DOCK Blaster 2.0 -

An Investigation of Automated Docking

Ian S. Knight, Khanh G. Tang and John J. Irwin*

UCSF Department of Pharmaceutical Chemistry, 1700 4th St, San Francisco CA 94158-2330

* Corresponding author jji@cgl.ucsf.edu

Abstract

Molecular docking is widely used to leverage protein structure for ligand discovery, but the technique retains important liabilities that make it challenging to deploy on a large scale. Notwithstanding multiple attempts at automation, molecular docking continues to require the guidance of an expert thus limiting its use by many investigators who could benefit from it. To make docking more accessible we have created new software that allows us to investigate the automation of molecular docking screens. Our method currently requires known ligands and decoys for model evaluation. Of 42 DUDEZ targets, all show automated docking results that are better than our previous automated protocol. The new system is available both as part of the UCSF DOCK 3.8 package, which is free to academics, as well as via our website tldr.docking.org/start/dockopt (free registration required), which is free to everyone.

Introduction

Molecular docking screens are widely used for ligand discovery in industry and academia[1-3]. The technique can screen libraries of billions of molecules and - unlike ligand-based methods - can often discover novel ligands entirely unrelated to what was previously known.[2, 4-13]. In select cases docking can even lead to compounds in the sub-nM range[4, 5, 8, 10] and some of these lead to compounds with interesting in vivo activities[5, 8-10]. But whereas other techniques in computational biology such as homology modeling[14-16] and sequence database searching[17] have been successfully automated on a proteomic scale, docking remains manually intensive. Docking programs are challenging to use, with many parameters to be chosen, file formats to be manipulated, and decisions to be made, particularly at the model building stage. Even in expert hands, there are targets for which docking simply fails to recapitulate experimentally known binding information. These barriers to entry have diminished the impact of the technique by making it less accessible to biologically oriented non-experts and challenging even for specialists to deploy on a large scale.

Automation would allow docking to be deployed more widely, without the involvement of a docking expert. As a result, there have been numerous attempts at automated molecular docking over the past 14 years[18-20] and some docking services are available to use via websites[21-25] . However, although docking model optimization protocols are widely use by experts when preparing a receptor for a docking screen[26-30], such optimization is generally not exploited in automated pipelines. Instead, automated pipelines have focused on handling the many

mechanical steps in a docking campaign, without tackling model parameter optimization for the optimal enrichment of known actives over property matched decoys.

Our work on docking automation began in 2009 with DOCK Blaster[18], our first attempt at automated molecular docking. Although we successfully ran retrospective docking against thousands of targets, our program had important limitations. It was unable to optimize the parameters used for scoring as an expert in our lab would. It was unable to perform self-assessment automatically using annotated actives and decoys. The program was a prototype, cobbled together from many scripts, brittle and difficult to maintain let alone develop. DOCK Blaster was also too “black-box-ish” for experts yet somehow still too complex for non-experts.

Since the appearance of DOCK Blaster, many other web-based docking systems have appeared [31] [23, 32-34] including some that are designed for the scalability of the cloud[35-37]. There have also been numerous reports of software that is increasingly automatic, but is not available in a public, ready-to-use website[6, 37-46].

Meanwhile we have been improving our docking methods, trying to make docking easier to use. We re-wrote the command-line docking pipeline to be more modular and modern[47] and we standardized and published our lab docking protocol[26]. In both cases, docking model parameter optimization and model testing, which are often

critical to docking success, remained manual and dependent on expert involvement. Neither advance was accompanied by a website to simplify access.

To be most useful an automatic docking solution should optimize docking parameters to perform as well as an expert would if given the same information. This system should automatically assess how well the docking model is working by testing its ability to recapitulate known ligands and to rank them higher than property-matched or other decoys[48]. The preparation of the docking model should be easy for non-experts to access and still allow an expert to tinker with the model. And whereas it is unreasonable to expect a free public docking server to dock at multi-billion molecule scale, an ideal system should make it easy to transition to a pay-for-scale system so that non-expert users can access large screening libraries at their own expense.

Here we investigate a new automated docking system, DOCK Blaster 2.0, benchmarking it against the DUDEZ benchmark[48].

Results

New software for automatic docking is now available as part of UCSF DOCK 3.8. This software is available free to academics (license via dock.compbio.ucsf.edu) and at modest cost otherwise (write dock_industry@googlegroups.com). **First**, we describe the software and how to use it on the command line. **Second**, we test the software retrospectively against 42 targets in the DUDEZ benchmark[48], investigating its performance, generality and limitations. **Third**, we present a web-based interface to this software that can build and optimize a docking model using self-assessment, followed by prospective docking to a small set of molecules. The docking model built using our webserver may be downloaded and deployed to other platforms such as the user's on-premises computers or to a cloud provider such as Amazon's AWS[49] to dock billions of molecules prospectively at the user's expense. We take up each of these in turn.

Dockopt. We created a new software pipeline that combines, condenses and refines what we learned from three prior projects: the original DOCK Blaster[18], the command line script blastermaster.py in DOCK 3.7[47] and the standard published lab protocol[26]. The new system is written in python 3.8 and is documented. (see **Supporting Information** S1). We describe the algorithm in detail in the **Methods**.

`Dockopt` is a single command for generating and evaluating many different docking configurations when control ligands are available. A docking configuration is a set of ready-to-dock files that describe a receptor structure and associated

parameter choices. Docking configurations may perform better or worse as evaluated by retrospective docking, where the ability of the docking program to distinguish between reported binders ("actives") and presumed non-binders ("decoys") is assessed. `Dockopt` wraps the entire parameter generation, testing and optimization protocol within a single script.

There are over a dozen parameters whose values can affect the docking configurations that are produced by `dockopt`, but a few are typically most impactful. Important parameters include the thickness of the layer of low dielectric and ligand desolvation regions in the binding site that affect the electrostatic score and the ligand desolvation calculation, respectively[50]. Other important parameters include the number and position of orientation spheres ("hotspots"), which affect how ligand poses in the binding site are generated[51]. Still other significant parameters include the target number of poses to generate (nmatch), how overlap between ligand and protein is treated (bump parameters), and whether conformations are biased for compatible matches during sampling (called coloring)[52, 53].

Retrospective docking of enumerated docking configurations is performed in parallel using a job scheduler such as Slurm, SGE or GNU parallel. It should be straightforward to adapt our scripts to any queuing system. When finished, the enrichment capacity and the parameters of each docking configuration are tabulated and sorted to identify the best performing configuration(s). Depending on choices made by the user, the optimization process may repeat until optimal parameters have

been identified. A report in PDF format of the best parameter set choices is generated, together with figures summarizing the other runs (see **Methods**, and below). A text file of the performance of all docking configurations is also saved as `results.csv`. The best job's dockfiles (docking configuration) are saved in its own directory. If other docking configurations are desired, they may be found in the `working/` directory, indexed within `results.csv`. In the absence of control ligands , `blastermaster`, a modernization of blastermaster.py from earlier work[47], will produce files ready for docking using standard, unoptimized parameter choices.

