DOCK Blaster 2.0 - Automated Optimization of Docking Models using Retrospective Docking

Ian S. Knight, Khanh G. Tang, Olivier Mailhot, 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 a widely used technique for leveraging protein structure in ligand discovery, but as a method, it remains difficult to utilize due to limitations that have not been adequately addressed. Despite some progress towards automation, docking still requires expert guidance, hindering its adoption by a broader range of investigators. To make docking more accessible, we have developed a new command-line utility called `dockopt`, which automates the creation, evaluation, and optimization of docking models prior to their deployment in large-scale prospective screens. `dockopt` outperforms our previous automated pipeline across all 43 targets in the DUDE-Z benchmark, and the generated models for 86% of targets demonstrate sufficient enrichment to warrant their use in prospective screens, with normalized LogAUC values of at least 15%. `dockopt` is available as part of the Python package `pydock3` included in the UCSF DOCK 3.8 distribution, which is available for free to academic researchers at https://dock.compbio.ucsf.edu, and free for everyone upon registration at https://tldr.docking.org.

Introduction

Molecular docking is widely used for ligand discovery, both in industry and academia[1-3]. The primary goal of the technique is to predict the binding affinity and pose of small molecules in the binding site of a target protein. The method can screen libraries of billions of molecules and, unlike ligand-based methods, often discovers novel ligands entirely unrelated to those previously known[2, 4-13]. In some cases, docking can lead to the discovery of compounds in the sub-nM range[4, 5, 8, 10], with some of these being active *in vivo*[5, 8-10]. However, unlike other techniques in computational biology, such as homology modeling[14-16] and sequence database searching[17], docking as a procedure remains labor-intensive and intimidating to new users, thereby limiting its wider adoption and hindering its application on a proteomic scale. Docking software is typically complicated and comes with a steep learning curve, making it difficult to utilize to its full potential. This is especially true during the model optimization stage of the docking process, which involves fine-tuning numerous parameters of the model to improve its accuracy and reliability. It does not help that, even when performed by experts, docking can still sometimes fail to accurately reproduce experimentally determined binding characteristics for some targets. These liabilities have diminished the technique's overall impact, not only by obscuring accessibility to researchers with limited computational backgrounds, but also by making it complicated for even experienced computational researchers to deploy docking models at a large scale on the order of billions of molecules.

Automating the several stages of the docking process all in a single pipeline

could significantly reduce the need for expert involvement, which would increase the accessibility of docking as a technique at large. An effective pipeline ideally would simplify the preparation of the docking model for those with less experience while still allowing experts the option to adjust the model as needed. Moreover, beyond merely create a docking model, an optimal pipeline would also optimize the model's parameters to ensure that its performance is at least comparable to that of a model produced by an expert given the same initial data. For this to be possible, the pipeline must first be capable of evaluating the quality of a given model. Typically, this evaluation is performed using retrospective docking[18]. This method involves assessing the model's ability to accurately reproduce the pose and binding characteristics of known ligands and consistently assign them more favorable docking scores compared to designated decoy molecules. These decoy molecules may be property-matched to the known ligands or selected by other methods[48].

Several attempts have been made to automate some parts of the docking process over the past 14 years[19-21], a few of which have web interfaces[22-26]. However, most of these pipelines merely automate the procedural steps for creating a docking model, omitting the practices of evaluation and optimization that experts typically employ when preparing models for large-scale screens[18, 27-30]. As both evaluation and optimization are essential for developing models that can reliably distinguish between binding and non-binding compounds[18], integrating them into these pipelines represents a crucial milestone toward automating the specialized skills of docking experts.

Our work on automating the docking process began in 2009 with the introduction of the web-based tool *DOCK Blaster*[19]. Although it successfully performed retrospective docking on thousands of targets, DOCK Blaster had noteworthy limitations. Notably, it lacked a framework for evaluating results, leaving it difficult to trust the predicted binding modes of resultant models without further assessment. Consequently, it was also unable to optimize the parameters of the DOCK scoring function, which estimates the binding affinity between a candidate molecule pose and the target protein. In effect, DOCK Blaster served merely as a prototype, composed of isolated scripts that made it fragile and difficult to maintain or develop further. In short, although DOCK Blaster demonstrated potential, its shortcomings highlighted the need for a more robust automated pipeline.

Since the appearance of DOCK Blaster, several other web-based docking pipelines have surfaced[31 24, 32-34], some designed with the scalability of the cloud in mind[35-37]. There have also been many reports of increasingly automated docking software without web interfaces[6, 37-46].

Given the mentioned limitations of existing methods, we focused our efforts on improving our own techniques to streamline the docking process. To that end, we re-wrote the command-line tool for creating docking models, `blastermaster`, making it more modular and feature-rich[47], and standardized and published our lab's docking protocol[18]. Despite these advancements, expert supervision remained necessary for conducting model evaluation and optimization, and the absence of a web interface

curtailed the potential for wider accessibility to these improvements.

To address the mentioned challenges, we introduce `dockopt`, a new automated docking pipeline that allows the creation, evaluation, and optimization of docking models using a single tool. `dockopt` is part of the Python package `pydock3`, a toolkit dedicated to the standardization and enhancement of docking methodologies, specifically designed to complement *UCSF DOCK 3.8* and subsequent versions. To evaluate the utility of `dockopt`, we benchmarked it against the DUDE-Z dataset[48].

Methods

The Python package `pydock3` is part of the DOCK 3.8 software distribution and is compatible with `python>=3.8.1,<3.11`. DOCK 3.8 is compatible with modern Linux operating systems. The scripts `dockopt` and `blastermaster` are included with `pydock3`, and all dependencies are defined in the `pyproject.toml` file.

Transitioning from `blastermaster` to `dockopt`

`blastermaster` [47] is a command line tool that generates docking models for protein target binding sites. `dockopt` builds upon `blastermaster`, by creating multiple models in a single pass and then optimizing parameters based on the retrospective docking performance observed for these models. `dockopt` evaluates models using a specified criterion, such as normalized LogAUC (AKA "enrichment score" [49]. To efficiently evaluate candidate models in parallel, `dockopt` employs a designated job scheduler (e.g., Slurm). In summary, `dockopt` enhances the functionality of `blastermaster` by integrating model evaluation and concurrent optimization into the process of generating docking models.

