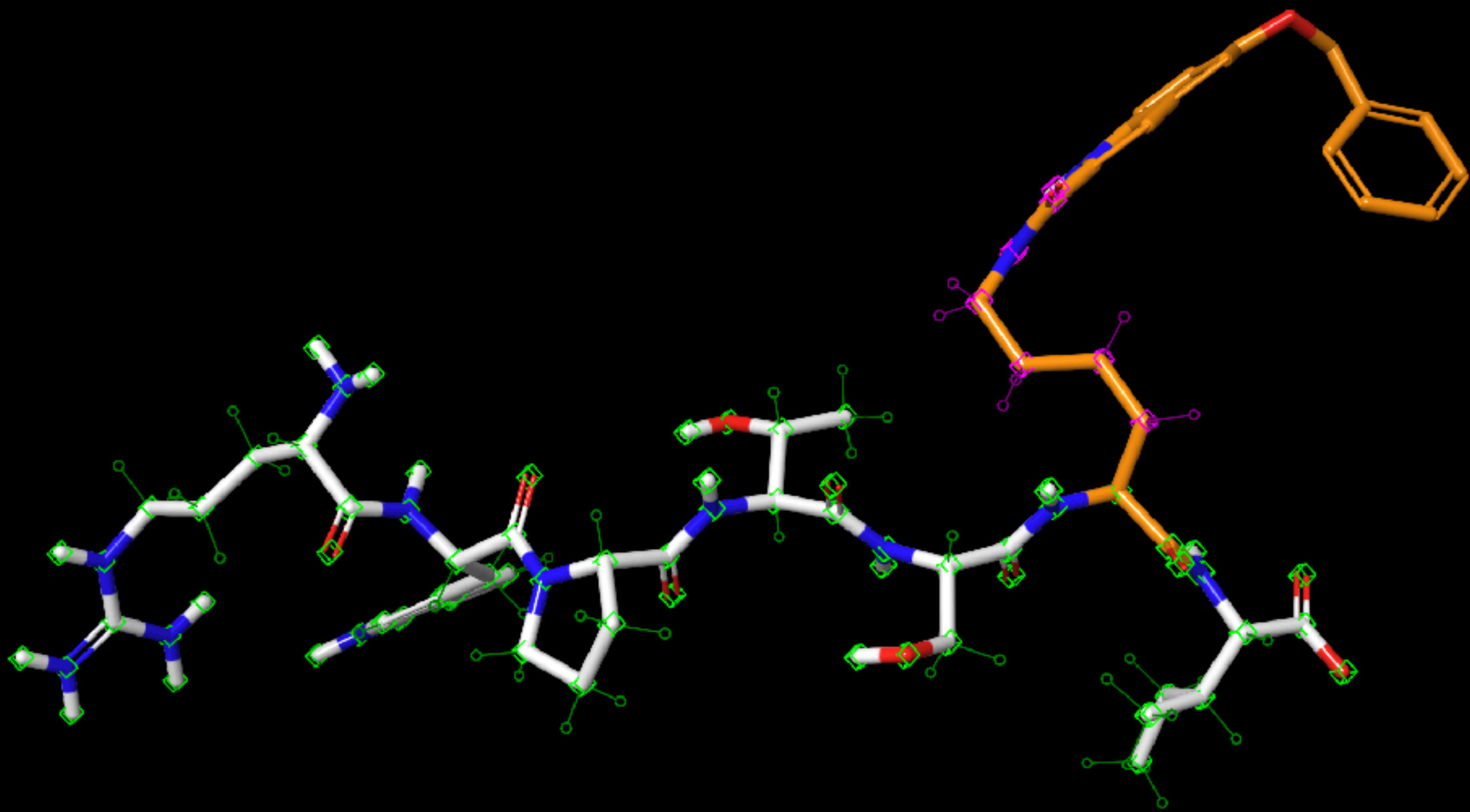
C([H])(C(=O)N([H])C([H])(C([O-])=O)C([H])(C([H])([H])[H])C([H])([H])C([H])([H])[H])N([H])C(=O)C([H])(C([H])([H])O[H])N([H])C(=O)C([H])(C([H])(C([H])([H])[H])O[H])N([H])C(=O)C1([H])C([H])([H])C([H])([H])C([H])([H])C([H])([H])N1C(=O)C([H])(C([H])([H])c2c([H])n([H])c(c23)c([H])c([H])c([H])c3[H])N([H])C(=O)C([H])(N([H])[H])C([H])([H])C([H])([H])C([H])([H])N([H])C(N([H])[H])=[N+]([H])[H]