Getting Started. To begin using the new code on your own computer, obtain, download, install and configure the software (see **Supporting Information** S2).

To use `dockopt`, the receptor and a single ligand or an indication of the binding site should be prepared as we recommend. (see **Supporting Information** S3). The DUDEZ benchmark contains forty-two examples of such files in ready-to-use formats (dudez2022.docking.org). To use `dockopt` on the command line, prepare rec.crg.pdb and xtal-lig.pdb, the receptor and crystallographic ligand, in an empty directory, together with actives.tgz and decoys.tgz, the annotated ligands and their decoys in ready-to-dock format (see **Supporting Information** S4). The user enters:

```
pydock3 dockopt – init
cd dockopt_job
pydock3 dockopt – run slurm
```

The program runs, here using slurm, preparing a report.

How well does Dockopt work? To benchmark the `dockopt` program, we ran forty-two DUDEZ benchmarks, using the data available on the DUDEZ website (dudez2022.docking.org). The resulting enrichment scores are all better than the enrichment using the docking grids that were used for development of the DUDEZ benchmark(**Figure 1**). The procedure for acquiring the benchmark is:

```
git clone https://github.com/docking-org/dude-z-benchmark
cd dude-z-benchmark
bash make_dataset.sh
```

To run a single benchmark, e.g. AMPC, in the `property_matched` directory:

```
bash run_benchmark.sh AMPC
```

To run all benchmarks,

```
bash run_all_benchmarks.sh
```

The enrichment of annotated ligands over property-matched decoys produced by the default automated procedure coded in `dockopt` are better for 40 of the 42 unoptimized grids from the DUDEZ benchmarking study[48] (**Figure 1**), whose parameters were mostly based on defaults from our previous blastermaster.py[47, 48] protocol. On average, we saw a 5 to 10% increase in enrichment score [54] following parameter optimization.

Figure 1. Performance of molecular docking as measured by enrichment of ligands over property matched decoys for 42 DUDEZ systems. Above: Docking configurations published in the DUDEZ paper[48] (blue) with docking configurations optimized by dockopt (orange, this work) Below. Improvement of enrichment using dockopt over the grids published with DUDEZ. In all but two cases the enrichment is improved using `dockopt`.

`Dockopt` produces a report for each target, and these are included in the dudez2022.docking.org website for reference. Here, we use two targets for illustration

and to demonstrate the features of the report (**Figure 2**). The Linear-log ROC plot of enrichment of ligands and decoys (**Figure 2A**) assesses the performance of a docking model reduced to a single graph and a single value, the area under the curve (AUC). The Dockopt report includes additional graphs that provide additional insight into the performance of the docking calculation, and into biases present in the dataset. Thus the split violin charts (**Figure 2B**) show the scores of actives and decoys grouped by net molecular charge. This graph, used in the construction of the DUDEZ benchmark, provides insight into the discrimination power of the docking model, in its ability to rank actives better than decoys, where each molecular charge is considered independently. The ridgeline plots (**Figure 2C**) show how actives and decoys are ranked by each contribution to the scoring function. These plots illustrate potential biases in the data or biases in the model parameterization that could be a problem for a prospective screen. Too vague. Re-write. Finally, the heatmaps summarize the dependence of the enrichment as a function of two variables: the electrostatic spheres (thin layer) and the desolvation spheres (thin boundary). Here, the enrichment of HIVPR is uniformly poor regardless of these two variables, whereas in Factor 7A, there is a sharp dependence on these parameters and a clear winner may be identified. Taken together, these charts, all generated automatically by Dockopt, may be used to gain insight into a docking model, its biases, and its likelihood of success for prospective docking screens.