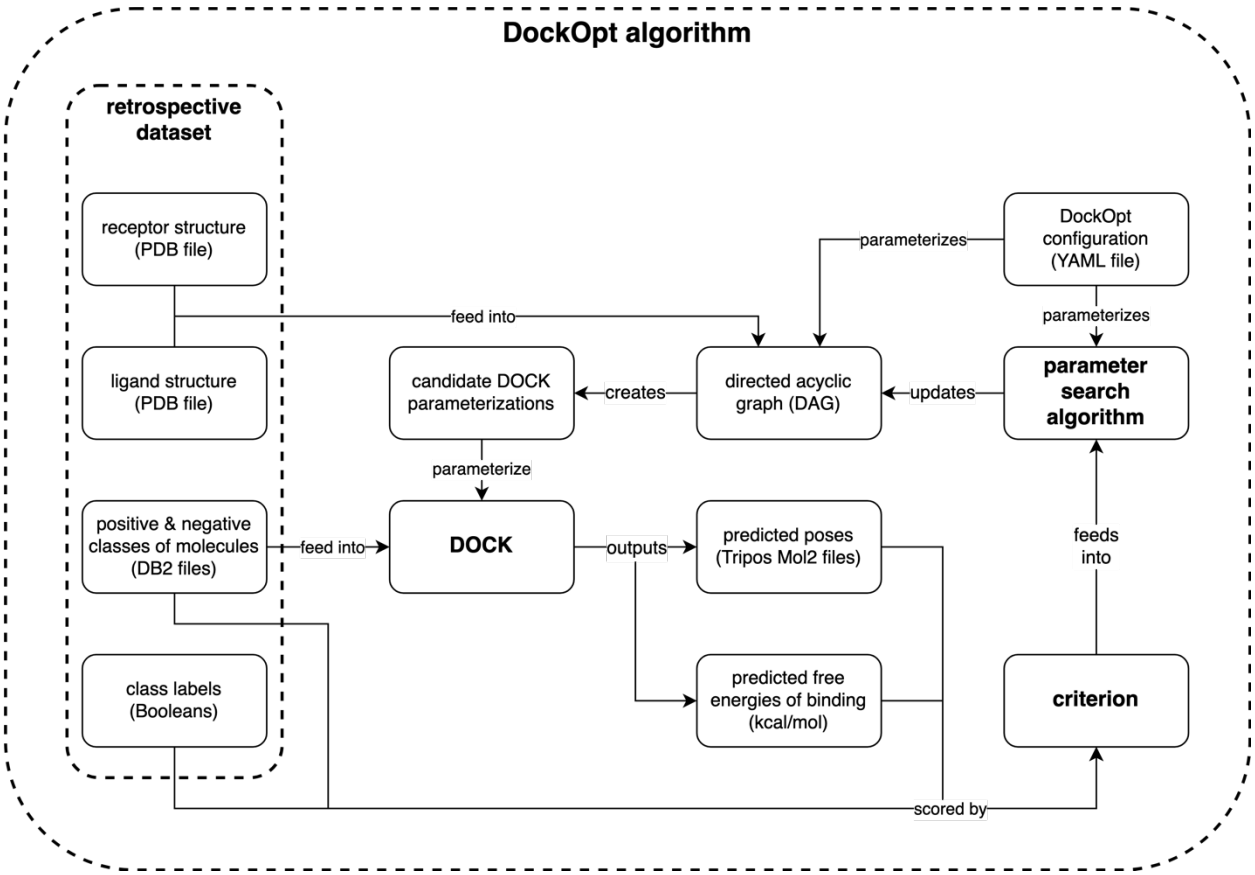


Figure 1. Schematic representation of the `dockopt` algorithm. The retrospective dataset provided by the user consists of (1) a receptor structure, (2) a ligand structure, (3) positive-class molecules (e.g., known ligands), and (4) negative-class molecules (e.g., property-matched decoys). The parameters in the `dockopt_config.yaml` file determine the structure of the directed acyclic graph (DAG), which takes the receptor and ligand structures as input and produces all candidate DOCK parameterizations as output. Each resultant parameterization modifies the DOCK program to form a unique docking model. Each resultant docking model runs retrospective docking on the provided molecules, each labeled as either “positive” or “negative”. The output docking scores and poses are used by the specified criterion to evaluate the parameterizations, which are then ranked accordingly. At this point, the parameters in the

`dockopt_config.yaml` file determine (1) whether the program should iterate, and if so, (2) what proportion of top parameterizations to advance to the next iteration, (3) how to modify these advanced parameterizations, and (4) what new parameterizations to generate. This description holds for all iterations.

Creating docking models with `dockopt`

The `dockopt` pipeline algorithm can be summarized in a diagram (**Figure 1**). The docking program DOCK is parameterized by several files, each controlling different aspects of the program's behavior, such as the sampling algorithm for molecular poses or the scoring function for estimating the free energy of binding for each molecular pose. These files are known as "dockfiles" and exist in custom formats exclusive to DOCK and related software (e.g., `matching_spheres.sph`). In this work, we use the term *DOCK parameterization* to refer to a specific set of dockfiles, and a DOCK parameterization combined with a DOCK executable constitutes a *docking model*.

A DOCK parameterization can be generated from the information contained in a few input files: (1) a receptor structure, (2) a crystallographic ligand structure, and (3) the `dockopt_config.yaml` parameters file. During the model creation phase (see the edge labeled “creates” in **Figure 1**) of the `dockopt` pipeline, a directed acyclic graph (DAG) is used as the data structure for managing the transformation of the input files (*DAG root nodes*) into several DOCK parameterizations (*DAG leaf nodes*) through a multiplex process involving numerous intermediate files. An edge in the DAG represents a dependency relation between a certain input-output pair involved in a particular step in the pipeline (i.e., a step *s* takes {*x*, ...} as input and produces {*y*, ...} as output, so output *y* depends on input *x*). For example, `matching_spheres.sph` depends on `rec.crg.pdb` in the matching spheres generation step

[50]. A child node can be created only once all its parent nodes exist. The input files completely determine the DAG (**Supporting Information** S5), which is automatically derived from them by a deterministic process. Starting from the root nodes of the DAG, a multitude of different branching paths are taken, with each path leading to a distinct leaf node. A specific combination of leaf nodes represents a specific DOCK parameterization, and their respective paths from the root nodes in sum completely define the creation of a certain DOCK parameterization from the input files. Once created, the DAG is mapped to a pipeline of steps that generates the DOCK parameterizations corresponding to the valid combinations of DAG leaf nodes.