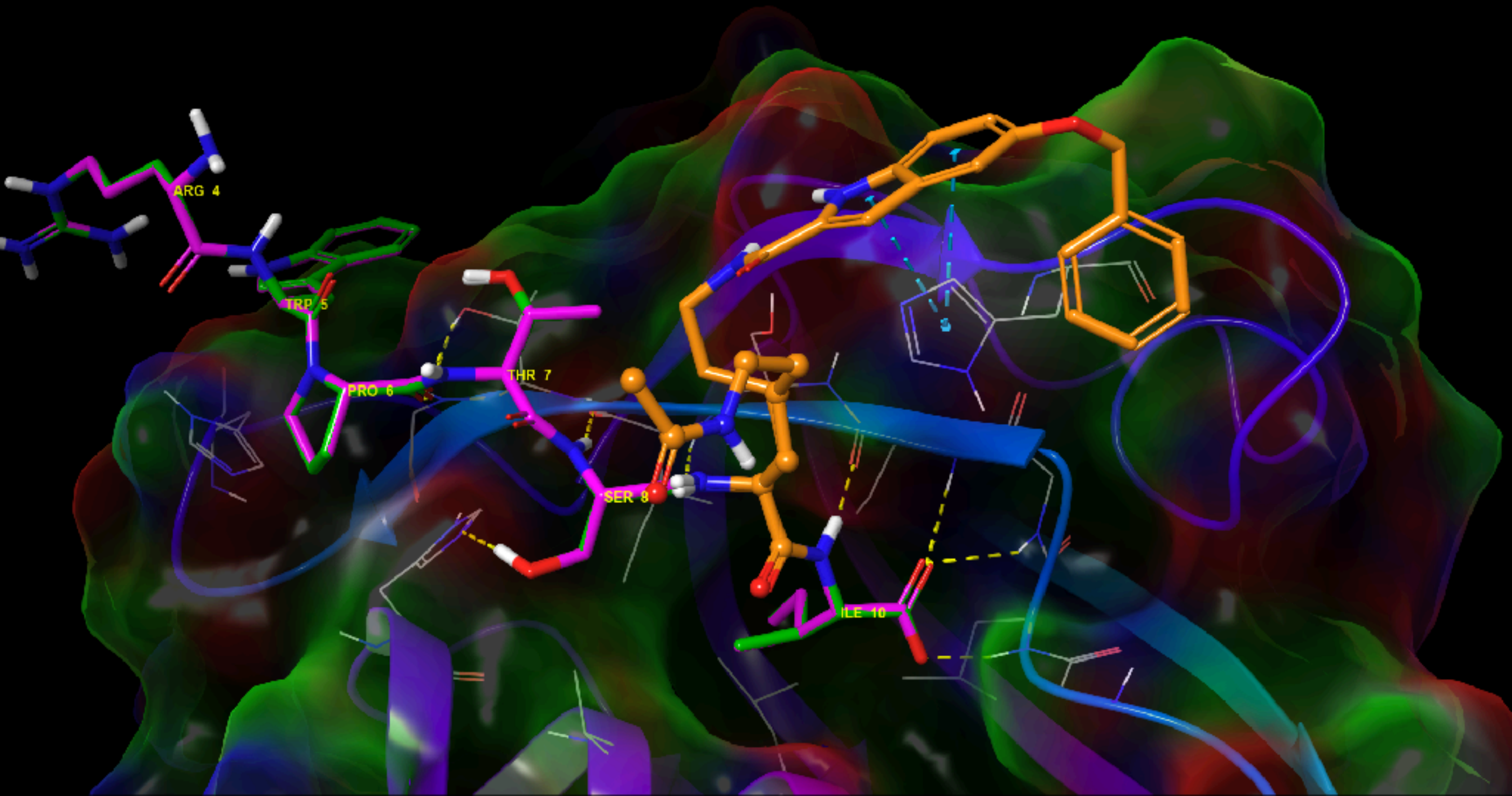
SMART PATTERN CONSTRAINTS



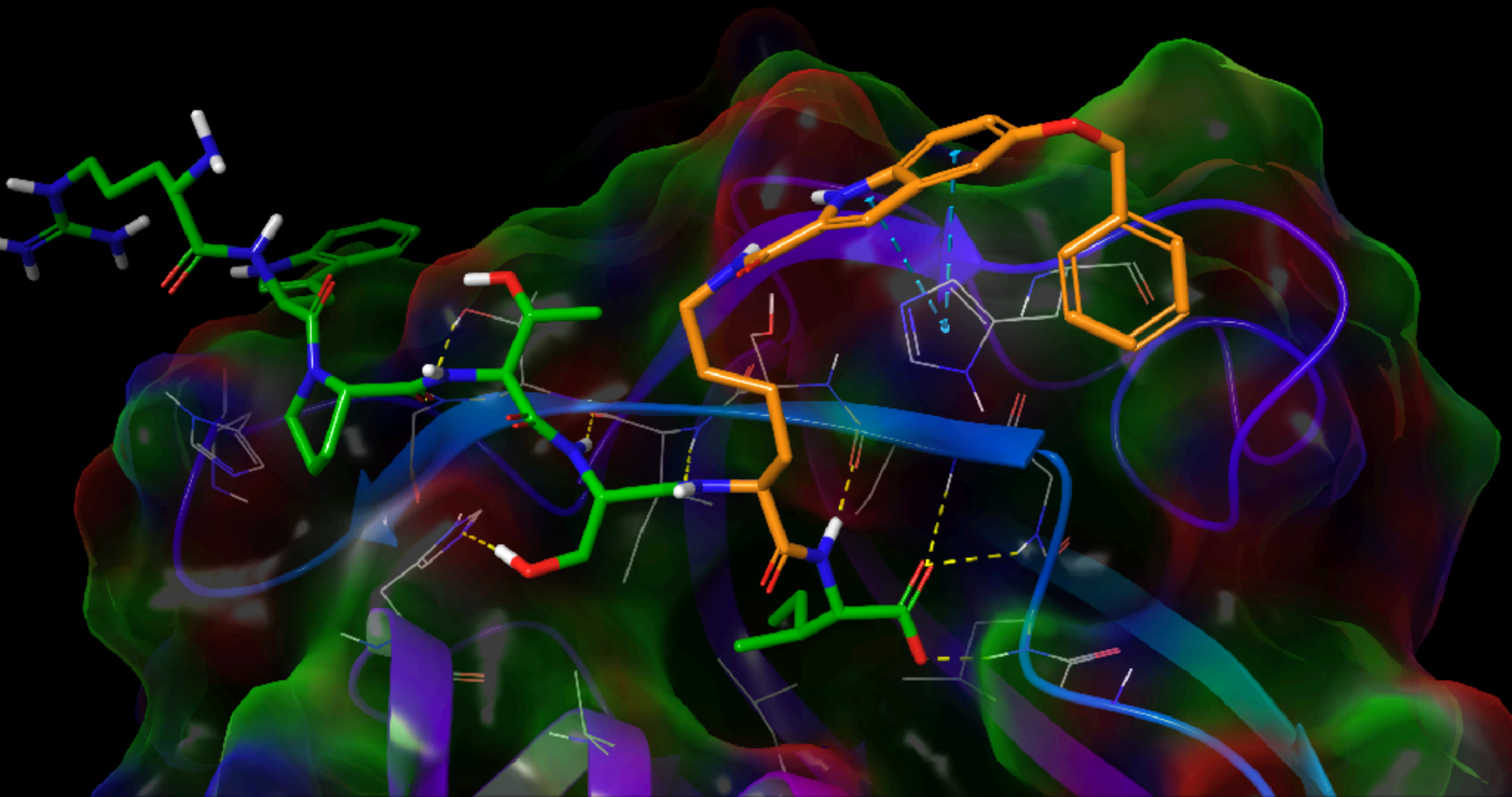
SMART PATTERN CONSTRAINTS



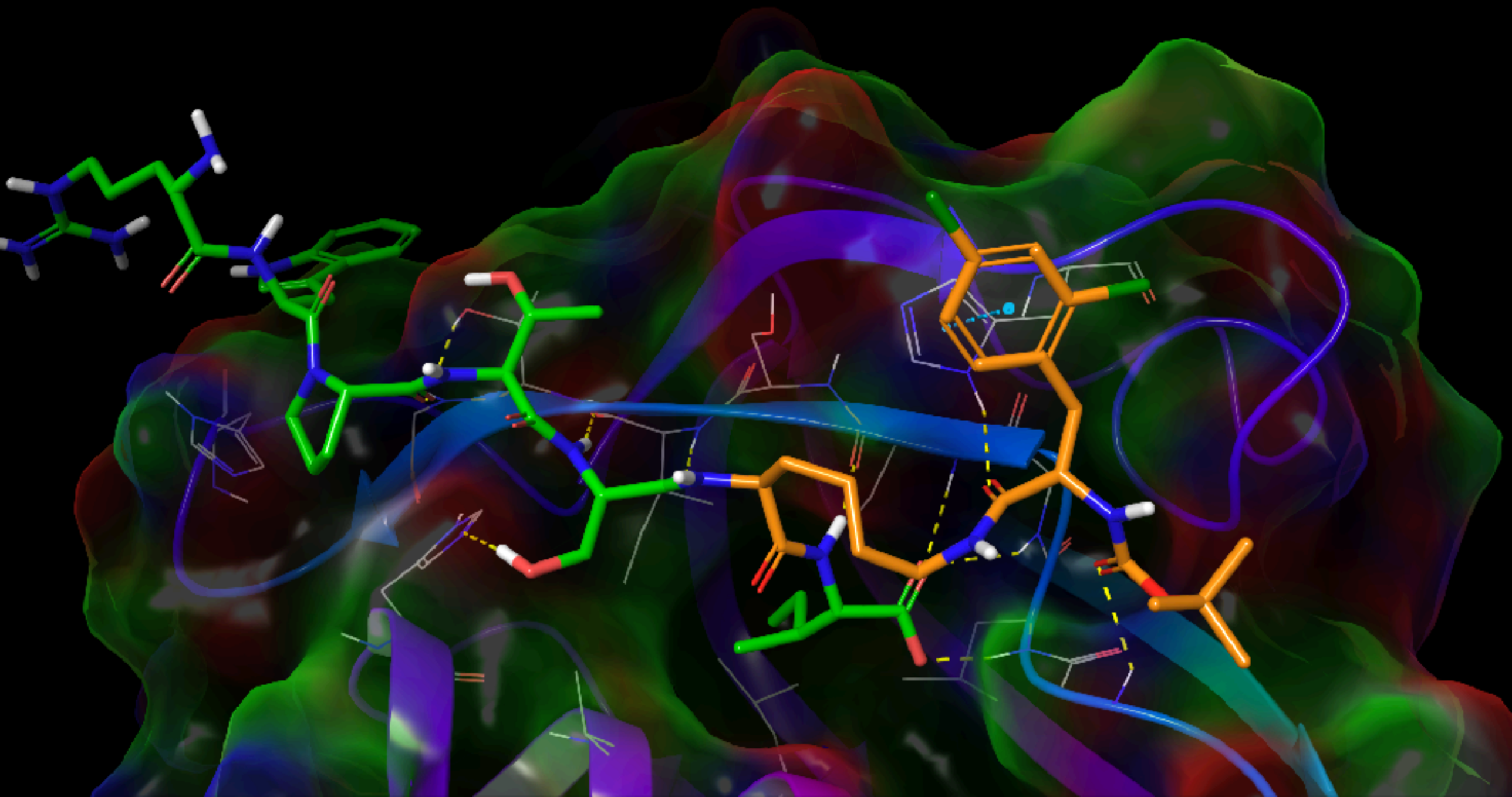
DOCKED STRUCTURES



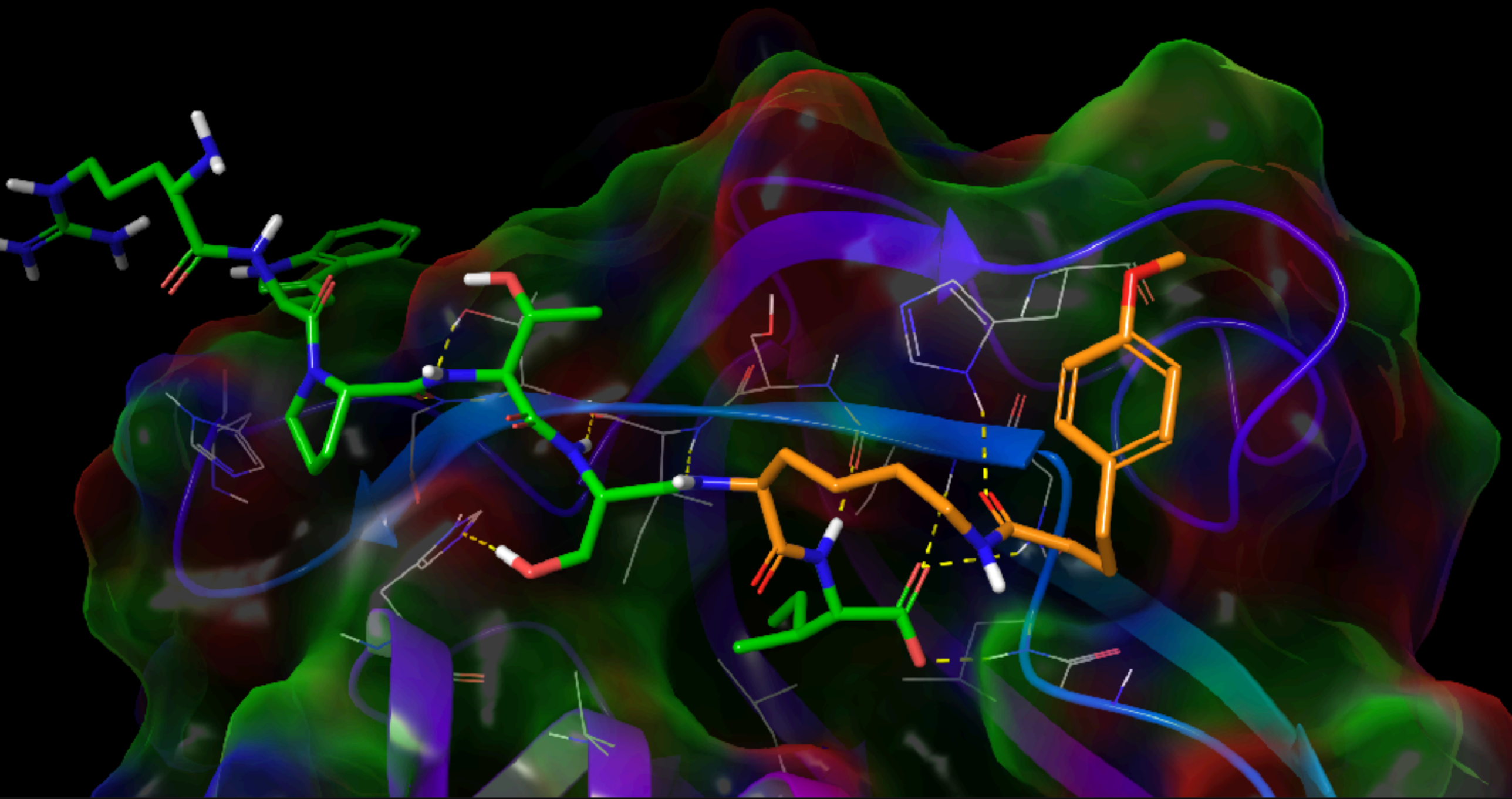
DOCKED STRUCTURES: #1/23,054



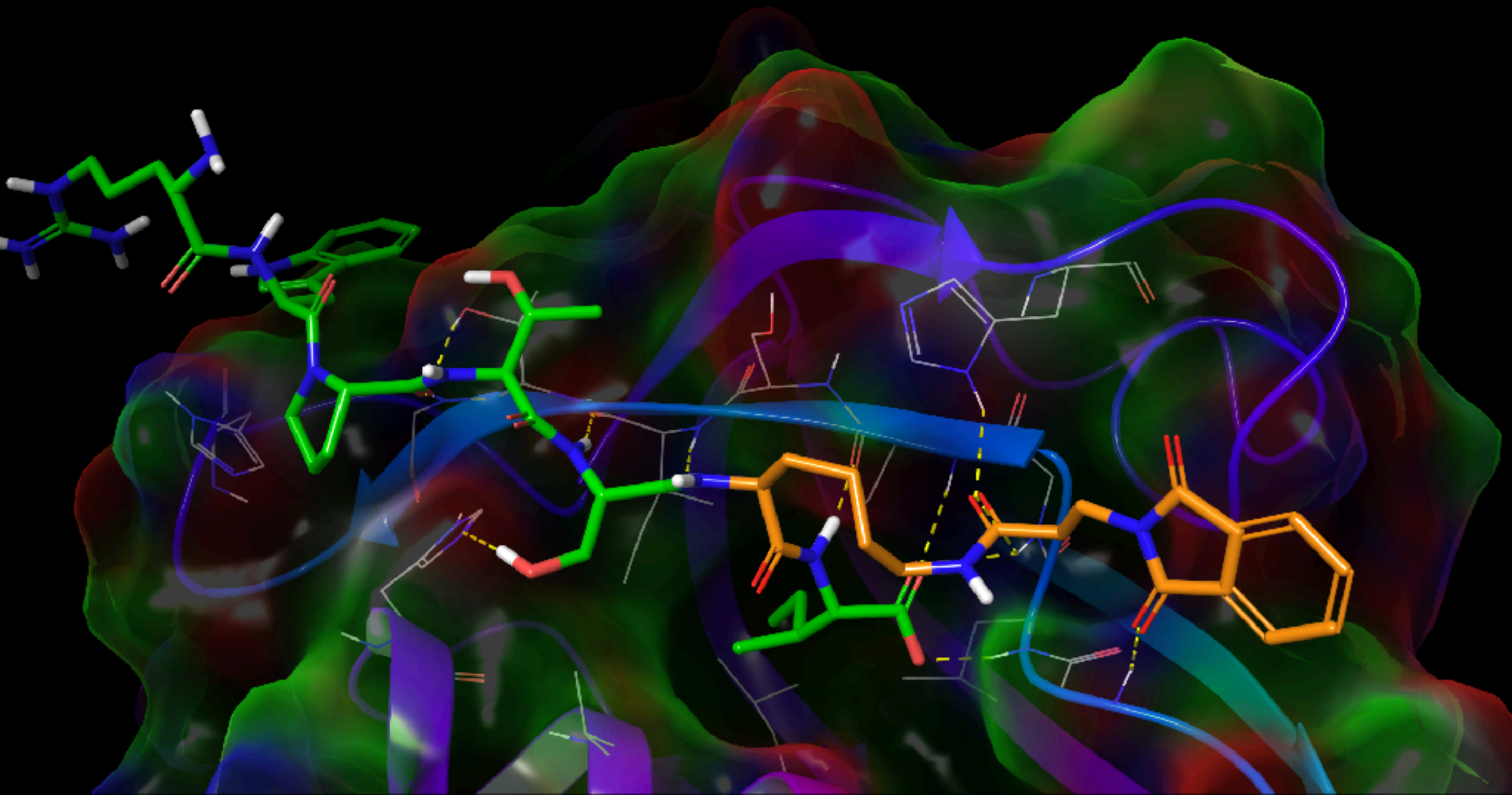
DOCKED STRUCTURES: #2/23,054



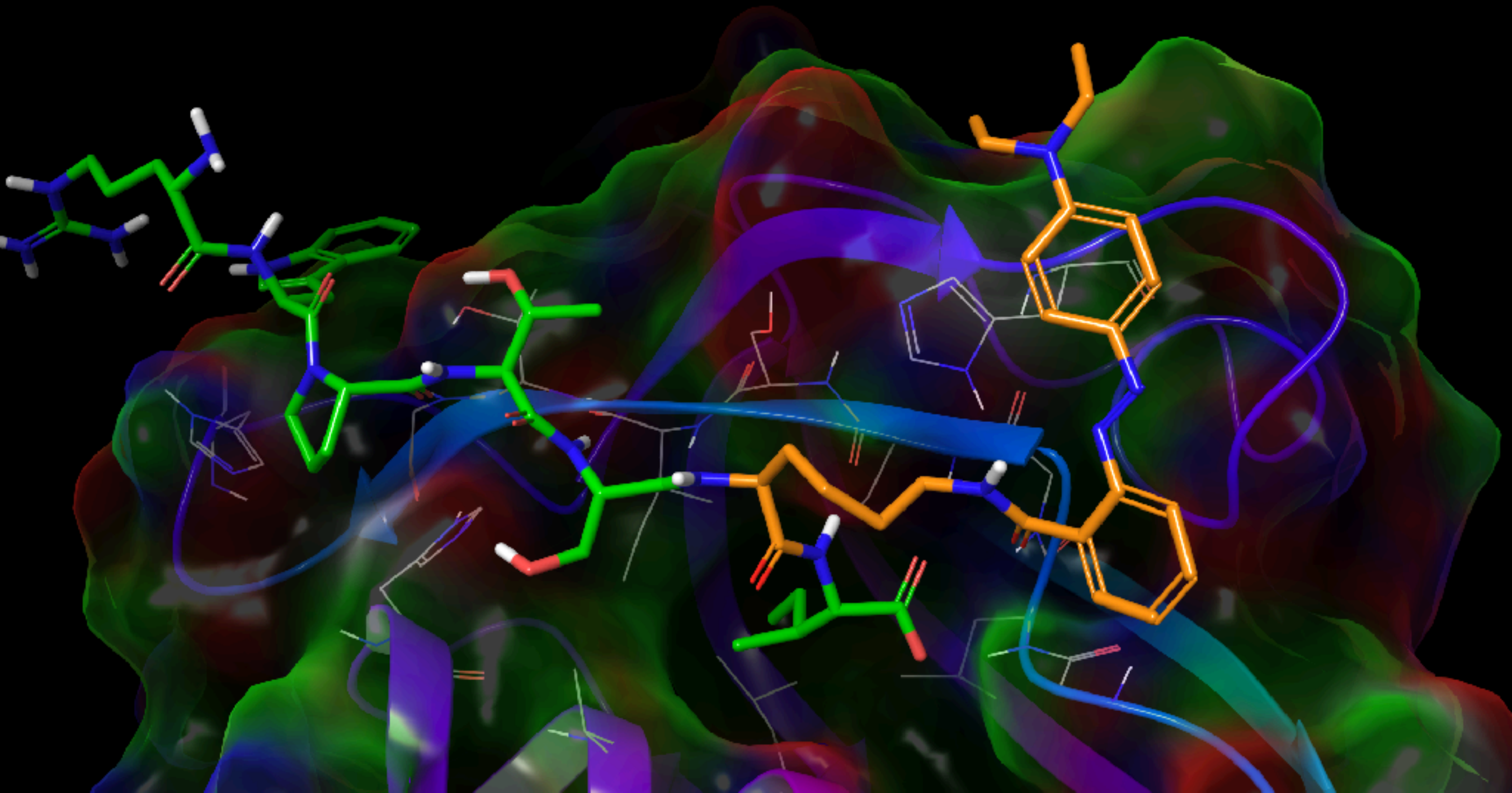
DOCKED STRUCTURES: #3/23,054



DOCKED STRUCTURES: #4/23,054



DOCKED STRUCTURES: #5/23,054

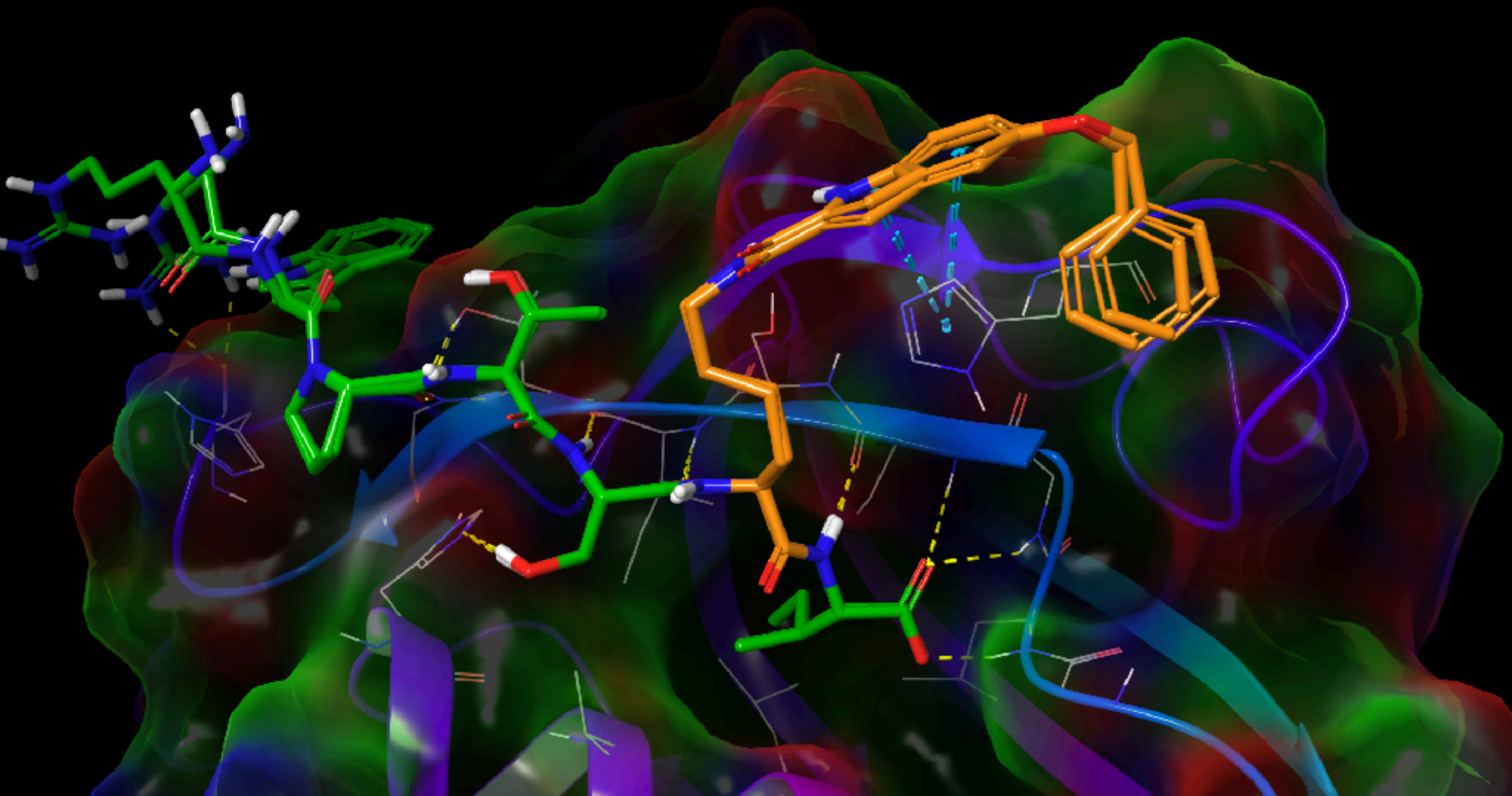


PRIME MM-GBSA SIMULATION