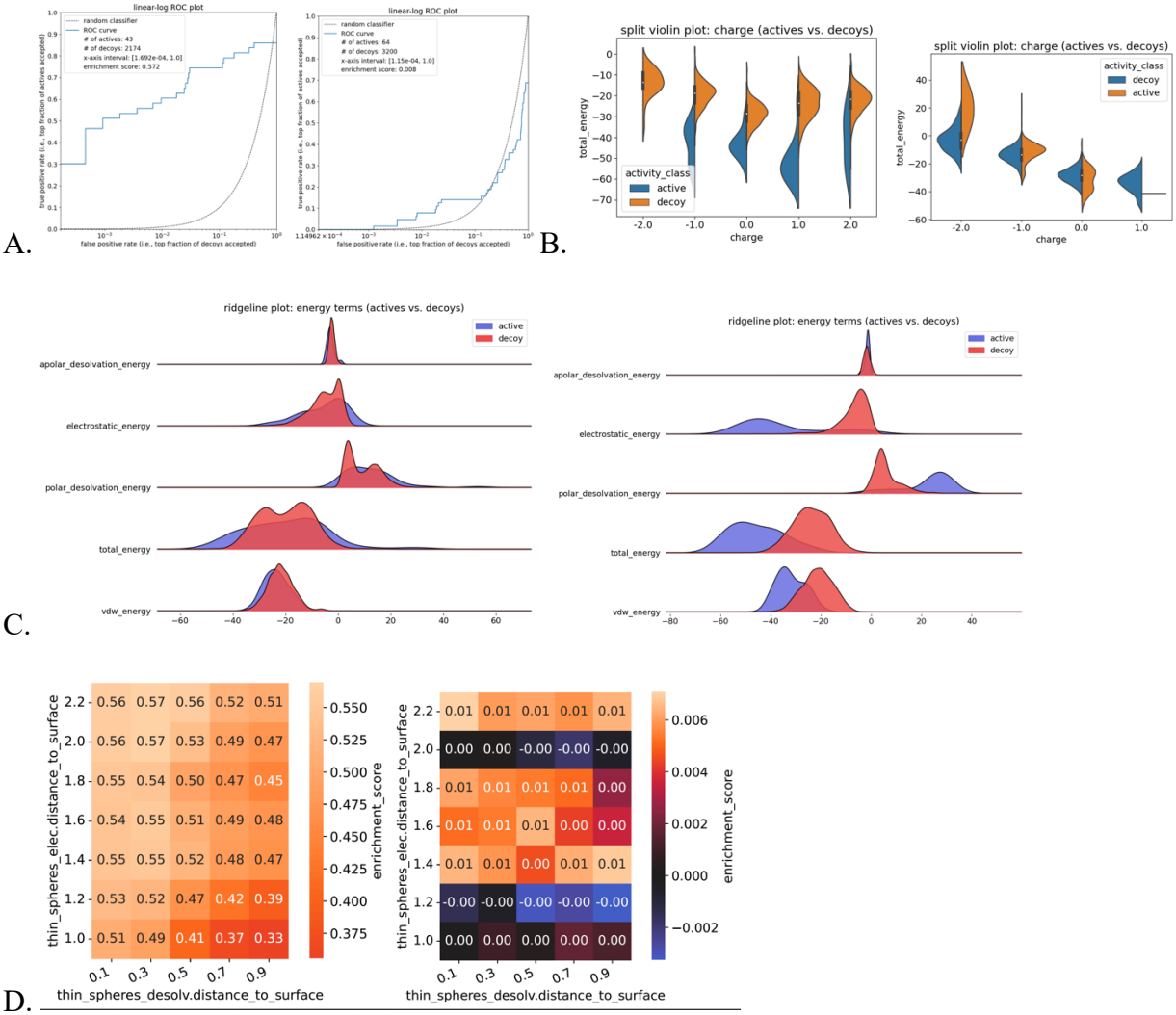


Figure 2. Selected graphical reports of docking models optimized for two targets. Left: FA7. Right: HIVPR. Linear-log ROC plot of enrichment of ligands vs decoys. B. Violin plots of the distribution of charges. C. Unidimensional plots of the distribution energy terms of docked ligands and decoys. D. Bidimensional plots of enrichment as function of electrostatic and desolvation radii.

A complete set of enrichment plots for all DUDEZ systems may be found at dude2022.docking.org.

Enrichment without the correct pose is erroneous. Whereas we do not know the pose of every active ligand, in general, we expect it to be in the same region as the crystallographic ligand, and often to make many of the same interactions with the receptor. We capture this in a chimera session that superposes the receptor, the crystallographic ligand and the docked ligands (**Figure 3**). Sometimes, the superposition is excellent, controlled by a single "warhead" as in HIV protease. Other times, the superposition may be less obvious, but at least we can check that the docked molecules occupy the same binding site.

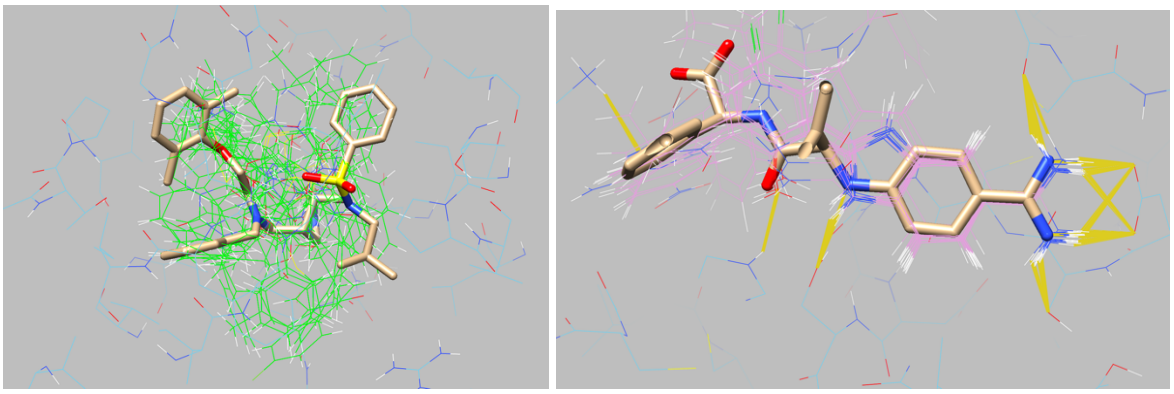


Figure 3. Superposition of the crystallographic ligand (sticks) with the docked ligands (wire) showing H-bonding and polar interactions with the protein in mustard. Left. HIVPR. Right: FA7.

Since the automated system appears to work well at least in some cases, and we can tell which based on retrospective tests, we built a web-based interface to run

automated docking. To use this interface (**Figure 4**), the user should browse to tldr.docking.org (login required), select the dockopt module. To demonstrate its use we suggest the use of one of the 40 test systems at dudez2022.docking.org.

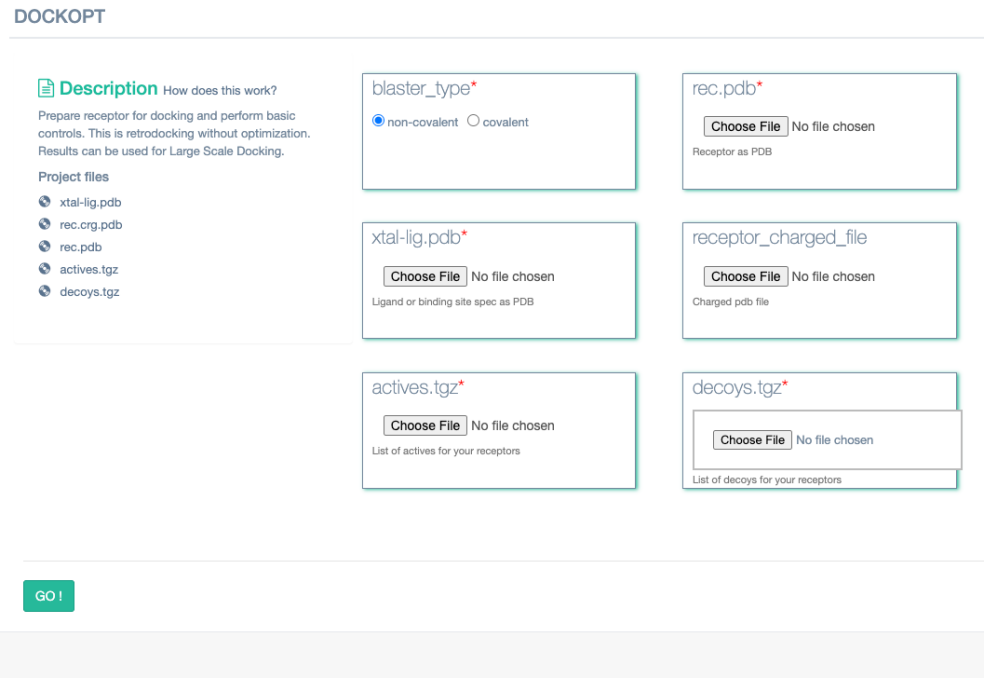


Figure 4. A web interface to dockopt. (free registration required).