Parameter search algorithms

Two parameter search algorithms are currently supported by `dockopt`: grid search and beam search[55]. Grid search explores the search space through a predefined grid of potential parameter value combinations. Due to its exhaustive nature, this approach can theoretically find the optimal parameter combination(s), given a sufficiently fine discretization of parameter space. However, when searching through anything but the coarsest resolutions of parameter space, this method can rapidly become computationally expensive to the point of intractability. In contrast, beam search narrows its search space by applying a selection criterion at each step to retain only the top fraction of candidate DOCK parameterizations for the next step. The range of considered values can be progressively refined, facilitating a more focused exploration of promising solutions. As a result, beam search may not explore all possible options, but it is more computationally efficient, as it constrains exploration to regions of parameter space likely to hold promising parameterizations, based on certain assumptions. For example, one assumption is that locally optimal choices (i.e., high-

scoring candidate solutions at each step) will yield globally optimal or near-optimal solutions. This may not always be true, as locally suboptimal choices can sometimes lead to better overall solutions.

In its pre-release version, `dockopt` used grid search as its search algorithm, exhaustively testing all possible combinations of parameters values, with each parameter taking a pool of possible values. For example, distance-to-surface values for electrostatic thin spheres[51] might be the set {1.0, 1.1, …, 1.9} and distance-to-surface values for ligand desolvation thin spheres might be the set {0.1, 0.2, …, 1.0}, resulting in a Cartesian product space of 10 × 10 = 100 combinations. This strategy works well enough for small numbers of parameters with a limited number of values per parameter. However, it is too inefficient to serve as a general search algorithm, as it would subject users to exponentially increasing computational cost when exploring higher dimensional parameter spaces at finer resolutions. Consequently, the first-release version of `dockopt` uses beam search to efficiently search for favorable DOCK parameterizations. Below, we demonstrate that the implementation of beam search consistently finds superior parameterizations to those found by grid search using comparable computational expense.

`dockopt` is controlled by parameters in the `dockopt_config.yaml` file

The `dockopt_config.yaml` file defines the architecture of the `dockopt` pipeline and controls the range of values that are explored for each parameter. The DAG is derived automatically from the settings in the `dockopt_config.yaml` file, and

a single step in a `dockopt` pipeline generates several different docking models which are then evaluated in parallel. A sequence of steps may be defined with optional iteration and/or early stopping, and recursive embedding of step sequences is supported.

`dockopt` allows rigorous, reproducible experimentation

The `dockopt` pipeline comprises predefined step sequences, which may be defined once and then reused or even algorithmically altered to create new ones at runtime. Thus `dockopt` greatly simplifies benchmarking by facilitating the rigorous comparison of DOCK parameterizations, DOCK executables, and even evaluation criteria. Moreover, different evaluation criteria can be applied in different steps within the same `dockopt` pipeline, such as using a measure of enrichment first, followed by a measure of pose reproduction, and so on. Therefore, an entire experiment intended to measure the efficacy of several variables or search strategies can be defined in a single `dockopt` pipeline and reproduced later simply by re-running the saved pipeline configuration.

`dockopt` pipelines are flexible

The range of possible pipeline structures in `dockopt` is far wider than the default configuration may suggest. Although we recommend that new users try the default configuration first, a wide range of search strategies are available to be explored and customized as users gain more experience and familiarity with the software. These

strategies can be tailored to suit the specific requirements of the users' respective research objectives, providing versatility and flexibility in docking optimization.

`dockopt` reports

`dockopt` generates comprehensive reporting, including a CSV file of results for each docking model tested and an HTML format report containing the following: a histogram of the performance across tested models; linear-log ROC plots showing enrichment; bar plots for performance of individual multi-valued parameters; heatmaps comparing performances across two multi-valued parameters; a ridge plot showing the breakdown of energy terms by binary class; a violin plot showing the charge distribution by binary class.

Results

New software for automatic optimization of docking models is now available as part of the UCSF DOCK 3.8 release. This software is available for free to academic researchers (see: dock.compbio.ucsf.edu) and at modest cost otherwise (email: dock_industry@googlegroups.com). **First**, we describe the software and how to use it on the command line. **Second**, we test the software's utility by using it to perform retrospective docking against the 43 targets of the DUDE-Z benchmark[48]. **Third**, we introduce a web service for this software. The resulting docking model can be downloaded and deployed for prospective docking on the user's on-premises computers, a cloud platform (such as AWS[52]), or any other system capable of large-scale docking. We take up each of these themes in turn.

`dockopt` is a single command for generating and evaluating many different docking models when a retrospective dataset of molecules is available. The performance of a docking model may be evaluated by retrospective docking, where the ability of the model to distinguish between reported binders (positive class) and presumed non-binders (negative class) is assessed. `dockopt` wraps the generation, evaluation, and optimization of docking models all in a single tool.

There are dozens of parameters whose values may affect the quality of the docking models produced by `dockopt`, but a few tend to have the most impact. These include the thickness of the layer of low dielectric and ligand desolvation regions in the binding site, which affect the electrostatic and the ligand desolvation scores,

respectively[51]. Other important parameters include the number and position of orientation spheres (AKA “hotspots” or “matching spheres”), which affect how ligand poses are sampled in the binding site[50]. Still other significant parameters include the target number of poses to generate (*match_goal*), how overlap between ligand and protein is treated (*bump_maximum* and *bump_rigid*), and whether conformations are biased for compatible matches during sampling (*chemical_matching*)[53, 54].

`dockopt` performs retrospective docking on multiple docking models in parallel using a job scheduler, such as Slurm or SGE. Although these two schedulers are the only ones currently supported, it should be straightforward to incorporate any queueing system into `dockopt`. After docking, the docking models are evaluated by the specified criterion (e.g., normalized LogAUC) and then ranked by their performance. Depending on the user’s specification of the configuration file controlling the program, the optimization process may repeat until the stopping criterion is met. A report in HTML format of the best parameter set choices is generated, together with figures summarizing all the runs (see **Methods**, and below). A CSV file of the performance of all docking models tested is saved as `results.csv`. The “dockfiles” that constitute the parameterization of the best job(s) in DOCK are saved in a dedicated directory. Dockfiles for all other parameterizations can be found in the directory `working/`, as indexed in `results.csv`.

In the absence of known ligands, `blastermaster` (the successor of `blastermaster.py` from earlier work[47]) will produce a ready-to-use DOCK

parameterization by means of standard, unoptimized parameter choices.