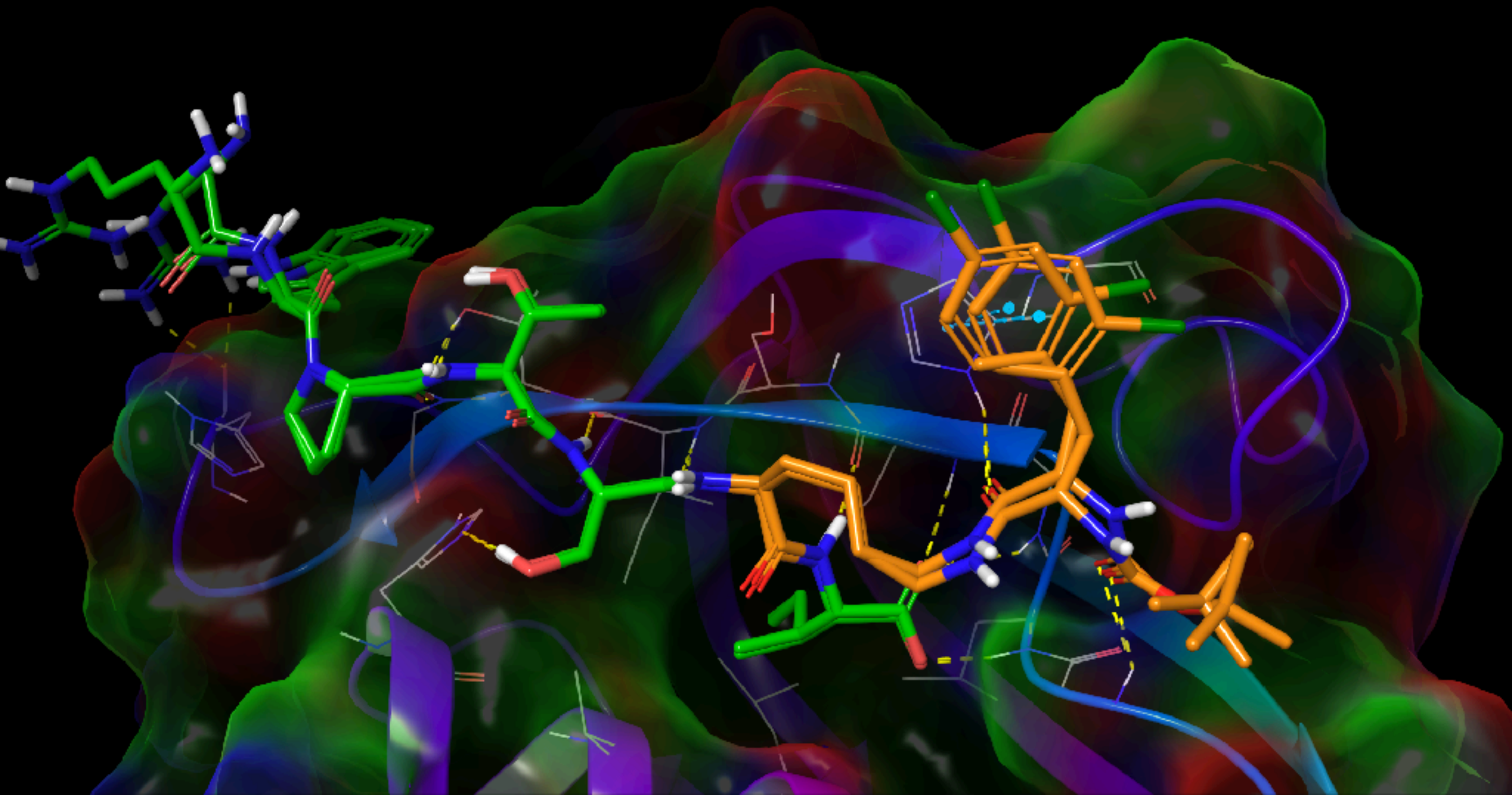
- Glide is docking software initially designed for small molecules
- There have been updates with better peptide support, but paper found that using MM-GBSA yielded more accurate results

- **Prime**: Schrödinger's protein folding / interaction software
- **MM-GBSA**: Molecular Mechanics — Generalized Born model augmented with Solvent Accessible Surface Area
    - Most commonly used implicit solvent model combination
    - Calculates free energy changes for protein, ligand, and solvent