When the receptor, ligand, actives and decoys have been specified, the user clicks “Dock” to start the automatic docking process. Building an optimized docking model can take a day, depending on the data supplied, how many other jobs are running on our server. When docking is complete, the user receives an email invitation to login to tldr.docking.org to view the results. The user may download the report, which includes bar plots of single multi-valued parameters, heatmaps of pairs of multi-valued parameters, charge distributions in violin plots, energy terms ridge plot, as well as the docked ligands and decoys themselves (**Figure 4**). We allow the user

to test-dock a set of 100,000 sample molecules. To dock at scale, the user should download the best docking model model, and use it on other hardware, such as in the AWS cloud [49]. The user may also download the docking model and relocate it to a departmental cluster, or the cloud, where a full-scale docking screen may be prosecuted.

The system can even - mechanically - dock without any experimentally known ligands, but here it wanders into the same terrain that a human would. Without controls, it is difficult to assess how well docking is performing, short of a prospective screen. Blastermaster uses sensible average values for parameters that would be optimized by DockOpt. To use blastermaster on the command line, put rec.crg.pdb and xtal-lig.pdb into a directory and run:

```
blastermaster
```

The program runs. Blastermaster uses average default values for all parameters that would normally be optimized by DockOpt, but cannot, given the lack of control ligands.

DOCK Blaster 2.0, a method that includes `dockopt` for parameter optimization and Blastermaster if no control ligands are available is now ready to use. The software may be accessed either by licensing, downloading and installing it on your local

computer or in the cloud, or by using it via the TLDR website on our server (tldr.docking.org) as described.

Discussion

Three themes emerge from this study. **First**, a new automated procedure for docking model optimization has been developed. Provided ligands and decoys are available for self-assessment, this automated procedure can optimize the parameters for docking as well as an expert and can self-assess whether this model warrants a large-scale prospective docking screen. **Second**, when tested against the DUDEZ benchmark, the automated protocol performs as well as an expert from our lab in most cases. In over half the cases studied, a single control ligand proved sufficient for model optimization and assessment. **Third**, the new pipeline can be installed locally, or accessed in a web browser. Although the website offers only very limited prospective docking services, the docking model may be downloaded and used for large scale docking in the cloud [49]. We take up each of these themes in turn.

An automated docking parameter optimization is now available, implementing many of the best practices embodied in our standard protocol[26]. We have augmented this protocol with optimization steps that capture many of the current best practices in our lab. In particular, the system optimizes parameters for the boundary of the region of low dielectric and ligand desolvation in the binding site. Work in our lab suggests that defining these boundaries are often critical to obtaining satisfactory retrospective enrichment during benchmarking. This same protocol is now available, fully automatic, to everyone via the `dockopt` script. The procedure also optimizes the matching spheres used for sampling ligand orientations. Although these parameters

are not used in the scoring function per se, they affect which configurations are evaluated, and thus often play a key role in obtaining satisfactory retrospective enrichment prior to running a successful docking campaign. The software has been written in a general way to allow other parameters to be optimized.

Our automated docking performs well against the DUDEZ benchmark we use regularly in our lab. In all but perhaps six of the 42 DUDEZ systems, this fully automated procedure yields a docking model that, in our experience, is suitable for prospective docking. These failures present opportunities for further improvements to our protocol.

Since model building and assessment is now fully automatic when control ligands are available, we have built a web interface, which is now available. The new system is capable of building and refining a docking model completely automatically and can evaluate the model's suitability for prospective use when it is finished. The method currently requires known actives to use for automated evaluation. Without known actives, we are unable to perform sanity checks as to whether prospective docking is likely to highly rank new ligands that actually bind or not.

There are numerous caveats to this work. The current protocol requires ligands – or at least one ligand - with which to optimize the docking model and later to score it for enrichment success. Absent ligand controls, we are currently unable to optimize the docking model, and certainly unable to say whether it should be useful for

prospective docking. This protocol does not work well on every target. Against the 42 DUDEZ targets, it appears to work well for perhaps 80% of systems tried, but this is no general guarantee, and the reader should not be misled that this approach is in any way a universal solution to automated docking.

Notwithstanding these concerns, fully automated docking software with automatic self-evaluation and parameter optimization is now available in DOCK 3.8 and via our website. The software is available free to academics and can be used via our website by everyone. We cannot guarantee the results of any docking screen – you must use the software at your own risk. We strongly recommend running sanity checks and controls at every stage, as exemplified in this work, to increase the opportunities for new ligand discovery. The software is available to use online at tldr.docking.org (authentication required) and license requests at dock.compbio.ucsf.edu.

Acknowledgements

We thank NIH for support via GM133836 (to J.J.I.). We thank OpenEye Scientific Software for a license to Omega and OEChem. We thank ChemAxon, Schrodinger, Molinspiration, Molecular Networks, and NextMove Software for software licenses, support and collaboration.

Conflict of Interest Statement

J.J.I. is a founder of Deep Apple Therapeutics Inc, as well as a founder of Blue Dolphin Lead Discovery LLC, a contract research organization providing fee-for-service docking services.

The ORCID of J.J.I. is 0000-0002-1195-6417

Software and Data Availability Statement.

The software `dockopt` and `blastermaster` are part of DOCK 3.8 which is distributed using a UCSF license. Free to academics (dock.compbio.ucsf.edu for a free license and download), modest cost to for-profit users. (dock_industry@googlegroups.com).

All data, including the DUDEZ benchmark, and all results reported in this paper, are available from dudez2022.docking.org. This data is available via a CC BY 4.0 license. Software and data are available on github.com/docking-org/ as follows: dude-z-benchmark – benchmarking data; pydock3 – dockopt, blastermaster; SUBDOCK – software for running jobs on clusters using queues; zinc22-3d – the ZINC-22 3D database preparation pipeline.

Methods

All software is written in Python 3.8 or later. All software libraries used within our code are defined in the pyproject.toml file of the pydock python package in the DOCK 3.8 software distribution.

From Blastermaster to DockOpt. Blastermaster is a program that uses computational methods to generate DOCK parametrizations for a given protein-ligand complex. DockOpt is an extension of Blastermaster that takes the process one step further by creating several DOCK configurations simultaneously and using retrospective docking to evaluate their performance. This allows DockOpt to identify the best parametrizations according to a specified criterion, here enrichment. DockOpt uses a specified job scheduler, such as Slurm, to efficiently evaluate candidate parametrizations in parallel. In summary, DockOpt is a program that improves upon Blastermaster by adding model testing and to the process of crafting DOCK parametrizations and by efficiently searching for the best parameterizations using parallel processing.