To install this new software on your own computer (Linux only), see **Supporting Information** S2. For the target of interest, you need (1) a PDB file of the receptor, (2) a PDB file of a corresponding ligand, (3) the identity as SMILES of at least a single known ligand (though a higher number is better), and (4) decoy molecules as SMILES for each known ligand (see **Supporting Information S3 and S4**). A ratio of 50 decoys per ligand is typically used (Bender, 2021). The DUDE-Z benchmark contains 43 examples of such files in a ready-to-use format (see dudez2022.docking.org). To use `dockopt` on the command line, prepare `rec.crg.pdb` (receptor structure) and `xtal-lig.pdb` (crystallographic ligand structure) in an empty directory, together with `positive.tgz` and `negatives.tgz`, both being tarballs of molecules in DB2 format. Now run:

```
pydock3 dockopt - new
cd dockopt_job/
pydock3 dockopt - run slurm
```

In the above example, Slurm is used as the job scheduler for submitting individual docking jobs. Here is an example using SGE instead:

```
pydock3 dockopt - run sge
```

How well does `dockopt` work?

To evaluate the performance of `dockopt`, we benchmarked it against all 43 DUDE-Z targets, using only the data available on the DUDE-Z website (dudez2022.docking.org). 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, specify a target (e.g., AMPC) as an argument:

```
bash run_benchmark.sh AMPC
```

To run all benchmarks, run with no arguments:

```
bash run_benchmark.sh
```

For all 43 DUDE-Z targets, the normalized LogAUC (AKA "enrichment score" [49]) of annotated ligands over property-matched decoys produced by the default `dockopt` configuration was found to be better than that produced by the unoptimized DOCK parameterization included for that target in DUDE-Z[48] (**Figure 2**), whose parameters were derived from default parameters of our previous protocol (`blastermaster.py`[47, 48]). Comparing the parameterizations published in DUDE-Z to those produced by both

grid search and beam search, we observed an average increase in normalized LogAUC of 7.3 percentage points and 16.3 percentage points, respectively. Using grid search, we observed a maximum increase of 33.8 percentage points for the target KITH, and using beam search, we observed a maximum increase of 45.9 percentage points, also for the target KITH.

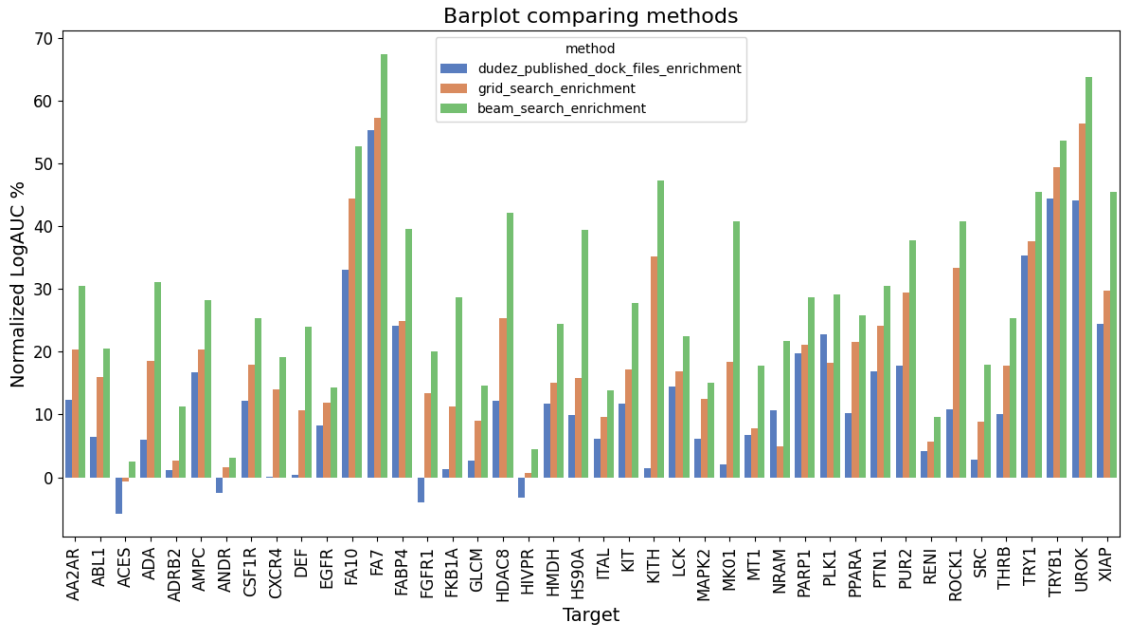


Figure 2. Comparison of beam search and grid search parameter optimizations with the previously published default protocol.