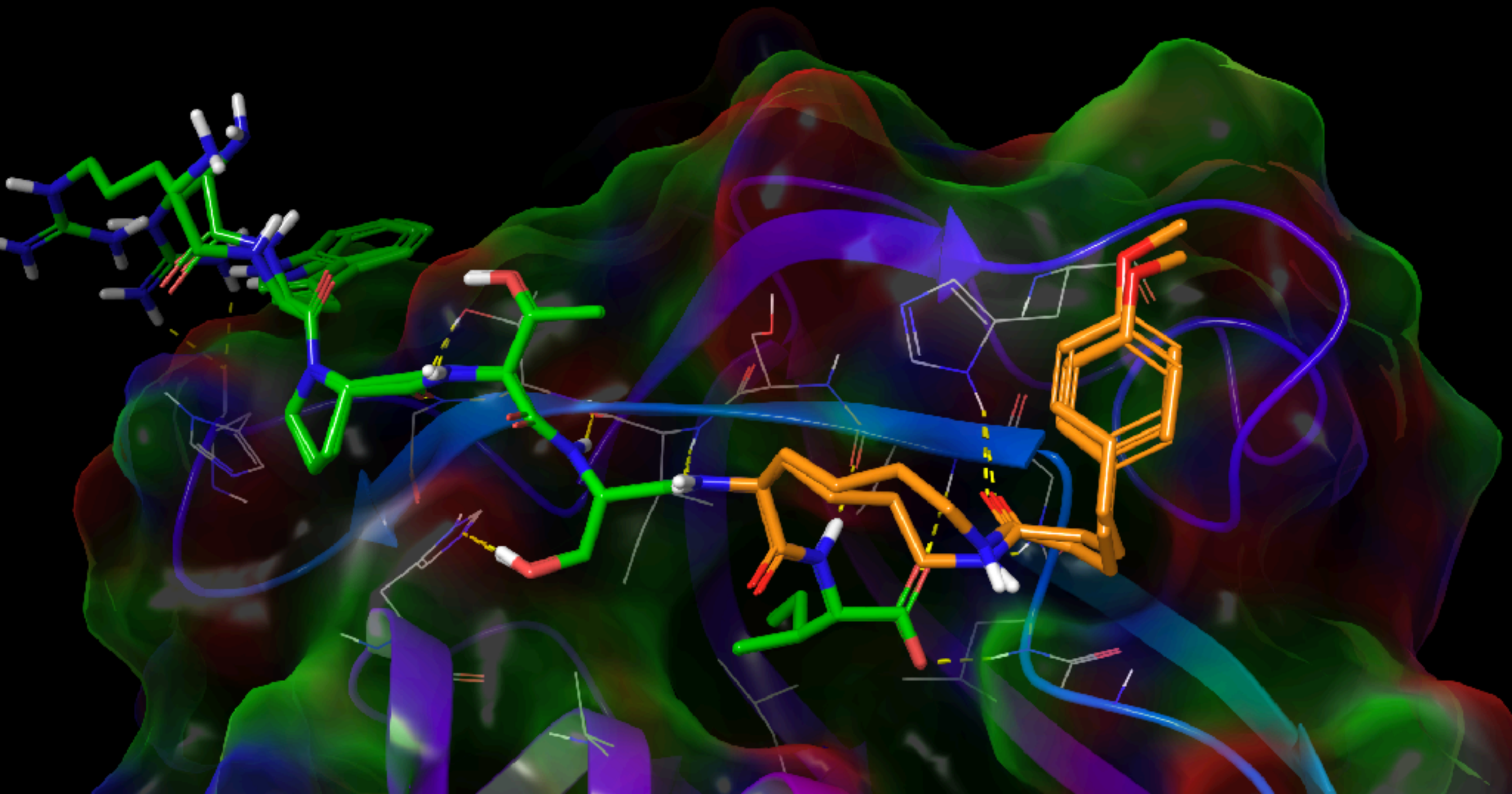
SIMULATED STRUCTURES: #1/23,054



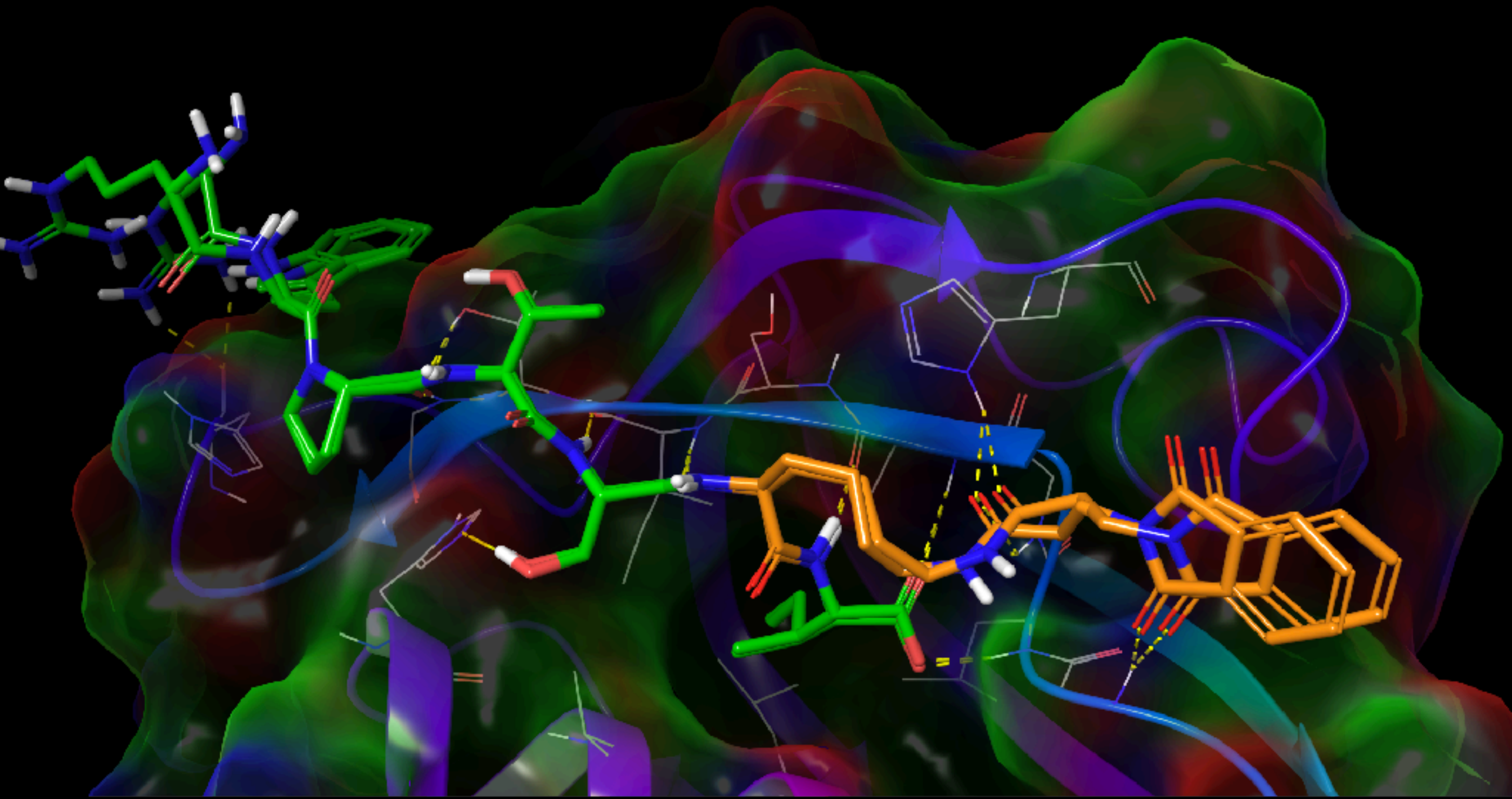
SIMULATED STRUCTURES: #2/23,054



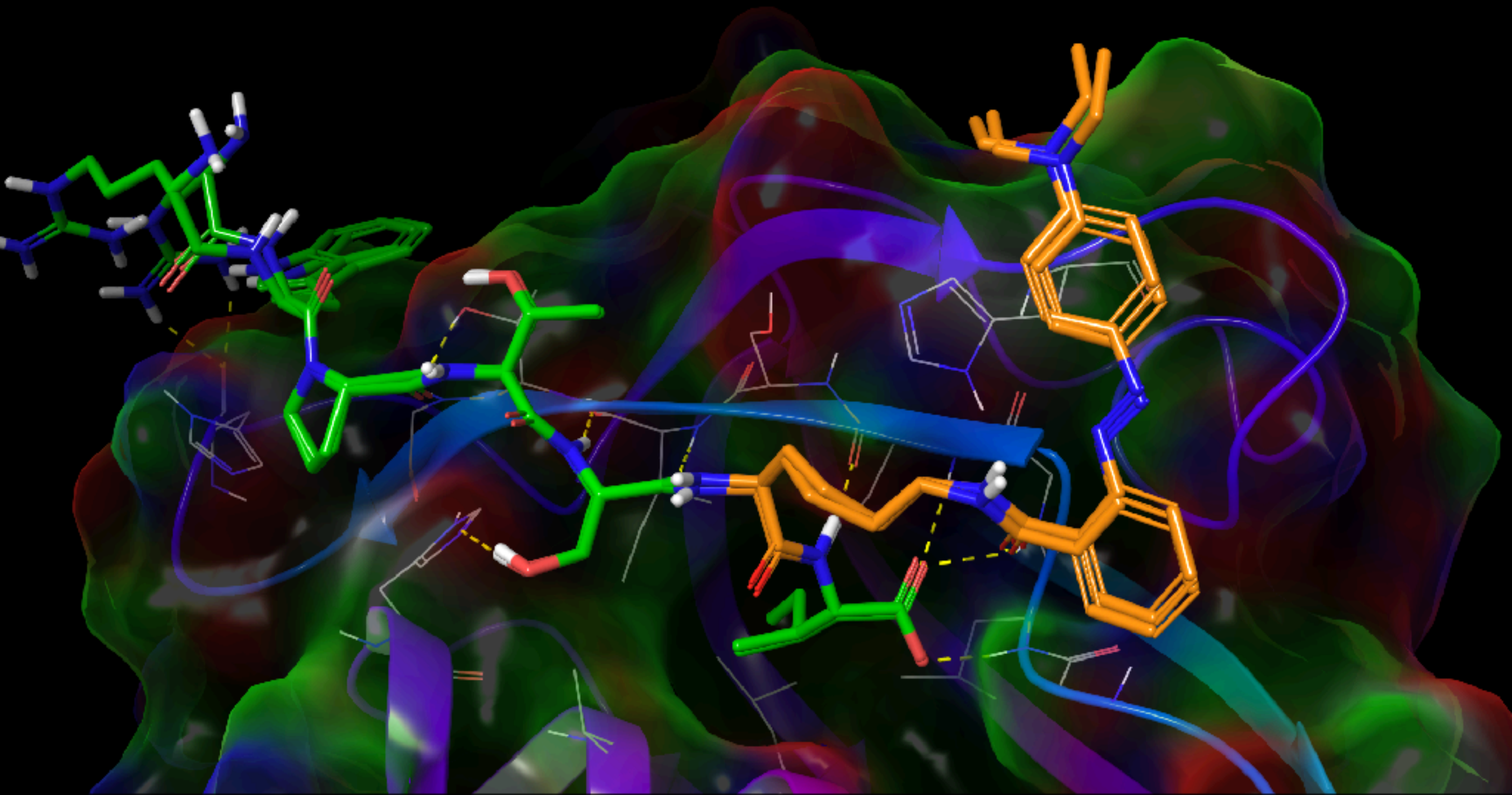
SIMULATED STRUCTURES: #3/23,054



SIMULATED STRUCTURES: #4/23,054



SIMULATED STRUCTURES: #5/23,054



THE PROCESSING PROBLEM

- Each peptide takes ~80 mins (on 1 CPU) to run through Glide SP-Peptide docking
- On average 32 docked poses generated per peptide
- Each pose takes ~5 mins to simulate
- Processing time per peptide: 345 mins
- Processing time for full (707) library: 243,915 mins = 169.4 days

THE PROCESSING PROBLEM

- Discovery Research Computing cluster offers 4 cores for free and 16 CPUs (up to 64 if available) for ~$5,000 for 5 years
- It would take 2.6 - 10.5 days to run the library
    - Difficult to maintain VPN access for that long
- Instead, setup our own “cluster” on Google Compute Engine