DockOpt algorithm (high-level). A high-level description of how the DockOpt algorithm works is summarized in a schematic (**Figure 5**)

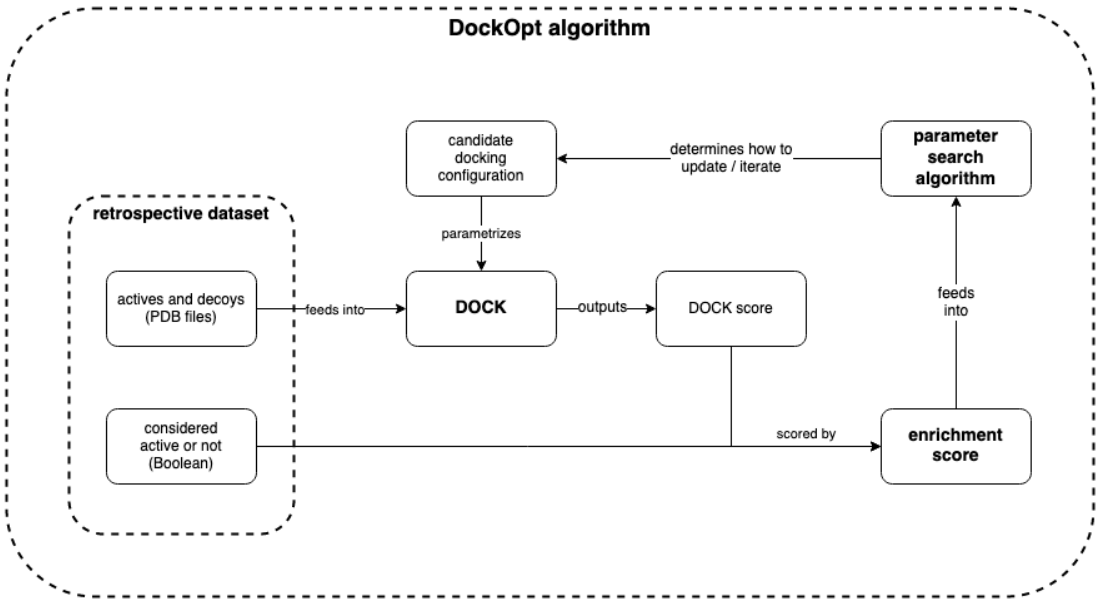


Figure 5. Schematic of the DockOpt algorithm.

How `dockopt` creates DOCK configurations. A directed acyclic graph (DAG) defines how input files and input parameters are transformed into the output files, called "dockfiles" which are actually used by DOCK for docking. A toy example of a such a DAG is given in the **Supporting Information** S5. The user may generate a DAG for any particular `dockopt` run using the command:

Command_to_generate_DAG. An edge in the DAG represents a dependence relation between an input and an output for a specific step in the pipeline. Every edge has data associated with it that points to the software that executes the step that produces the corresponding child node (output). Child nodes can only be created when all their parent nodes exist already.

The full DAG is derived automatically from all the different sequences of steps induced by all the combinations of parameters specified for the given DockOpt pipeline step.

Two parameter search algorithms are supported: grid search and beam search. Beam search, the default, is a heuristic search that sacrifices the exhaustive nature of grid search for significantly shorter run times. It is recommended.

In its first instantiation, `dockopt` performed simple grid search of the Cartesian product of several parameters which take multiple values. E.g., distance to surface for electrostatic thin spheres might be in the range (1.0, 1.1, …, 1.9) and distance to surface for ligand desolvation thin spheres might be in the range (0.1, 0.2, …, 1.0), resulting in a Cartesian product space of 10 * 10 = 100 combinations. This worked well enough for small numbers of parameters with a reasonable number of values per parameter. However, it clearly is too inefficient to serve as a general search algorithm that would allow users to explore parameter space at a fine resolution without being subjected to exponentially increasing computational cost. Therefore, the current instantiation of `dockopt` uses beam search[55] instead to efficiently search for DOCK parametrizations.

Paragraph describing beam search (Ian ???).

Below, we show that beam search allows one to find better DOCK parametrizations in practice than by using grid search.

DockOpt pipelines are flexible. The software framework in dockopt is far more general than the default usage case might imply. We recommend that new users adopt our default configuration or something similar. The system is capable of far more than it is currently being used for.

Dockopt is controlled by parameters in the dockopt_config.yaml file. This is where the users can control the range of parameters that should be explored and define the architecture of the optimization pipeline.
A single step in dockopt generated several parameterizations that can are created and evaluated in parallel. The user may use dockopt_config.yaml to define a sequence of steps, including iteration and early stopping. Sequences may be recursively embedded.

Ian writes a paragraph about all the paramters that can be controlled in a single step.

Each step may use the best parameterization from the preceeding step as the seed for determinating what new range of parameter space to search. For example, evaluating increasingly narrow windows of values around a center value of the same parameter across steps.

DockOpt is modular. The DAG is derived automatically from the dockopt_config.yaml. Steps may be modified and new ones introduced without additional complexity. The DockOpt pipeline consists of pre-defined sequences of steps. Individual sequences or steps may be defined once and re-used and optionally modified to form new ones. Different DOCK executables may be used / tested against each other. Different evaulation criteria may be used in different steps of the same DockOpt pipeline (e.g. enrichment first, RMSD last).

DockOpt allows algorithmic experimentation. A core feature of DockOpt is that it allows the rigorous comparison of several DOCK parametrizations and even several different DOCK executables. Therefore, an entire experiment intended to measure the efficacy of a modification to one or several variables can be defined in a single DockOpt job and reproduced at a later time simply by re-running the job. DockOpt makes benchmarking incredibly easy.

DockOpt Reports. DockOpt generates comprehensive reporting of the results of a job, including a CSV of results for each DOCK parametrization tested and a report PDF containing: ROC plots showing enrichment; Bar plots for performance of single multi-valued parameters; Heat maps comparing performances across 2 multi-valued parameters with respect to each other; Joy plot showing the importance of energy terms for actives vs. decoys; Violin plot showing the charge distribution for actives vs. Decoys

How to run the benchmarks.

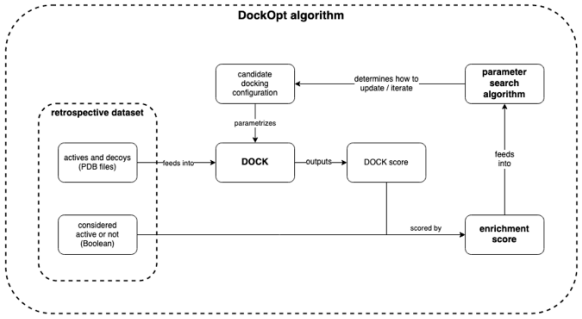
(describe wget/git clone)

run on our machine

get results.