a **b**

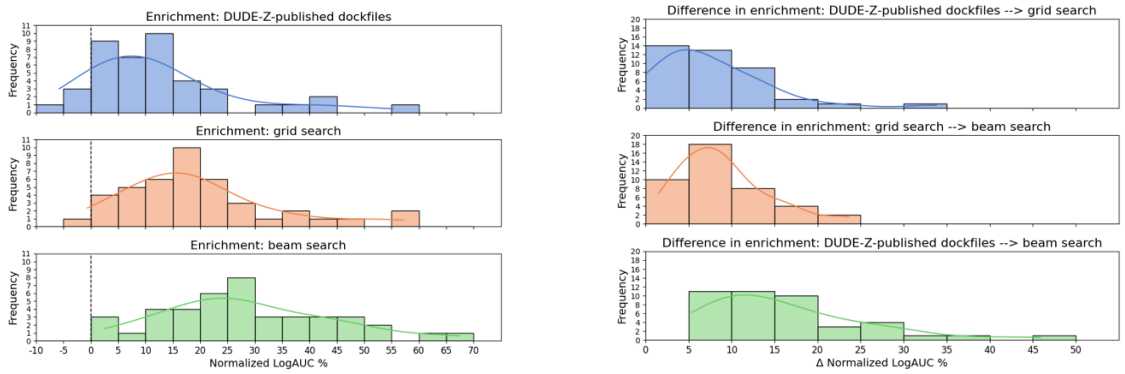


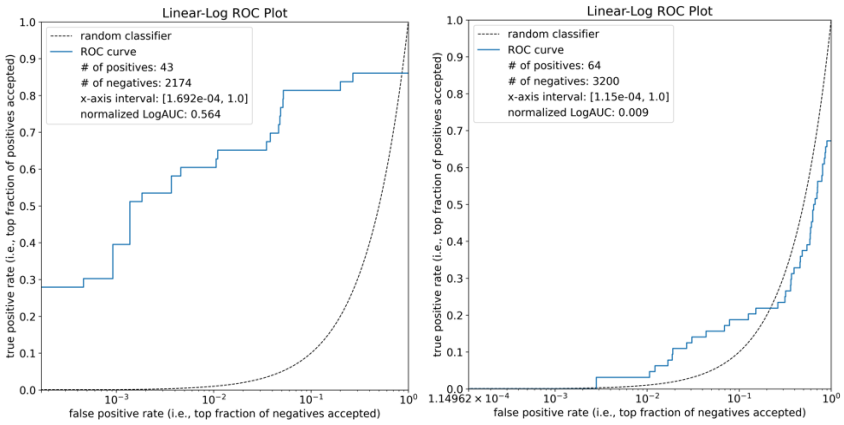
Figure 3. Quality of molecular docking performance as measured by enrichment of known ligands over property-matched decoys for 43 DUDE-Z systems. Recall that normalized LogAUC satisfies: a max value of 1 corresponding to a perfect classifier; a value of 0 for a random classifier; a positive value for a better-than-random classifier; a negative value for a worse-than-random classifier. **(3a):** Comparison of DOCK parameterizations published in the DUDE-Z paper[48] (blue) with parameterizations found by the two search algorithms supported by `dockopt`, grid search (orange) and beam search (green). **(3b):** Improvement in enrichment using `dockopt`. Top: from DUDEZ published to grid search; Middle: from grid search to beam search; Bottom: from DUDEZ published to beam search.

`dockopt` creates a comprehensive report for each target, which can be accessed at dudez2022.docking.org. In this paper, we illustrate the features of these reports using actual results for two DUDE-Z targets as examples (**Figure 4**). Each

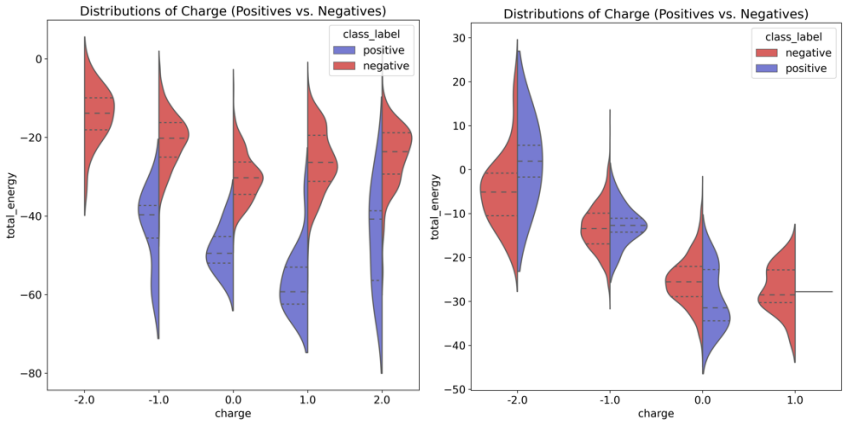
report includes a linear-log ROC plot of the enrichment of the positive class over the negative class. (**Figure 4A**). This visualization captures the performance of a docking model in a single visualization, with the area under the curve (AUC) serving as a quantitative measure of model quality. Next, the report includes additional plots that provide insight into the respective contributions of the terms of the scoring function of the docking program, thereby revealing potential biases that may influence the evaluation of docking models (e.g., imbalanced representations of properties in the dataset, such as charge). The split violin charts (**Figure 4B**) show the scores of the binary classes grouped by net molecular charge. The ridgeline plots (**Figure 4C**) show how the binary classes compare across the energy terms whose sum constitutes the predicted free energy of binding. Boxplots of the evaluation criterion are generated for parameters for which multiple values were tested, providing a visual comparison across different parameter values (**Figure 4D**). Finally, heatmaps (**Figure 4E**) summarizing the distribution of the evaluation criterion as a function of two variables are generated for every pair of parameters for which multiple values were attempted. The value of each 2D coordinate in the heatmap corresponds to the maximum criterion value obtained across all parameterizations that used the combination of parameter values indicated by the coordinate; for example, a heatmap may show the behavior of the normalized LogAUC as a function of the electrostatic spheres (thin layer) and the desolvation spheres (thin boundary). Looking at the example targets from DUDE-Z, we observe that HIVPR shows consistently low enrichment, irrespective of the changes in these two variables. In contrast, for Factor 7A, a strong dependency on these parameters is noted, indicating the possibility of pinpointing an optimal or near-optimal parameter

combination. Collectively, these auto-generated plots provide a deeper understanding of a docking model, highlight any biases, and predict its potential performance in prospective docking screenings. Comprehensive reports featuring these visualizations for all DUDE-Z systems can be accessed at dude2022.docking.org.

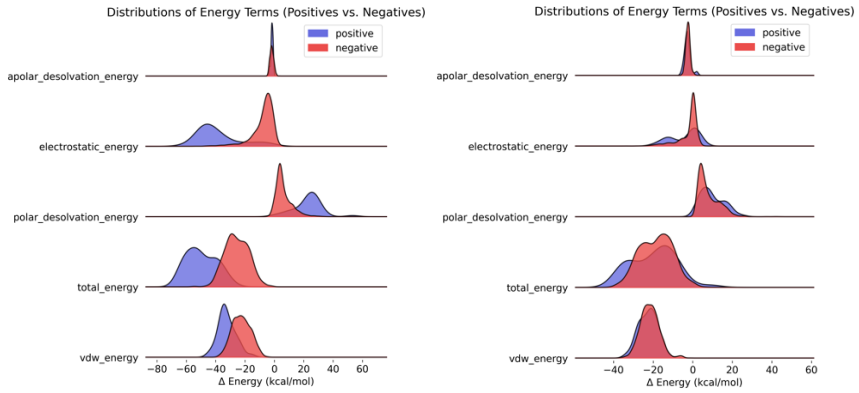
A.



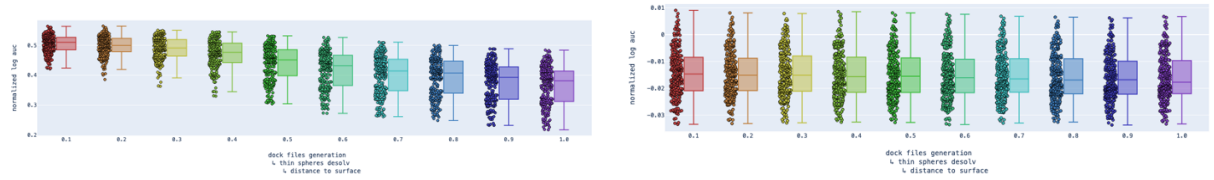
B.



C.



D.



E.