GOOGLE COMPUTE ENGINE

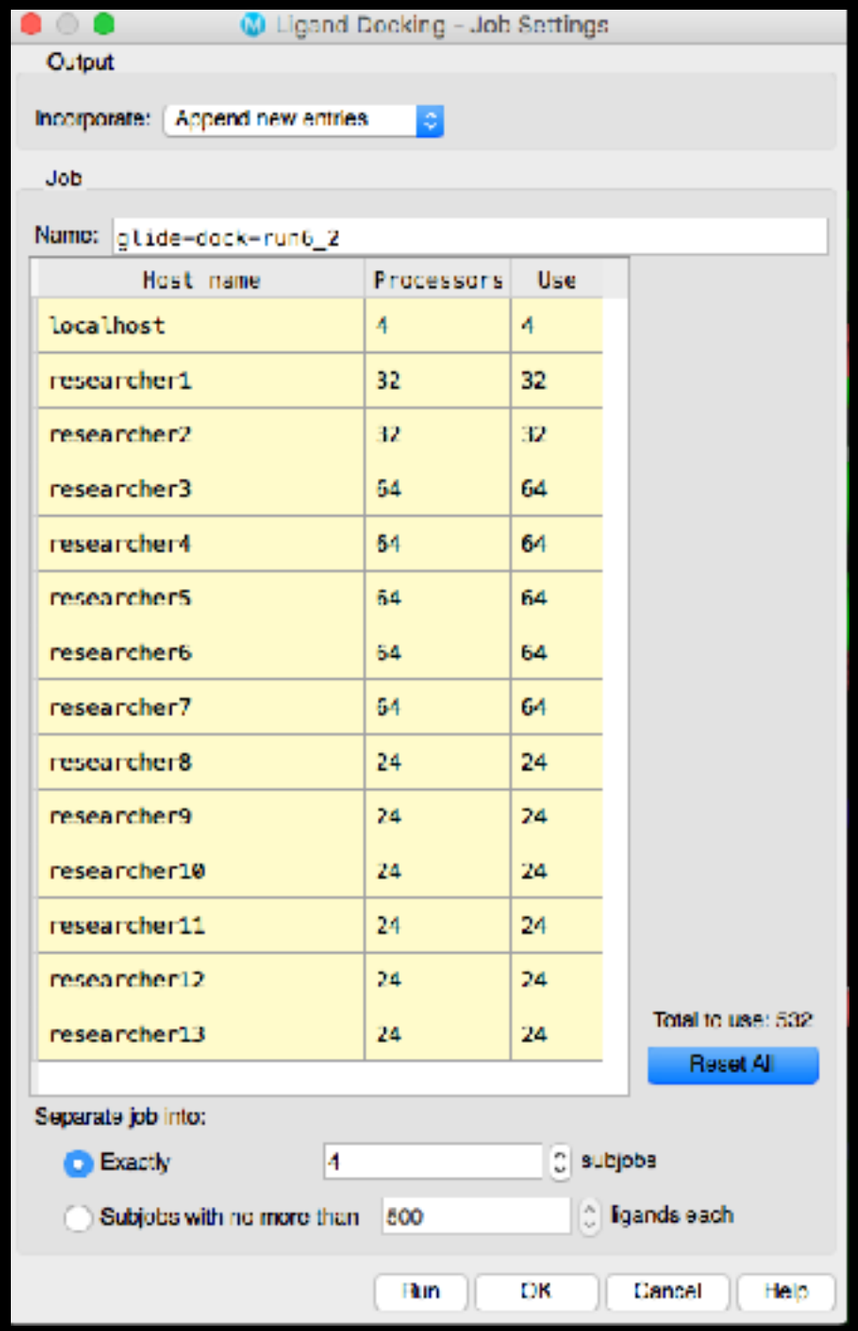
- Ran entire job on 528 CPU cluster in ~7.5 hours — nearly 4,000 hours of computing time
- **Quite a few issues getting setup:**
    - RAM and CPU ratios correct—very memory intensive, using SSD for memory swap files
    - Biggest bottleneck in early runs was internet speed to upload data to each virtual machine (VM)
    - Ended up running all jobs from a VM with super fast internet—sped up completion time by ~10x
    - Auto-extending VPN to keep licenses active

GOOGLE COMPUTE ENGINE

- Lost quite a bit of time setting up and deleting instances and creating ssh tunnel for license access to my computer
- Now have bash scripts to automatically create instances: ~30 mins to setup the full cluster
- Further steps:
    - Further automate setup, VPN setup, swap creation, etc.
    - Automate deleting instances upon job completion
    - Minimize costs by fine tuning RAM / SSD / CPU ratios
    - MM-GBSA is less memory intensive than docking so use cheaper VMs for those jobs

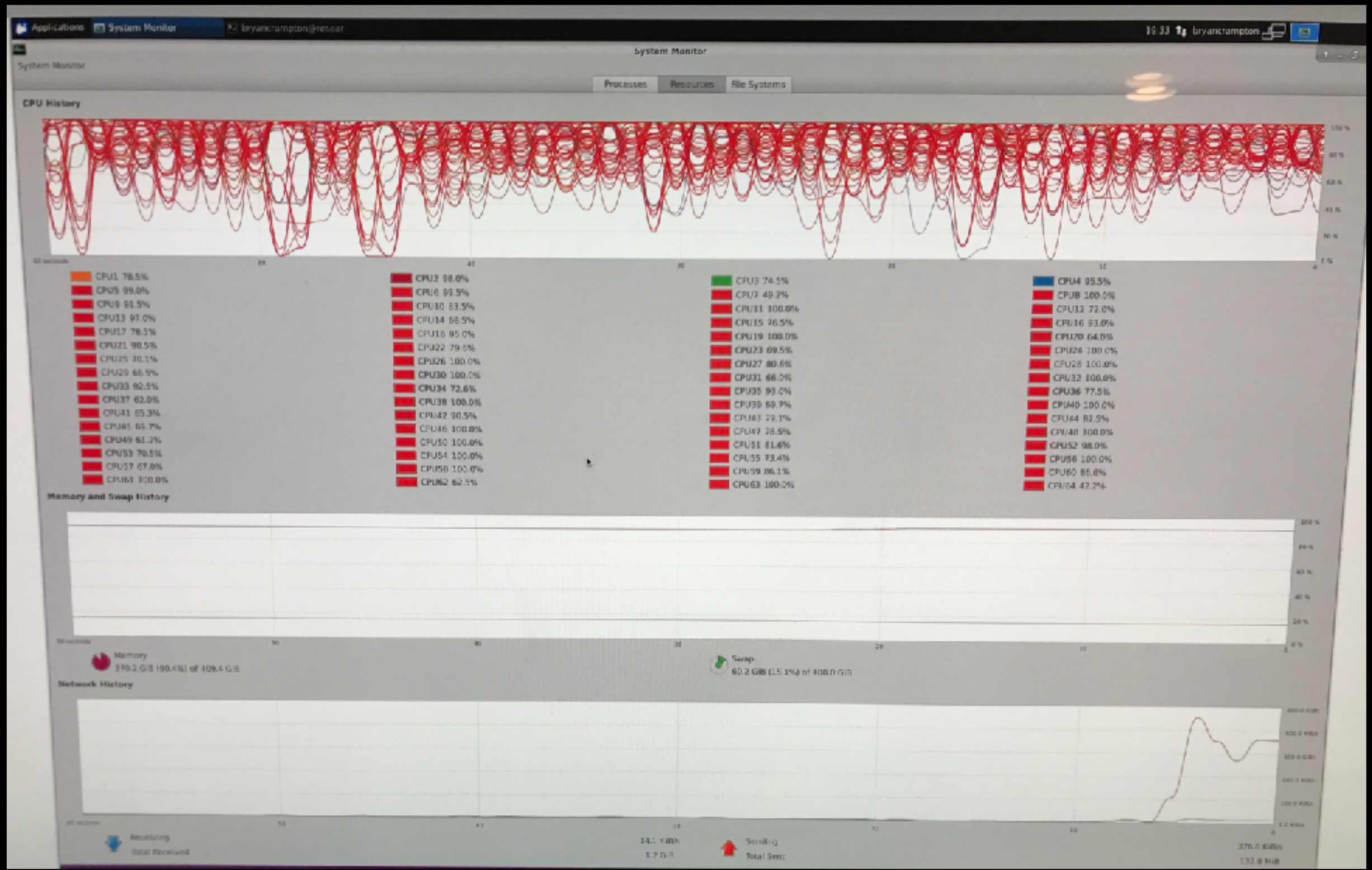
GOOGLE COMPUTE ENGINE

```
#!/usr/local/bin/bash

declare -A zones
declare -A cpu_counts
declare -A memory_counts

# Setup mapping of researcher # to zone
zones[1]="asia-east1-b"
cpu_counts[1]=32
memory_counts[1]=208GiB
zones[2]="asia-east1-b"
cpu_counts[2]=32
memory_counts[2]=208GiB
zones[3]="asia-northeast1-b"
cpu_counts[3]=64
memory_counts[3]=416GiB
zones[4]="europe-west1-b"
cpu_counts[4]=64
memory_counts[4]=416GiB
zones[5]="us-central1-b"
cpu_counts[5]=64
memory_counts[5]=416GiB
zones[6]="us-east1-b"
cpu_counts[6]=64
memory_counts[6]=416GiB
zones[7]="us-west1-b"
cpu_counts[7]=64
memory_counts[7]=416GiB

length=${#zones[@]}

for i in `seq 1 $length`; do
    name=researcher$i
    zone=${zones[$i]}
    cpus=${cpu_counts[$i]}
    memory=${memory_counts[$i]}

    echo "Creating instance: $name in zone: $zone CPU
    echo "gcloud compute instances create $name --dis
    gcloud compute instances create $name --image=res
done
```




```
#!/usr/local/bin/bash

if [ "$EUID" -eq 0 ]
  then echo "Please DO NOT run this as root"
  exit
fi

declare -A zones
declare -A addresses
declare -A cpu_counts

# Setup mapping of researcher # to zone
zones[1]="asia-east1-b"
cpu_counts[1]=32
zones[2]="asia-east1-b"
cpu_counts[2]=32
zones[3]="asia-northeast1-b"
cpu_counts[3]=64
zones[4]="europe-west1-b"
cpu_counts[4]=64
zones[5]="us-central1-b"
cpu_counts[5]=64
zones[6]="us-east1-b"
cpu_counts[6]=64
zones[7]="us-west1-b"
cpu_counts[7]=64

length=${#zones[@]}

for i in `seq 1 $length`; do
    name=researcher$i
    zone=${zones[$i]}

    output=$(gcloud compute instances describe $name --zone=$zone)
    address=$(echo $output | grep -Eow 'natIP: [0-9\.]+')
    address=${address#natIP: }
    addresses[$i]=$address

    echo "Got host: $name in zone: $zone with IP: $address"
done

# Generate and replace schrodinger.hosts file
host_file=""

for i in `seq 1 $length`; do
    name=researcher$i
    address=${addresses[$i]}
    cpus=${cpu_counts[$i]}

    host_entry="
# $name
name: $name
host: $address
processors: $cpus
tmpdir: /home/bryancrampton/scratch
schrodinger: /home/bryancrampton/schrodinger2017-1
"
    host_file+=$host_entry
done

# Replace hosts file
echo "$host_file"
```

GOOGLE COMPUTE ENGINE

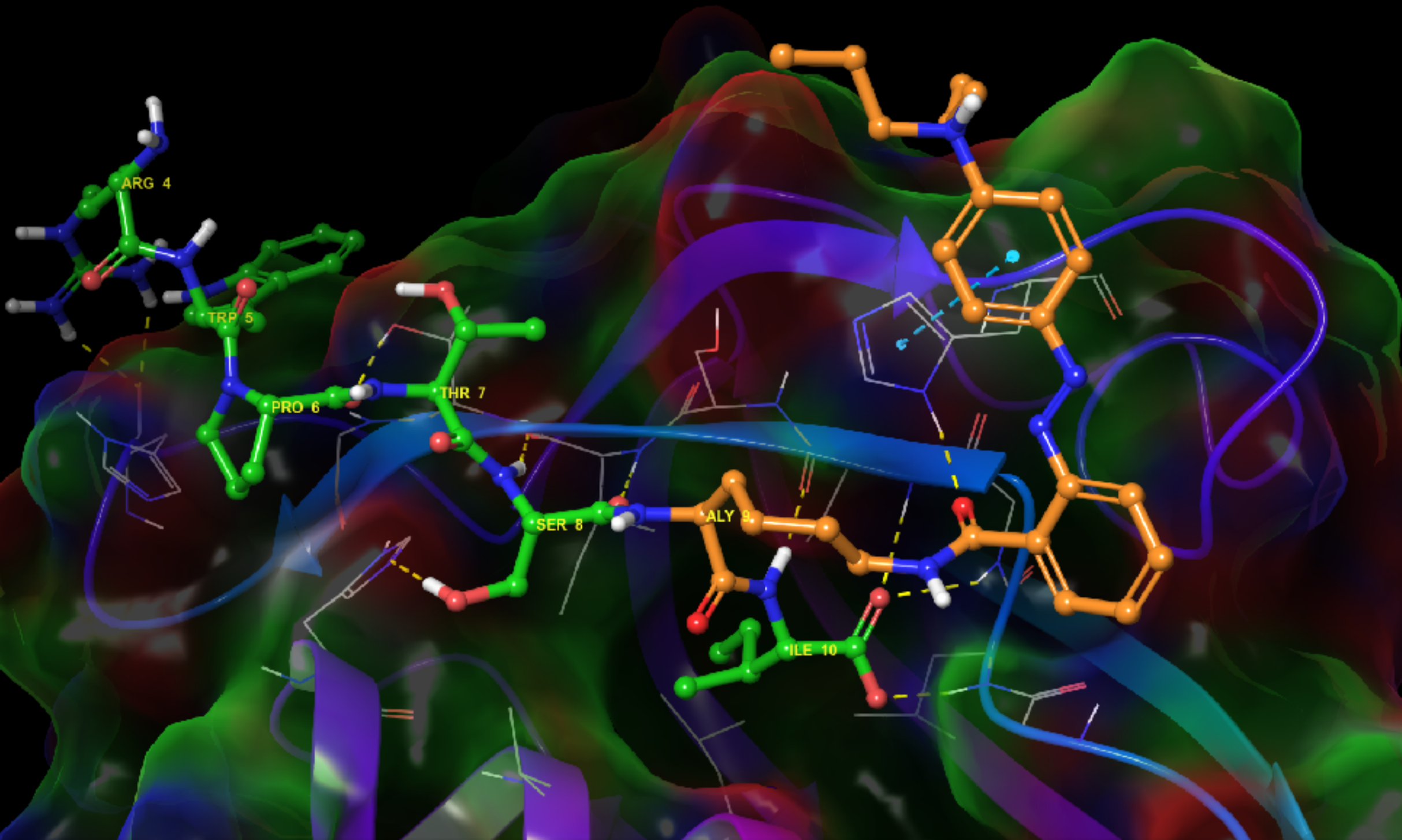


RESULTS!