Literature Cited.

TOC Graphic (maybe simplified for miniaturization)



1. Grebner, C.; Malmerberg, E.; Shewmaker, A.; Batista, J.; Nicholls, A.; Sadowski, J., Virtual Screening in the Cloud: How Big Is Big Enough? *J. Chem. Inf. Model* **2020**, 60, 4274-4282.

2. Zhu, H.; Zhang, Y.; Li, W.; Huang, N., A Comprehensive Survey of Prospective Structure-Based Virtual Screening for Early Drug Discovery in the Past Fifteen Years. *Int J Mol Sci* **2022**, 23.

3. Muegge, I.; Bergner, A.; Kriegl, J. M., Computer-Aided Drug Design at Boehringer Ingelheim. *J. Comput.-Aided Mol. Des.* **2017**, 31, 275-285.

4. Lyu, J.; Wang, S.; Balius, T. E.; Singh, I.; Levit, A.; Moroz, Y. S.; O'Meara, M. J.; Che, T.; Algaa, E.; Tolmachova, K.; Tolmachev, A. A.; Shoichet, B. K.; Roth, B. L.; Irwin, J. J., Ultra-Large Library Docking for Discovering New Chemotypes. *Nature* **2019**, 566, 224-229.

5. Stein, R. M.; Kang, H. J.; McCorvy, J. D.; Glatfelter, G. C.; Jones, A. J.; Che, T.; Slocum, S.; Huang, X. P.; Savych, O.; Moroz, Y. S.; Stauch, B.; Johansson, L. C.; Cherezov, V.; Kenakin, T.; Irwin, J. J.; Shoichet, B. K.; Roth, B. L.; Dubocovich, M. L., Virtual Discovery of Melatonin Receptor Ligands to Modulate Circadian Rhythms. *Nature* **2020**, 579, 609-614.

6. Gorgulla, C.; Boeszoermenyi, A.; Wang, Z. F.; Fischer, P. D.; Coote, P. W.; Padmanabha Das, K. M.; Malets, Y. S.; Radchenko, D. S.; Moroz, Y. S.; Scott, D. A.; Fackeldey, K.; Hoffmann, M.;

Iavniuk, I.; Wagner, G.; Arthanari, H., An Open-Source Drug Discovery Platform Enables Ultra-Large Virtual Screens. *Nature* **2020**, 580, 663-668.

7. Sadybekov, A. A.; Brouillette, R. L.; Marin, E.; Sadybekov, A. V.; Luginina, A.; Gusach, A.; Mishin, A.; Besserer-Offroy, E.; Longpre, J. M.; Borshchevskiy, V.; Cherezov, V.; Sarret, P.; Katritch, V., Structure-Based Virtual Screening of Ultra-Large Library Yields Potent Antagonists for a Lipid Gpcr. *Biomolecules* **2020**, 10.

8. Alon, A.; Lyu, J.; Braz, J. M.; Tummino, T. A.; Craik, V.; O'Meara, M. J.; Webb, C. M.; Radchenko, D. S.; Moroz, Y. S.; Huang, X. P.; Liu, Y.; Roth, B. L.; Irwin, J. J.; Basbaum, A. I.; Shoichet, B. K.; Kruse, A. C., Structures of the Sigma2 Receptor Enable Docking for Bioactive Ligand Discovery. *Nature* **2021**, 600, 759-764.

9. Kaplan, A. L.; Confair, D. N.; Kim, K.; Barros-Alvarez, X.; Rodriguiz, R. M.; Yang, Y.; Kweon, O. S.; Che, T.; McCorvy, J. D.; Kamber, D. N.; Phelan, J. P.; Martins, L. C.; Pogorelov, V. M.; DiBerto, J. F.; Slocum, S. T.; Huang, X. P.; Kumar, J. M.; Robertson, M. J.; Panova, O.; Seven, A. B.; Wetsel, A. Q.; Wetsel, W. C.; Irwin, J. J.; Skiniotis, G.; Shoichet, B. K.; Roth, B. L.; Ellman, J. A., Bespoke Library Docking for 5-Ht(2a) Receptor Agonists with Antidepressant Activity. *Nature* **2022**, 610, 582-591.

10. Fink, E. A.; Xu, J.; Hubner, H.; Braz, J. M.; Seemann, P.; Avet, C.; Craik, V.; Weikert, D.; Schmidt, M. F.; Webb, C. M.; Tolmachova, N. A.; Moroz, Y. S.; Huang, X. P.; Kalyanaraman, C.; Gahbauer, S.; Chen, G.; Liu, Z.; Jacobson, M. P.; Irwin, J. J.; Bouvier, M.; Du, Y.; Shoichet, B. K.; Basbaum, A. I.; Gmeiner, P., Structure-Based Discovery of Nonopioid Analgesics Acting through the Alpha(2a)-Adrenergic Receptor. *Science* **2022**, 377, eabn7065.

11. Gahbauer, S.; Correy, G. J.; Schuller, M.; Ferla, M. P.; Doruk, Y. U.; Rachman, M.; Wu, T.; Diolaiti, M.; Wang, S.; Neitz, R. J.; Fearon, D.; Radchenko, D. S.; Moroz, Y. S.; Irwin, J. J.; Renslo, A. R.; Taylor, J. C.; Gestwicki, J. E.; von Delft, F.; Ashworth, A.; Ahel, I.; Shoichet, B. K.; Fraser, J. S., Iterative Computational Design and Crystallographic Screening Identifies Potent Inhibitors Targeting the Nsp3 Macrodomain of Sars-Cov-2. *Proc Natl Acad Sci U S A* **2023**, 120, e2212931120.

12. Singh, I.; Fengling, L.; Fink, E.; Chau, A. L.; ARodriguez-Hernandez, A.; Glenn, I.; Zapatero-Belinchon, F. J.; Rodriguez, M.; Devkota, K.; Deng, Z.; White, K.; Wan, X.; Tolmachova, N. A.; Moroz, Y. S.; Kaniskan, U.; Ott, M.; Gastia-Sastre, A.; Jin, J.; Fujimori, D. G.; Irwin, J. J.; Vedadi, M.; Shoichet, B. K., Structure-Based Discovery of Inhibitors of the Sars-Cov-2 Nsp14 N7-Methyltransferase. *Science* **2023**, in press.

13. Sadybekov, A. A.; Sadybekov, A. V.; Liu, Y.; Iliopoulos-Tsoutsouvas, C.; Huang, X. P.; Pickett, J.; Houser, B.; Patel, N.; Tran, N. K.; Tong, F.; Zvonok, N.; Jain, M. K.; Savych, O.; Radchenko, D. S.; Nikas, S. P.; Petasis, N. A.; Moroz, Y. S.; Roth, B. L.; Makriyannis, A.; Katritch, V., Synthon-Based Ligand Discovery in Virtual Libraries of over 11 Billion Compounds. *Nature* **2022**, 601, 452-459.

14. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Zidek, A.; Potapenko, A.; Bridgland, A.; Meyer, C.; Kohl, S. A. A.; Ballard, A. J.; Cowie, A.; Romera-Paredes, B.; Nikolov, S.; Jain, R.; Adler, J.; Back, T.; Petersen, S.; Reiman, D.; Clancy, E.; Zielinski, M.; Steinegger, M.; Pacholska, M.; Berghammer, T.; Bodenstein, S.; Silver, D.; Vinyals, O.; Senior, A. W.; Kavukcuoglu, K.; Kohli, P.; Hassabis, D., Highly Accurate Protein Structure Prediction with Alphafold. *Nature* **2021**, 596, 583-589.

15. Webb, B.; Sali, A., Comparative Protein Structure Modeling Using Modeller. *Curr Protoc Bioinformatics* **2016**, 54, 5 6 1-5 6 37.

16. Baek, M.; DiMaio, F.; Anishchenko, I.; Dauparas, J.; Ovchinnikov, S.; Lee, G. R.; Wang, J.; Cong, Q.; Kinch, L. N.; Schaeffer, R. D.; Millan, C.; Park, H.; Adams, C.; Glassman, C. R.; DeGiovanni, A.; Pereira, J. H.; Rodrigues, A. V.; van Dijk, A. A.; Ebrecht, A. C.; Opperman, D. J.; Sagmeister, T.; Buhlheller, C.; Pavkov-Keller, T.; Rathinaswamy, M. K.; Dalwadi, U.; Yip, C. K.; Burke, J. E.; Garcia, K. C.; Grishin, N. V.; Adams, P. D.; Read, R. J.; Baker, D., Accurate Prediction of Protein Structures and Interactions Using a Three-Track Neural Network. *Science* **2021**, 373, 871-876.

17. Altschul, S. F.; Gish, W.; Miller, W.; Myers, E. W.; Lipman, D. J., Basic Local Alignment Search Tool. *J. Mol. Biol.* **1990**, 215, 403-10.

18. Irwin, J. J.; Shoichet, B. K.; Mysinger, M. M.; Huang, N.; Colizzi, F.; Wassam, P.; Cao, Y., Automated Docking Screens: A Feasibility Study. *J. Med. Chem.* **2009**, 52, 5712-20.

19. Gorgulla, C.; Padmanabha Das, K. M.; Leigh, K. E.; Cespugli, M.; Fischer, P. D.; Wang, Z. F.; Tesseyre, G.; Pandita, S.; Shnapir, A.; Calderaio, A.; Gechev, M.; Rose, A.; Lewis, N.; Hutcheson, C.; Yaffe, E.; Luxenburg, R.; Herce, H. D.; Durmaz, V.; Halazonetis, T. D.; Fackeldey, K.; Patten, J. J.; Chuprina, A.; Dziuba, I.; Plekhova, A.; Moroz, Y.; Radchenko, D.; Tarkhanova, O.; Yavnyuk, I.; Gruber, C.; Yust, R.; Payne, D.; Naar, A. M.; Namchuk, M. N.; Davey, R. A.; Wagner, G.; Kinney, J.; Arthanari, H., A Multi-Pronged Approach Targeting Sars-Cov-2 Proteins Using Ultra-Large Virtual Screening. *iScience* **2021**, 24, 102021.

20. Arul Murugan, N.; Ruba Priya, G.; Narahari Sastry, G.; Markidis, S., Artificial Intelligence in Virtual Screening: Models Versus Experiments. *Drug Discovery Today.* **2022**, 27, 1913-1923.

21. Murail, S.; de Vries, S. J.; Rey, J.; Moroy, G.; Tuffery, P., Seamdock: An Interactive and Collaborative Online Docking Resource to Assist Small Compound Molecular Docking. *Front Mol Biosci* **2021**, 8, 716466.

22. Kochnev, Y.; Hellemann, E.; Cassidy, K. C.; Durrant, J. D., Webina: An Open-Source Library and Web App That Runs Autodock Vina Entirely in the Web Browser. *Bioinformatics* **2020**, 36, 4513-4515.

23. Grosdidier, A.; Zoete, V.; Michielin, O., Swissdock, a Protein-Small Molecule Docking Web Service Based on Eadock Dss. *Nucleic Acids Res* **2011**, 39, W270-7.

24. Guedes, I. A.; Costa, L. S. C.; Dos Santos, K. B.; Karl, A. L. M.; Rocha, G. K.; Teixeira, I. M.; Galheigo, M. M.; Medeiros, V.; Krempser, E.; Custodio, F. L.; Barbosa, H. J. C.; Nicolas, M. F.; Dardenne, L. E., Drug Design and Repurposing with Dockthor-Vs Web Server Focusing on Sars-Cov-2 Therapeutic Targets and Their Non-Synonym Variants. *Sci Rep* **2021**, 11, 5543.

25. Wang, J.; Dokholyan, N. V., Medusadock 2.0: Efficient and Accurate Protein-Ligand Docking with Constraints. *J. Chem. Inf. Model* **2019**, 59, 2509-2515.

26. Bender, B. J.; Gahbauer, S.; Luttens, A.; Lyu, J.; Webb, C. M.; Stein, R. M.; Fink, E. A.; Balius, T. E.; Carlsson, J.; Irwin, J. J.; Shoichet, B. K., A Practical Guide to Large-Scale Docking. *Nat Protoc* **2021**, 16, 4799-4832.

27. McNutt, A. T.; Francoeur, P.; Aggarwal, R.; Masuda, T.; Meli, R.; Ragoza, M.; Sunseri, J.; Koes, D. R., Gnina 1.0: Molecular Docking with Deep Learning. *J Cheminform* **2021**, 13, 43.

28. Li, H.; Sze, K.-H.; Lu, G.; Ballester, P. J. Machine-Learning Scoring Functions for Structure-Based Drug Lead Optimization. In *Wires Computational Molecular Science*; 2020; Vol. 10, Chapter 5, p e1465.

29. Poli, G.; Martinelli, A.; Tuccinardi, T., Reliability Analysis and Optimization of the Consensus Docking Approach for the Development of Virtual Screening Studies. *J. Enzyme Inhib. Med. Chem.* **2016**, 31, 167-173.

30. Ng, M. C.; Fong, S.; Siu, S. W., Psovina: The Hybrid Particle Swarm Optimization Algorithm for Protein-Ligand Docking. *J Bioinform Comput Biol* **2015**, 13, 1541007.