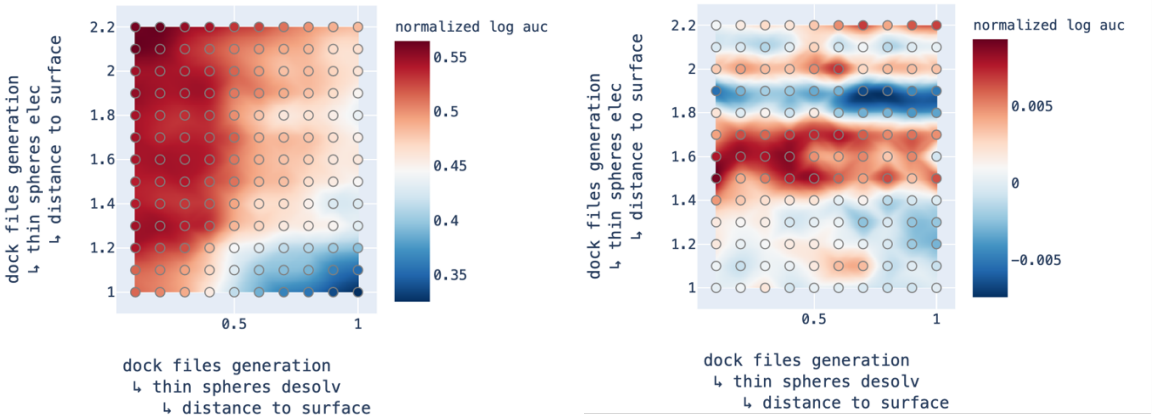


Figure 4. Selected graphical reports of docking models optimized for two targets. Left: FA7. Right: HIVPR. A. Linear-log ROC plot of enrichment of ligands vs decoys. B. Violin plots of the distribution of charges. C. Unidimensional plots of the distribution of

energy terms for docked ligands vs. decoys. D. Unidimensional plot of the distributions of enrichment across different values of desolvation thin spheres radii. E. Bidimensional plots of enrichment as function of electrostatic and desolvation thin spheres radii.

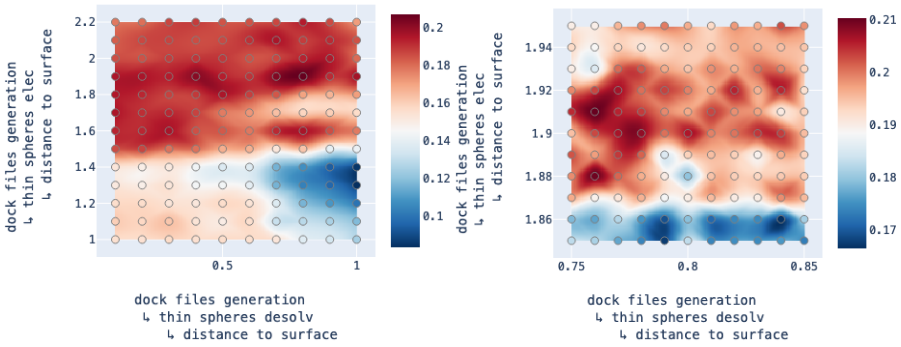


Figure 5. Heatmaps produced by two adjacent steps in a `dockopt` pipeline for target CSF1R, demonstrating the ability of beam search to narrow the range of considered values in a greedy fashion. Left: The heatmap for the former step shows a coarser resolution of exploration with a wider range of parameter values on both axes. The optimum tested coordinate is found to be (0.8, 1.9). Right: The heatmap for the latter step shows a finer resolution of exploration in the neighborhood around the optimum witnessed in the previous step. Note the nontrivial degree of fluctuation in enrichment across tested coordinates, even at the finer resolution.

Docking models that consistently produce incorrect poses for known ligands are defective, regardless of whether they yield high enrichment. Although we typically do not know the exact pose of every known ligand in the retrospective dataset, we generally expect that most predicted poses should overlap the crystallographic ligand pose and mirror its receptor interactions. We illustrate such pose-oriented

considerations in two UCSF Chimera[55] sessions, each displaying a superimposition of the receptor, the crystallographic ligand, and the predicted poses of the known ligands for comparison (**Figure 6**). In some cases, most ligands present a compact superimposition with high overlap, which often occurs when a single "warhead" dominates receptor-ligand interactions, as seen in coagulation factor VIIa (FA7). In other cases, the superimposition might be less defined, but it can still confirm that the predicted poses occupy the same region as the experimentally observed ligand, as seen in HIV-1 protease (HIVPR).

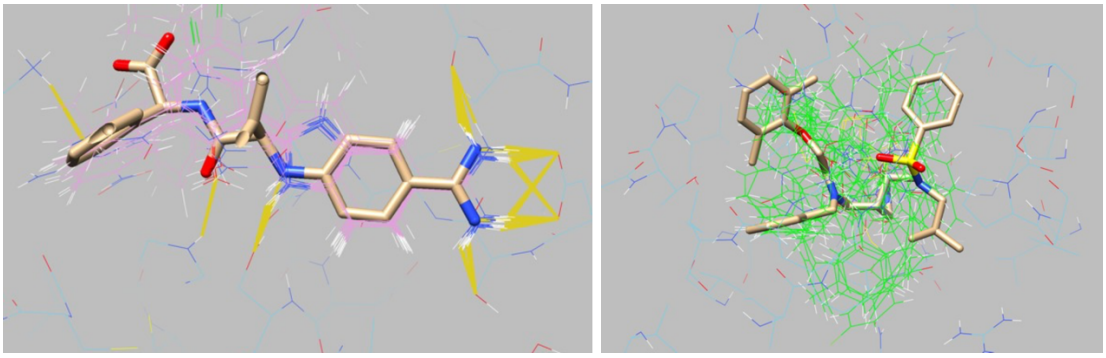


Figure 6. Superimposition of the crystallographic ligand (sticks) and the docked ligands (wire). Hydrogen bonds and polar interactions with the protein are shown in mustard. Left. Coagulation factor VIIa (FA7). Right: HIV-1 protease (HIVPR).

Encouraged by the ability of `dockopt` to produce docking models apparently suitable for prospective docking (**Figure 2**), we built a web-based interface for it at tldr.docking.org under the "dockopt" module (**Figure 6**). Registration is free. Sample data for 43 targets in ready-to-use formats are available at dudez2022.docking.org.