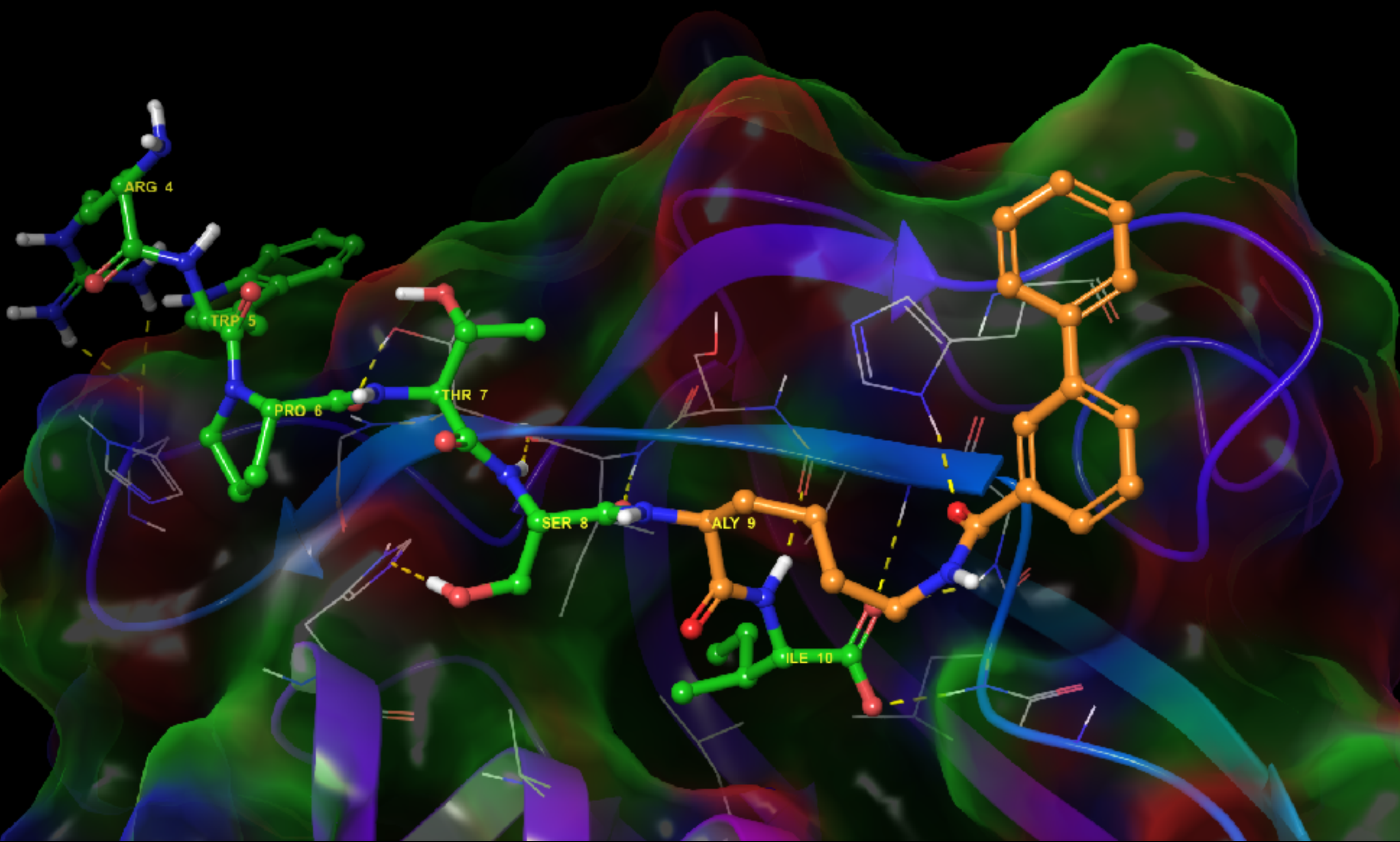
Title	glide gscore	glide emodel	MMGBSA dG Bind
▼ prime-mmgbsa-run2-out (12824)			
co prime-mmgbsa-run2-out (12824) S	-9.737	-141.060	-108.668
core; [molecule-cg118]	-9.905	-118.281	-108.545
core; [molecule-cg76]	-10.339	-130.746	-108.180
core; [molecule-cg127]	-9.539	-106.820	-108.048
core; [molecule-cg5]	-11.703	-132.281	-107.477
core; [molecule-cg122]	-9.803	-112.656	-107.406
core; [molecule-cg130]	-11.552	-131.284	-107.280
core; [molecule-cg8]	-10.000	-119.812	-107.082
core; [molecule-cg126]	-11.315	-127.971	-106.984
core; [molecule-cg117]	-9.907	-118.752	-106.771
core; [molecule-cg35]	-10.388	-119.824	-106.758
core; [molecule-cg101]	-11.628	-141.151	-106.444
core; [molecule-cg116]	-11.428	-134.590	-106.177
core; [molecule-cg48]	-10.567	-118.724	-106.174
core; [molecule-cg76]	-10.529	-146.134	-106.068
core; [molecule-cg48]	-10.235	-114.817	-106.038
core; [molecule-cg1]	-9.863	-126.037	-105.851
core; [molecule-cg122]	-10.547	-122.095	-105.795