31. Alekseenko, A.; Kotelnikov, S.; Ignatov, M.; Egbert, M.; Kholodov, Y.; Vajda, S.; Kozakov, D., Cluspro Ligtbm: Automated Template-Based Small Molecule Docking. *J. Mol. Biol.* **2020**, 432, 3404-3410.

32. Labbe, C. M.; Rey, J.; Lagorce, D.; Vavrusa, M.; Becot, J.; Sperandio, O.; Villoutreix, B. O.; Tuffery, P.; Miteva, M. A., Mtiopenscreen: A Web Server for Structure-Based Virtual Screening. *Nucleic Acids Res* **2015**, 43, W448-54.

33. Li, H.; Leung, K. S.; Ballester, P. J.; Wong, M. H., Istar: A Web Platform for Large-Scale Protein-Ligand Docking. *PLoS One* **2014**, 9, e85678.

34. Pires, D. E. V.; Veloso, W. N. P.; Myung, Y.; Rodrigues, C. H. M.; Silk, M.; Rezende, P. M.; Silva, F.; Xavier, J. S.; Velloso, J. P. L.; da Silveira, C. H.; Ascher, D. B., Easyvs: A User-Friendly Web-Based Tool for Molecule Library Selection and Structure-Based Virtual Screening. *Bioinformatics* **2020**, 36, 4200-4202.

35. Tsai, T. Y.; Chang, K. W.; Chen, C. Y., Iscreen: World's First Cloud-Computing Web Server for Virtual Screening and De Novo Drug Design Based on Tcm Database@Taiwan. *J. Comput.-Aided Mol. Des.* **2011**, 25, 525-31.

36. Gorgulla, C.; Cinaroglu, S. S.; Fischer, P. D.; Fackeldey, K.; Wagner, G.; Arthanari, H., Virtualflow Ants-Ultra-Large Virtual Screenings with Artificial Intelligence Driven Docking Algorithm Based on Ant Colony Optimization. *Int J Mol Sci* **2021**, 22.

37. Zhang, B.; Li, H.; Yu, K.; Jin, Z., Molecular Docking-Based Computational Platform for High-Throughput Virtual Screening. *CCF Trans High Perform Comput* **2022**, 4, 63-74.

38. Pihan, E.; Kotev, M.; Rabal, O.; Beato, C.; Diaz Gonzalez, C., Fine Tuning for Success in Structure-Based Virtual Screening. *J. Comput.-Aided Mol. Des.* **2021**, 35, 1195-1206.

39. Guo, J.; Janet, J. P.; Bauer, M. R.; Nittinger, E.; Giblin, K. A.; Papadopoulos, K.; Voronov, A.; Patronov, A.; Engkvist, O.; Margreitter, C., Dockstream: A Docking Wrapper to Enhance De Novo Molecular Design. *J Cheminform* **2021**, 13, 89.

40. Gadioli, D.; Vitali, E.; Ficarelli, F.; Latini, C.; Manelfi, C.; Talarico, C.; Silvano, C.; Beccari, A. R.; Palermo, G. An Extreme-Scale Virtual Screening Platform for Drug Discovery. In CF '22: Proceedings of the 19th ACM International Conference on Computing Frontiers, 2022; 2022.

41. Park, H.; Lee, J.; Lee, S., Critical Assessment of the Automated Autodock as a New Docking Tool for Virtual Screening. *Proteins* **2006**, 65, 549-54.

42. Zhang, S.; Kumar, K.; Jiang, X.; Wallqvist, A.; Reifman, J., Dovis: An Implementation for High-Throughput Virtual Screening Using Autodock. *BMC Bioinformatics* **2008**, 9, 126.

43. Trott, O.; Olson, A. J., Autodock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading. *J. Comput. Chem.* **2010**, 31, 455-61.

44. Gentile, F.; Yaacoub, J. C.; Gleave, J.; Fernandez, M.; Ton, A. T.; Ban, F.; Stern, A.; Cherkasov, A., Artificial Intelligence-Enabled Virtual Screening of Ultra-Large Chemical Libraries with Deep Docking. *Nat Protoc* **2022**, 17, 672-697.

45. Yaacoub, J. C.; Gleave, J.; Gentile, F.; Stern, A.; Cherkasov, A., Dd-Gui: A Graphical User Interface for Deep Learning-Accelerated Virtual Screening of Large Chemical Libraries (Deep Docking). *Bioinformatics* **2022**, 38, 1146-1148.

46. Lu, H.; Wei, Z.; Wang, C.; Guo, J.; Zhou, Y.; Wang, Z.; Liu, H., Redesigning Vina@Qnlm for Ultra-Large-Scale Molecular Docking and Screening on a Sunway Supercomputer. *Front Chem* **2021**, 9, 750325.

47. Coleman, R. G.; Carchia, M.; Sterling, T.; Irwin, J. J.; Shoichet, B. K., Ligand Pose and Orientational Sampling in Molecular Docking. *PLoS One* **2013**, 8, e75992.

48. Stein, R. M.; Yang, Y.; Balius, T. E.; O'Meara, M. J.; Lyu, J.; Young, J.; Tang, K.; Shoichet, B. K.; Irwin, J. J., Property-Unmatched Decoys in Docking Benchmarks. *J. Chem. Inf. Model* **2021**, 61, 699-714.

49. Tingle, B. I.; Irwin, J. J., Docking in the Cloud. *ChemXive* **2023**.

50. Mysinger, M. M.; Shoichet, B. K., Rapid Context-Dependent Ligand Desolvation in Molecular Docking. *J. Chem. Inf. Model* **2010**, 50, 1561-73.

51. Meng, E. C.; Gschwend, D. C.; Blaney, J. M.; Kuntz, I. D., Orientational Sampling and Rigid-Body Minimization in Molecular Docking. *Proteins* **1993**, 17, 266-278.

52. Lorber, D. M.; Shoichet, B. K., Fast, Hierarchical Ligand Docking. *man. in prep* **2001**.

53. Lorber, D. M.; Shoichet, B. K., Hierarchical Docking of Databases of Multiple Ligand Conformations. *Curr. Top. Med. Chem.* **2005**, 5, 739-49.

54. Knight, I. S.; Naprienko, S.; Irwin, J. J., Enrichment Score: A Better Quantitative Metric for Evaluating the Enrichment Capacity of Molecular Docking Models. *arXiv* **2021**, 2210.10905v3.

55. Reddy, D. R. *Speech Understanding Systems: A Summary of Results of the Five-Year Research Effort*; Carnegie Mellon University, Department of Computer Science: Pittsburg, 1977.