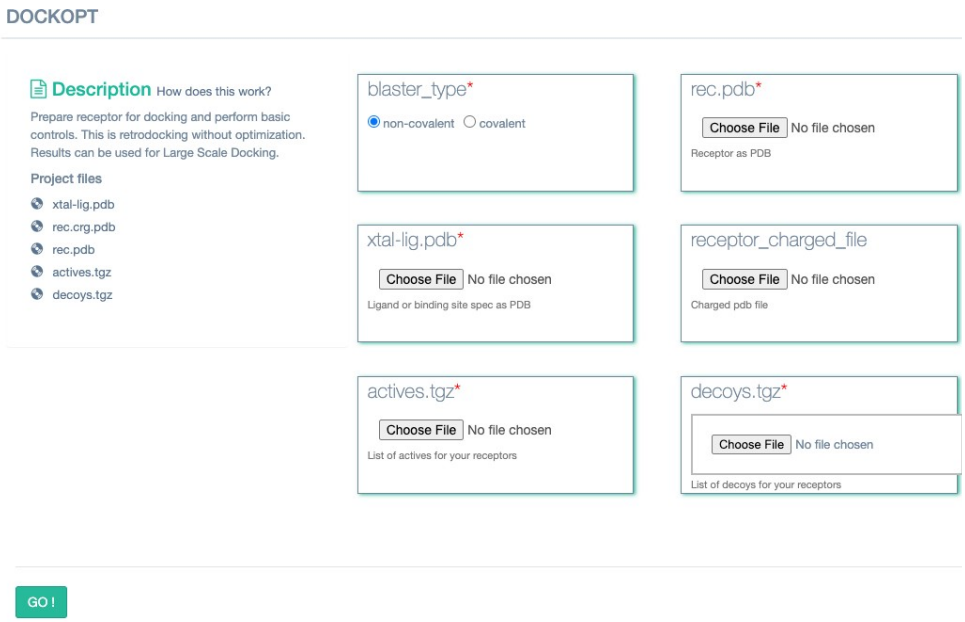


Figure 6. A web interface for `dockopt` via tldr.docking.org.

Currently, only the default parameters are available to submit `dockopt` jobs, which can take up to about 24 hours for larger retrospective datasets of thousands of molecules, and obviously depends on the current server load. The user receives an email upon job completion, at which point they may download the best model(s), the predicted poses in Tripos Mol2 format, and a report in HTML format containing various plots about the performance of the tested docking models across different sets of parameters (**Figure 6**). The downloaded model can be deployed in a large-scale docking screen on any system with the necessary compute resources, such as a departmental cluster or via a cloud platform (e.g., AWS[52]).

`dockopt` can even dock without any experimentally known ligands, but here it runs into the same problems that a human would. Without controls, it is difficult to ascertain how well docking is performing, short of running a prospective screen.

`blastermaster` uses sensible, typical 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:

```
pydock3 blastermaster - new
cd blastermaster_job/
pydock3 blastermaster - run
```

Both `dockopt` and `blastermaster` are now available and ready to use. They may be accessed either by licensing and installing DOCK 3.8 or via tldr.docking.org as previously described.

The normalized LogAUC of a random classifier tends to 0% as the numbers of positive-class molecules and negative-class molecules both approach infinity [49]. However, whereas a typical retrospective docking campaign usually has between 10 and 30 positive-class molecules, curated datasets such as DUDE-Z often have targets with more than 100. To obtain a conditional probability distribution of normalized LogAUC produced by a random classifier given a specific number of positive-class molecules, we performed 1 billion simulations for each number of positives, ranging from 1 to 50, with all datasets maintaining a negative-to-positive ratio of 50:1. The conditional empirical distribution obtained from these simulations allows us to compute the probability (p-value) that, purely by chance, a random classifier would produce a normalized LogAUC greater than or equal to a certain value.

To evaluate the possibility that the increased enrichment observed during optimization with `dockopt` was simply due to the large number of docking models tested, we employ the Bonferroni correction to adjust the p-value threshold for statistical significance. The Bonferroni correction is a widely used method to control the family-wise error rate in multiple hypothesis testing, accounting for the increased likelihood of false positives when conducting a larger number of tests. The correction involves dividing the desired p-value by the number of tests performed; for example, a p-value of $p = 0.01$ (the default in `dockopt`) corrected for 1000 tests would become 10^{-5}. The Bonferroni correction assumes that all tests performed are independent, which is not necessarily the case for parameterizations generated by `dockopt`, as parameterizations nearby in parameter space are expected to yield similar results. However, the significance threshold obtained by the Bonferroni correction is strictly more stringent than that obtained by any method that accounts for dependent tests, so using it only further mitigates the risk of falsely identifying insignificant results as significant. As a corollary, it is also important to recognize that this conservative approach may increase the risk of false negatives, particularly in contexts of high test correlation.

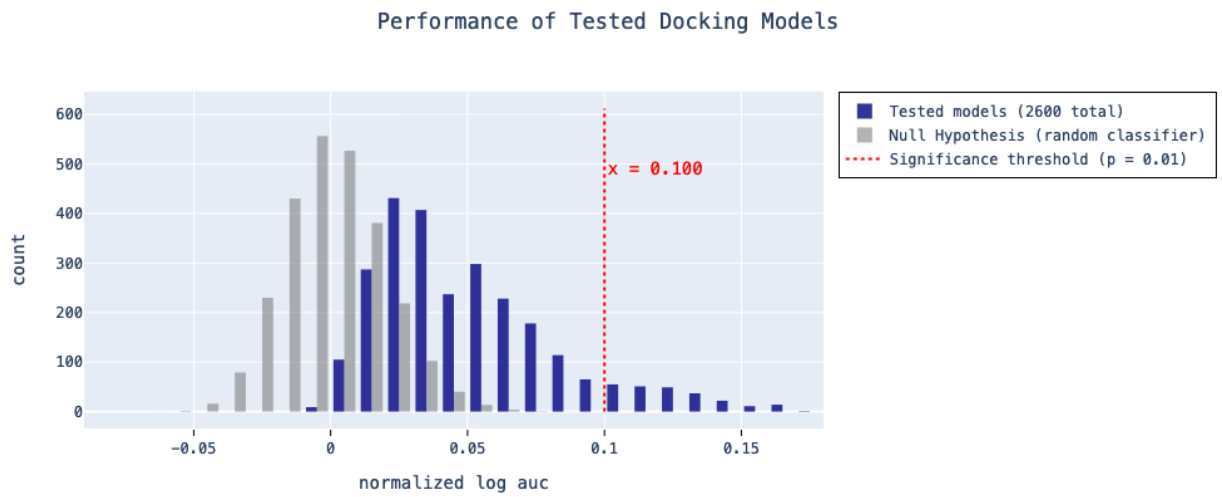


Figure 7. An example histogram of the performance of 2600 tested docking models for the target ADA. Using p = 0.01 with a Bonferroni correction to account for the multiple tests (2600 total), a significance threshold of 0.100 normalized LogAUC was derived from a cumulative distribution function of the conditional empirical distribution of normalized LogAUC produced by a random classifier.