RWPTS[Ac-K]I: -80.343

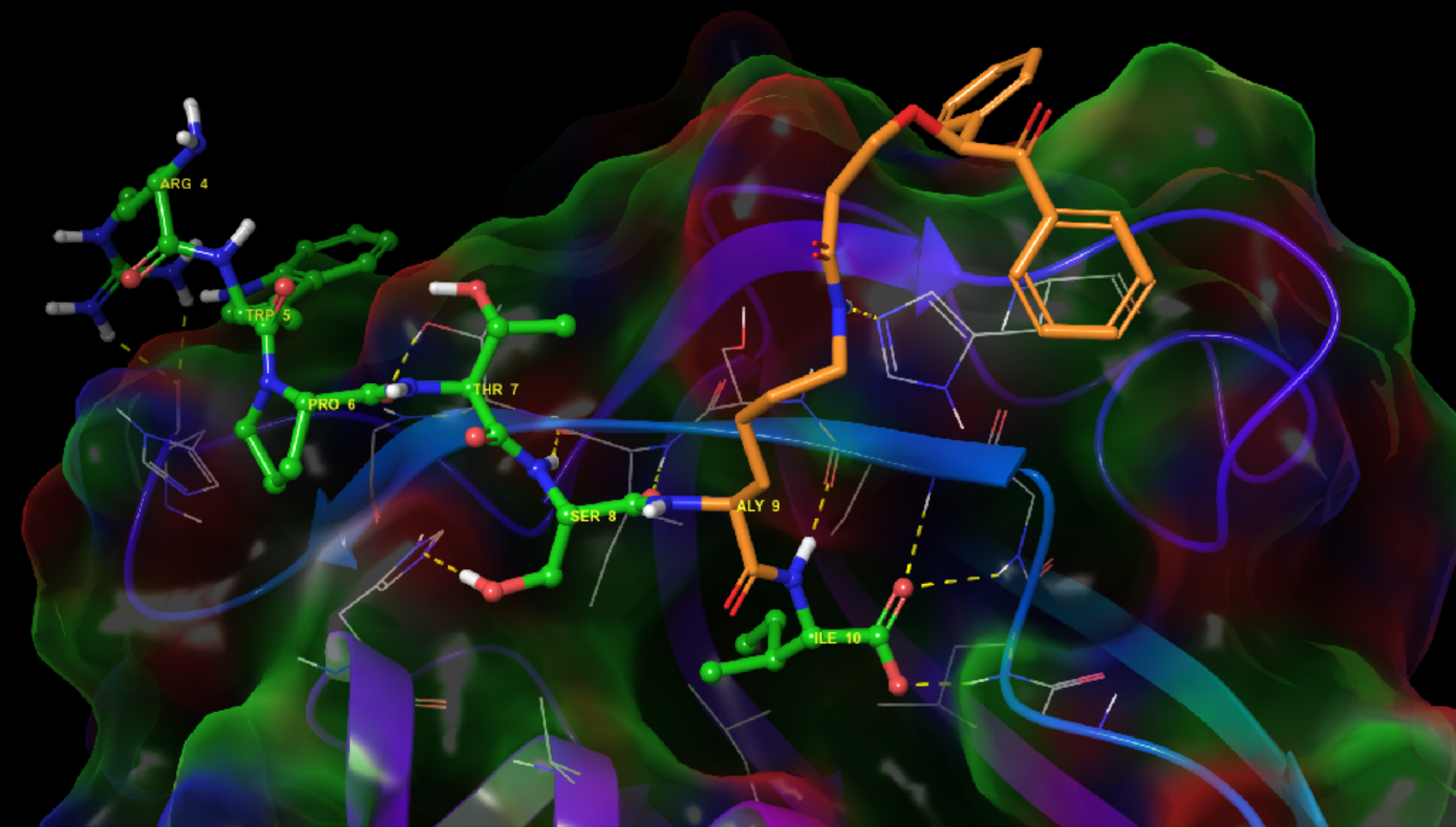
TOP HIT: #1



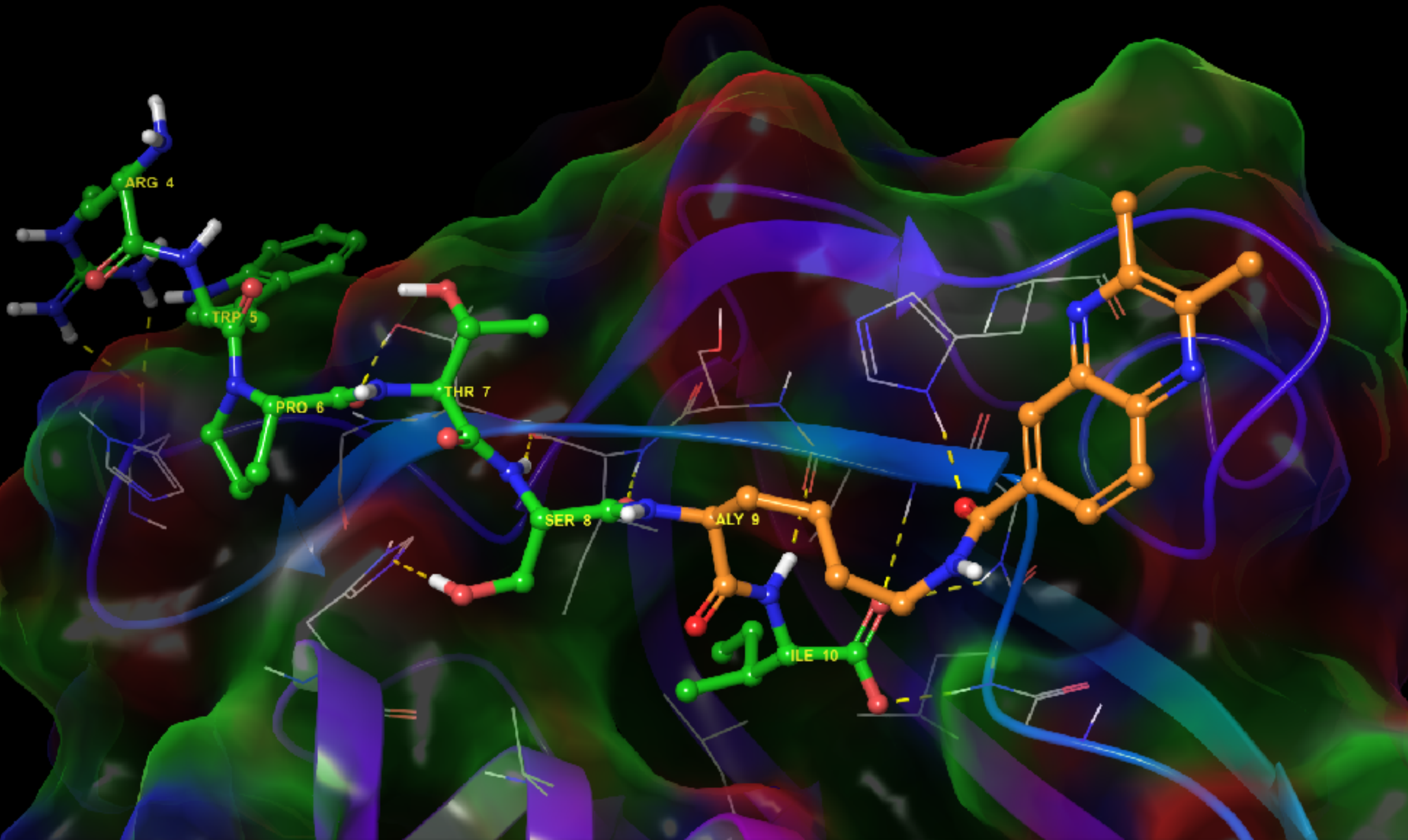
TOP HIT: #2



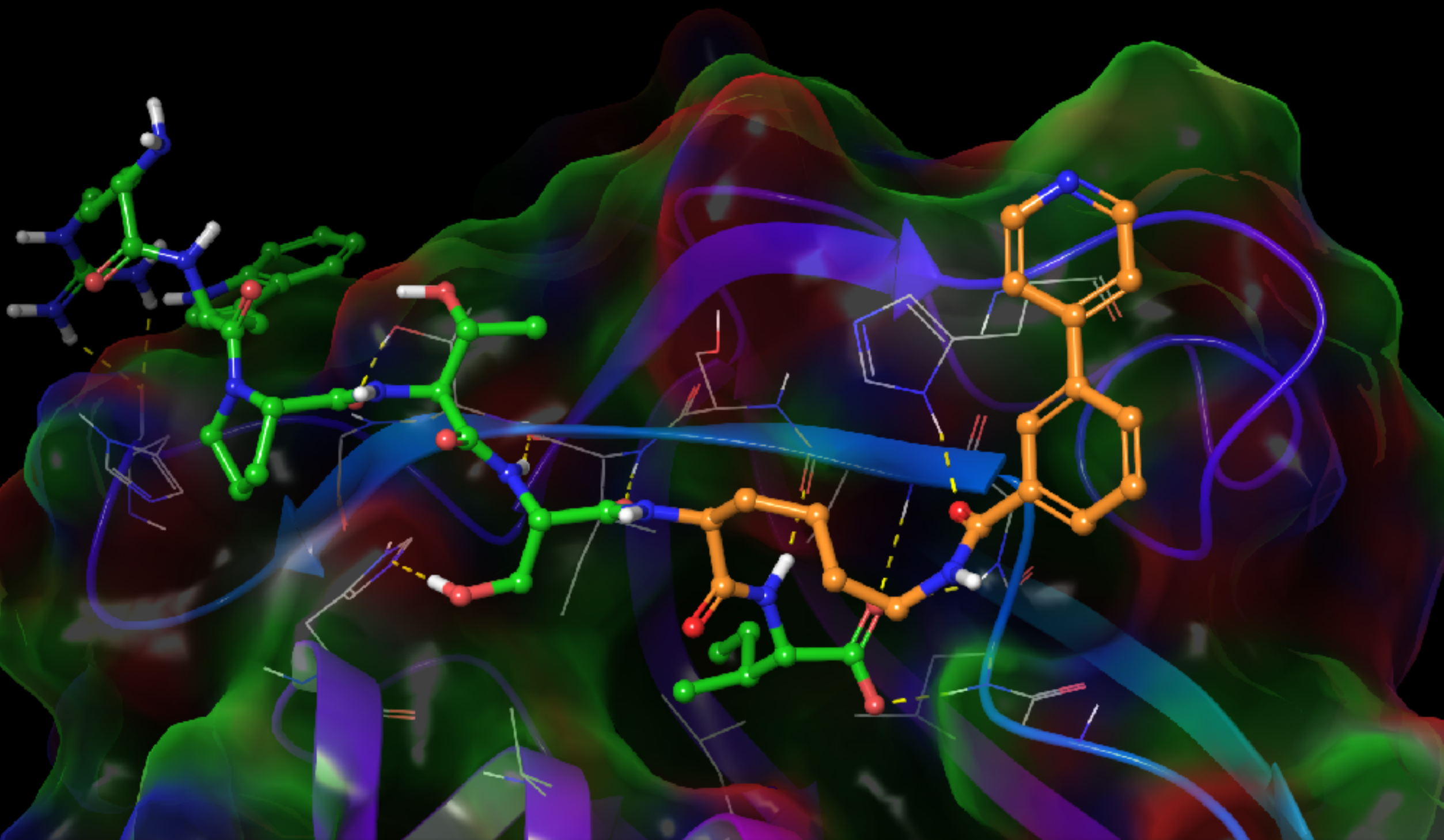
TOP HIT: #3



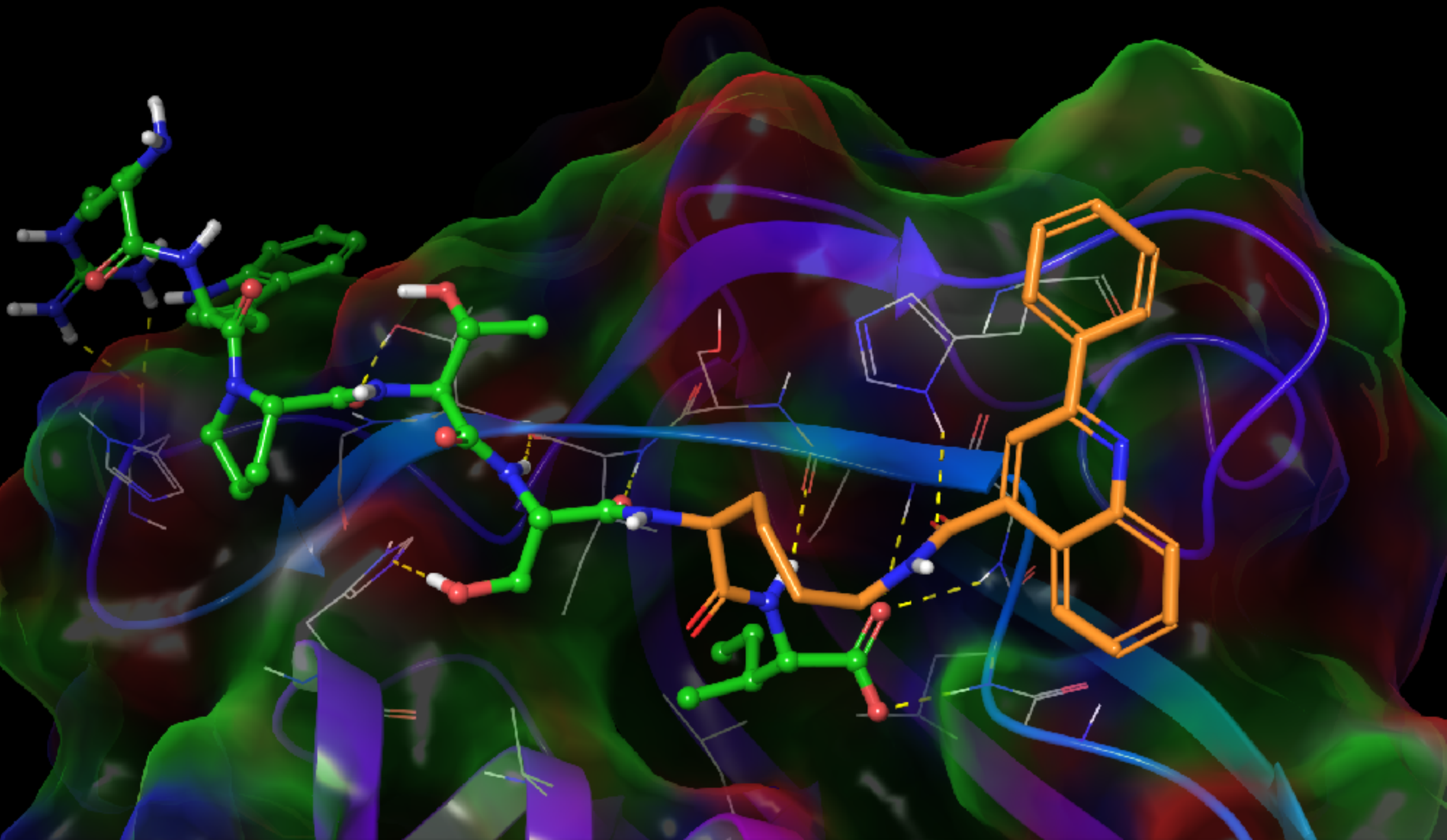
TOP HIT: #4



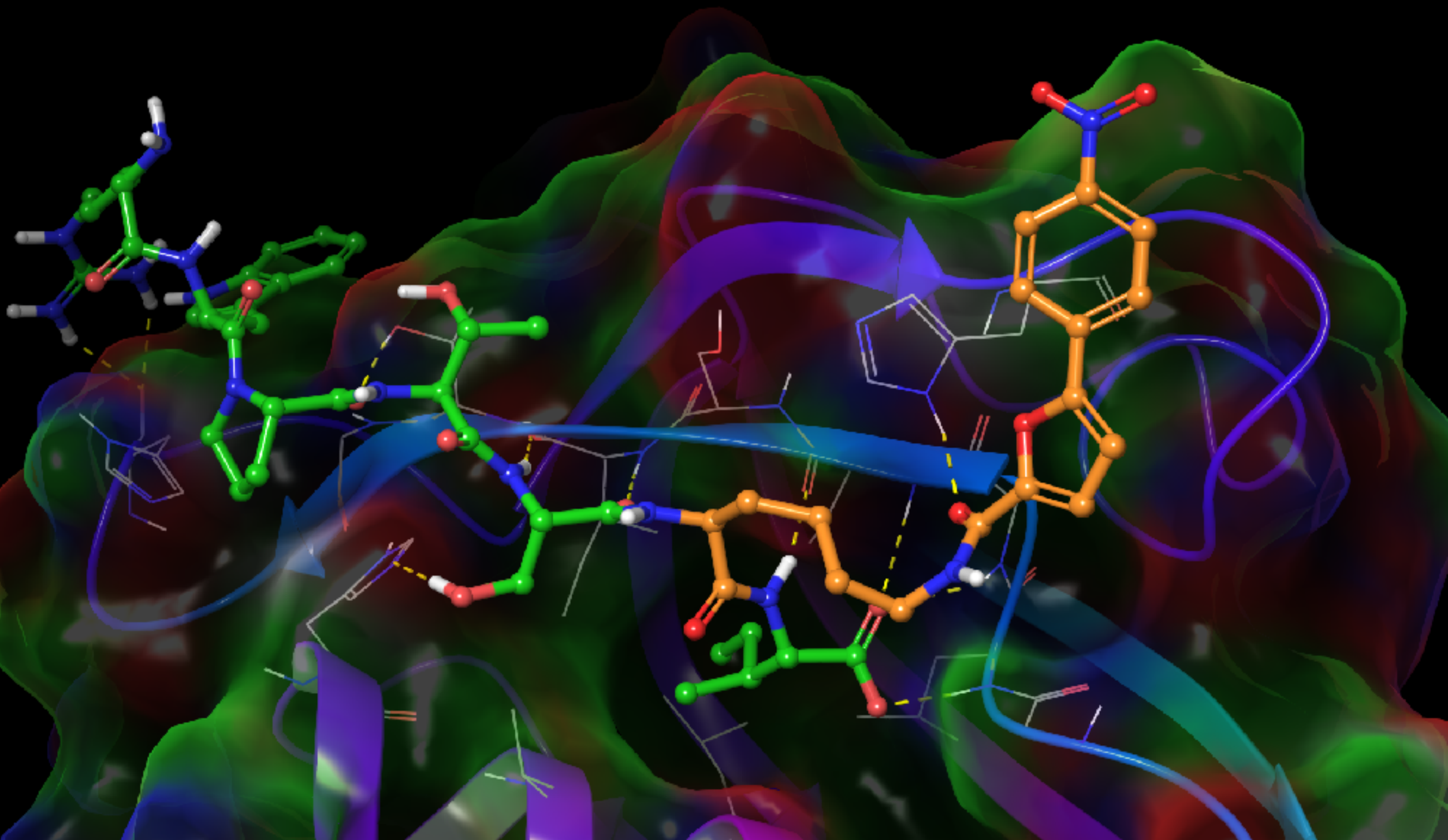
TOP HIT: #5



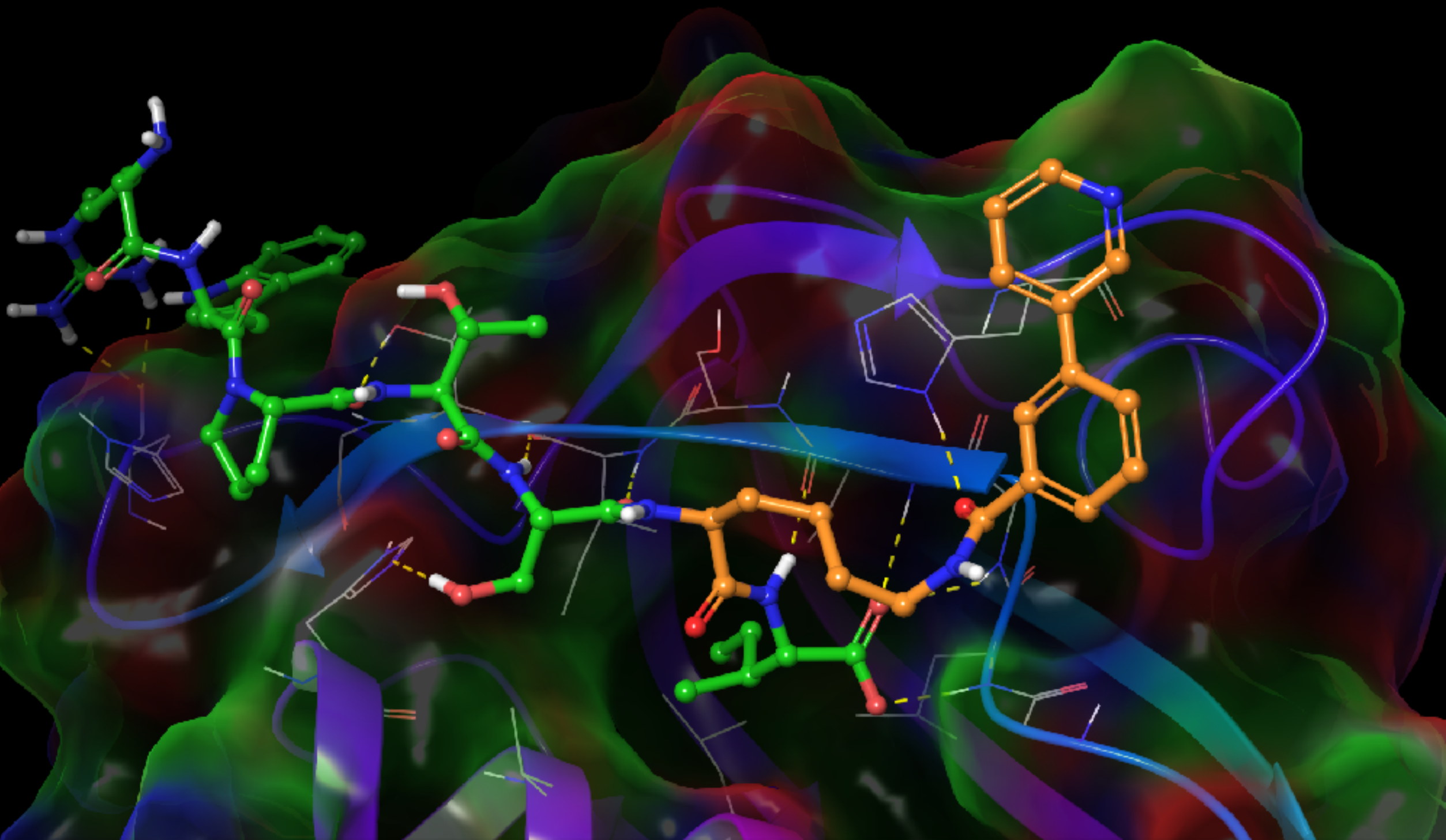
TOP HIT: #6



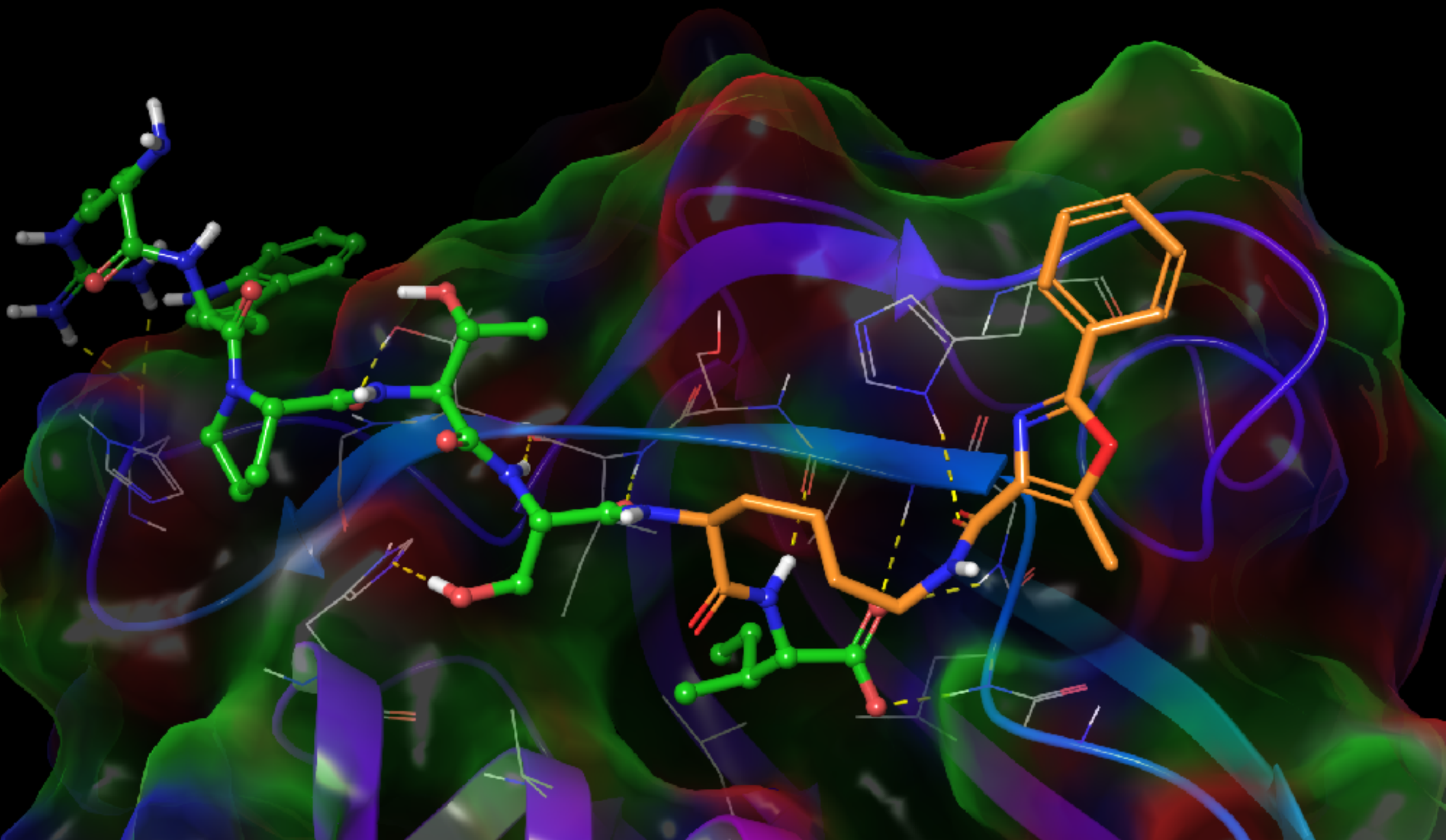
TOP HIT: #7



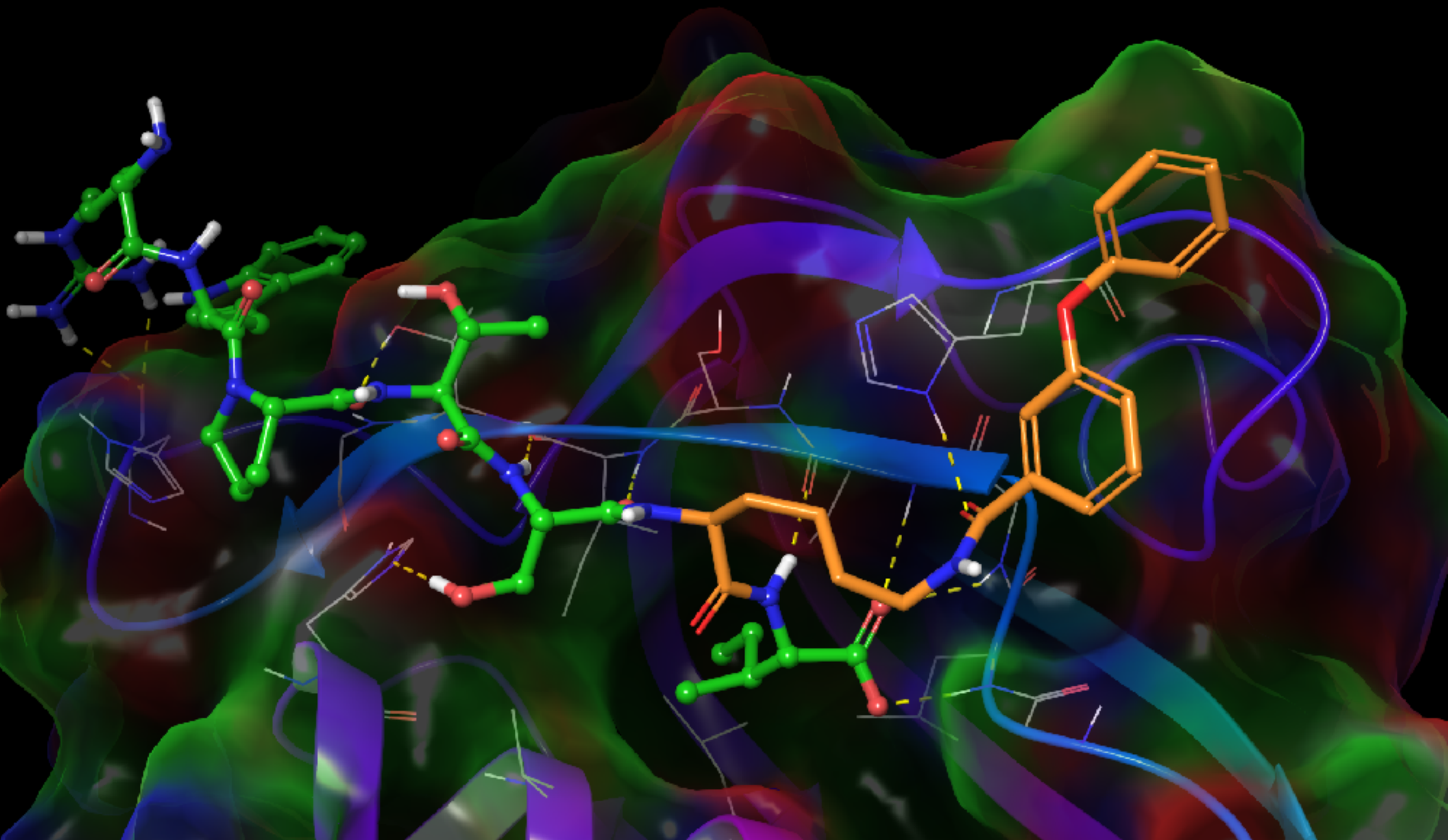
TOP HIT: #8



TOP HIT: #9



TOP HIT: #10



QUESTIONS