Applying the Bonferroni correction to our analysis allows us to rigorously assess the statistical significance of the observed enrichments, ensuring that the reported improvements in docking performance are not merely an artifact of testing multiple docking models. Our results demonstrate that the superior performance of `dockopt` over the parameterizations published in DUDE-Z remains significant even after accounting for the multiple comparisons, thereby providing robust evidence for the effectiveness of our method.

Discussion

Three themes emerge from this study. **First**, a new automated pipeline for docking model creation, evaluation, and optimization has been developed. **Second**, the automated procedure can optimize the docking parameters significantly better than our previous automated system, and for most targets produces docking models that are suitable for large-scale prospective screens. **Third**, the new pipeline can be installed locally or accessed via a new web interface we have created, and the resultant docking model may be downloaded and deployed for large-scale docking, either on-premises or in the cloud[52]. We take up each of these themes in turn.

`dockopt` implements many of the best practices in our standard lab protocol[26] in an automated fashion. We have augmented this procedure to include optimization techqniques that capture many, but not all, of the current best practices in our lab. For example, the boundaries of the regions of low dielectric and ligand desolvation in the binding site are optimized. Work in our lab suggests that defining these boundaries is often critical to obtaining satisfactory retrospective enrichment during model optimization[51]. The pipeline also optimizes the matching spheres used for sampling ligand orientations, which play a key role in determining whether a docking model is likely to succeed in a prospective screen. Furthermore, the software has been designed with flexibility in mind, allowing for the optimization of dozens of additional parameters, and minimal development effort is required to incorporate and optimize any new parameters or search strategies that a researcher might consider desirable.

`dockopt` demonstrates competent docking performance against the DUDE-Z benchmark regularly used in our research. In 37 out of the 43 DUDE-Z systems (~86%), this fully automated procedure with beam search produces a docking model exhibiting a normalized LogAUC of at least 15%. This threshold is a good heuristic for whether a docking model is suitable for prospective docking. This performance represents a striking improvement compared to the docking models published in DUDE-Z, which resulted in only 13 successes (~29%) by the same standard.

The use of the Bonferroni correction in our analysis provides rigorous evidence for the efficacy of `dockopt`. By adjusting the significance threshold to account for the use of multiple tests, the Bonferroni correction effectively reduces the probability of Type I errors (i.e., false positives). This statistical correction serves as a key control mechanism, providing confidence that the performance improvements we report in this work are not merely statistical contrivances but indeed genuine indicators of the ability of `dockopt` to effectively optimize docking models. Although the Bonferroni correction assumes that the tests are independent, which is likely not the case with `dockopt`, it nevertheless provides a reliable conservative measure of statistical significance.

We set up a public web interface for `dockopt` at https://tldr.docking.org, which showcases this software's ability to build and refine docking models completely automatically, given a retrospective dataset of molecules labelled positive or negative. This platform also evaluates the resultant model's suitability for prospective docking screens, estimating the likelihood of the model to consistently prioritize new ligands.

There are several caveats to this work. `dockopt` requires a retrospective dataset of positive and negative molecules (e.g., known ligands and property-matched decoys) with which to optimize the docking model and evaluate its enrichment capacity, pose reproducibility, or any other supported criterion of interest. Without these control molecules, `dockopt` cannot assess whether a given model is of sufficient quality to use for prospective docking. Furthermore, `dockopt` works on most of the targets we tried, but not all. Therefore, our approach should not be mistaken for a universal solution to automatic optimization of docking models.

Because measures of enrichment capacity do not incorporate any structural information of predicted poses, the use of normalized LogAUC as the single criterion for model evaluation means that there is the possibility of the model overfitting to the ligands provided by the user, which usually number at most a few dozen. However, this potential issue could be ameliorated by using an alternate criterion measuring pose reproduction, or perhaps even a criterion measuring both enrichment and pose reproduction. Until such evaluation criteria are implemented in `dockopt`, users are advised to provide as many diverse known ligands as possible to mitigate the risk of overfitting.

These caveats should not obscure the main results of this work. We have developed a fully automated tool for the creation, evaluation, and optimization of docking models, which is now available as part of the UCSF DOCK 3.8 distribution. In

addition to being offered free of charge to academic researchers via https://dock.compbio.ucsf.edu, the software can also be used for free by non-academic researchers upon registration at https://tldr.docking.org. It is important to note that we cannot guarantee the results of any docking screen using `dockopt`; this software is to be utilized at the user's own risk. For optimal ligand discovery outcomes, we strongly encourage the use of sanity checks and controls at every stage of the docking process, as discussed in this work.

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 authors declare no other conflicts.

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. This software is free to academics (dock.compbio.ucsf.edu for a free license and download) and at modest cost to for-profit users (dock_industry@googlegroups.com).

All data, including the DUDE-Z 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 at github.com/docking-org/ in the repositories: `dude-z-benchmark` (benchmarking data); `pydock3` (`dockopt` & `blastermaster`); `zinc22-3d` (the ZINC-22 3D database preparation pipeline).

Literature Cited.

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. 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.
19. 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.
20. 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.
21. 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.
22. 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.

23. 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.
24. 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.
25. 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.
26. Wang, J.; Dokholyan, N. V., Medusadock 2.0: Efficient and Accurate Protein-Ligand Docking with Constraints. *J. Chem. Inf. Model* **2019**, *59*, 2509-2515.
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. Knight, I. S.; Naprienko, S.; Irwin, J. J., Enrichment Score: A Better Quantitative Metric for Evaluating the Enrichment Capacity of Molecular Docking Models. *arXiv* **2022**, 2210.10905v3.
50. 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.
51. Mysinger, M. M.; Shoichet, B. K., Rapid Context-Dependent Ligand Desolvation in Molecular Docking. *J. Chem. Inf. Model* **2010**, *50*, 1561-73.
52. Tingle, B. I.; Irwin, J. J., Docking in the Cloud. *ChemRxiv* **2023**.
53. Lorber, D. M.; Shoichet, B. K., Fast, Hierarchical Ligand Docking. *man. in prep* **2001**.
54. Lorber, D. M.; Shoichet, B. K., Hierarchical Docking of Databases of Multiple Ligand Conformations. *Curr. Top. Med. Chem.* **2005**, *5*, 739-49.
55. Pettersen, E. F.; Goddard, T. D.; Huang, C. C.; Couch, G. S.; Greenblatt, D. M.; Meng, E. C.; Ferrin, T. E., Ucsf Chimera--a Visualization System for Exploratory Research and Analysis. *J Comput Chem* **2004**, *25*, 1605-12.

TOC Graphic