

SOFTWARE

Open Access



AutoGrow4: an open-source genetic algorithm for de novo drug design and lead optimization

Jacob O. Spiegel and Jacob D. Durrant*

Abstract

We here present AutoGrow4, an open-source program for semi-automated computer-aided drug discovery. AutoGrow4 uses a genetic algorithm to evolve predicted ligands on demand and so is not limited to a virtual library of pre-enumerated compounds. It is a useful tool for generating entirely novel drug-like molecules and for optimizing preexisting ligands. By leveraging recent computational and cheminformatics advancements, AutoGrow4 is faster, more stable, and more modular than previous versions. It implements new docking-program compatibility, chemical filters, multithreading options, and selection methods to support a wide range of user needs. To illustrate both de novo design and lead optimization, we here apply AutoGrow4 to the catalytic domain of poly(ADP-ribose) polymerase 1 (PARP-1), a well characterized DNA-damage-recognition protein. AutoGrow4 produces drug-like compounds with better predicted binding affinities than FDA-approved PARP-1 inhibitors (positive controls). The predicted binding modes of the AutoGrow4 compounds mimic those of the known inhibitors, even when AutoGrow4 is seeded with random small molecules. AutoGrow4 is available under the terms of the Apache License, Version 2.0. A copy can be downloaded free of charge from <http://durrantlab.com/autogrow4>.

Keywords: Autogrow, Genetic algorithm, Computer-aided drug design, Virtual screening, PARP-1

Introduction

Computer-aided drug discovery (CADD), a critical component of many pharmaceutical pipelines, is a powerful tool that complements the expertise of medicinal chemists and biologists. Given that there are 10^{20} – 10^{23} synthesizable drug-like molecules [1–3], no method—experimental or computational—can hope to explore even a small subset of drug space. But CADD enables in silico experiments at scales much larger than are possible ex silico [1–5] and so can prioritize which candidate ligands warrant further testing in enzymatic or biophysical assays. CADD has been successfully applied to hit discovery, lead optimization, and compound synthesis [4–7].

CADD can be broadly divided into two categories: ligand-based drug design (LBDD) and structure-based drug design (SBDD) [6, 8]. To predict ligand binding, LBDD considers the physicochemical properties of known ligands without regard for the atomic structure of the target macromolecular receptor (e.g., protein). In contrast, SBDD predicts binding based on the receptor structure [6, 8]. SBDD can be further divided into screening and de novo approaches. Screening considers a finite database of pre-enumerated compounds, and de novo approaches generate new compounds in silico using algorithms that explore a wider range of chemistry space [4–6, 9].

We here describe AutoGrow4, a free Python-based open-source program for de novo SBDD CADD. AutoGrow4 uses a genetic algorithm (GA) to create new predicted ligands. It draws on an initial population of seed molecules to create a new population (i.e., a generation) of potential solutions (ligands). It then docks these

*Correspondence: durrantj@pitt.edu
Department of Biological Sciences, University of Pittsburgh, 4200 Fifth Ave, Pittsburgh, PA 15260, USA



© The Author(s) 2020. This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

compounds into a user-specified target protein and ranks each by its calculated fitness. New generations are seeded with the top-scoring molecules of the previous generation.

AutoGrow4 expands on the approach used in previous versions of the algorithm. The original AutoGrow, released in 2009, was one of the first de novo CADD programs to use fully flexible docking and was one of only a few free open-source programs for de novo CADD [4, 5]. More recent advances in docking software, cheminformatics libraries, and multithreading approaches [10, 11] have now enabled further improvements. AutoGrow4 has an entirely rewritten codebase that is designed to be faster, more stable, and more modular than previous versions.

To demonstrate utility, we show how AutoGrow4 can be used both to design entirely novel drug-like molecules and to optimize preexisting inhibitors. In both cases, we apply AutoGrow4 to poly(ADP-ribose) polymerase 1 (PARP-1), a well-characterized DNA-damage recognition protein. We chose PARP-1 because (1) the many known PARP-1 inhibitors (PARPi) serve as positive controls and leads to optimize [12–15]; (2) PARPi are effective treatments for many cancers with defects in the *Breast Cancer (BRCA) 1* and *2* genes [16, 17]; and (3) the PARP-1 catalytic domain has a well-characterized druggable pocket [18–20].

AutoGrow4 will be a useful tool for the CADD community. We release it under the terms of the Apache License, Version 2.0. A copy can be downloaded free of charge from <http://durrantlab.com/autogrow4>.

Methods

AutoGrow4 design and implementation

AutoGrow4 starts with an initial (input) population of compounds. This source population, called generation 0, consists of a set of chemically diverse molecular fragments (for de novo design) or known ligands (for lead optimization). AutoGrow4 creates the first generation by applying three operations to the source population: elitism, mutation, and crossover (Fig. 1). Subsequent generations are created similarly from the compounds of the immediately preceding generation.

Population generation via elitism

The elitism operator progresses a sub-population of the fittest compounds from one generation to the next without alterations. The AutoGrow4 elitism operator is similar to that of previous AutoGrow implementations, with two notable improvements. First, users can now optionally choose to regenerate and redock elite compounds. Many docking programs (e.g., AutoDock Vina [21]) are stochastic, so users may get slightly different poses each

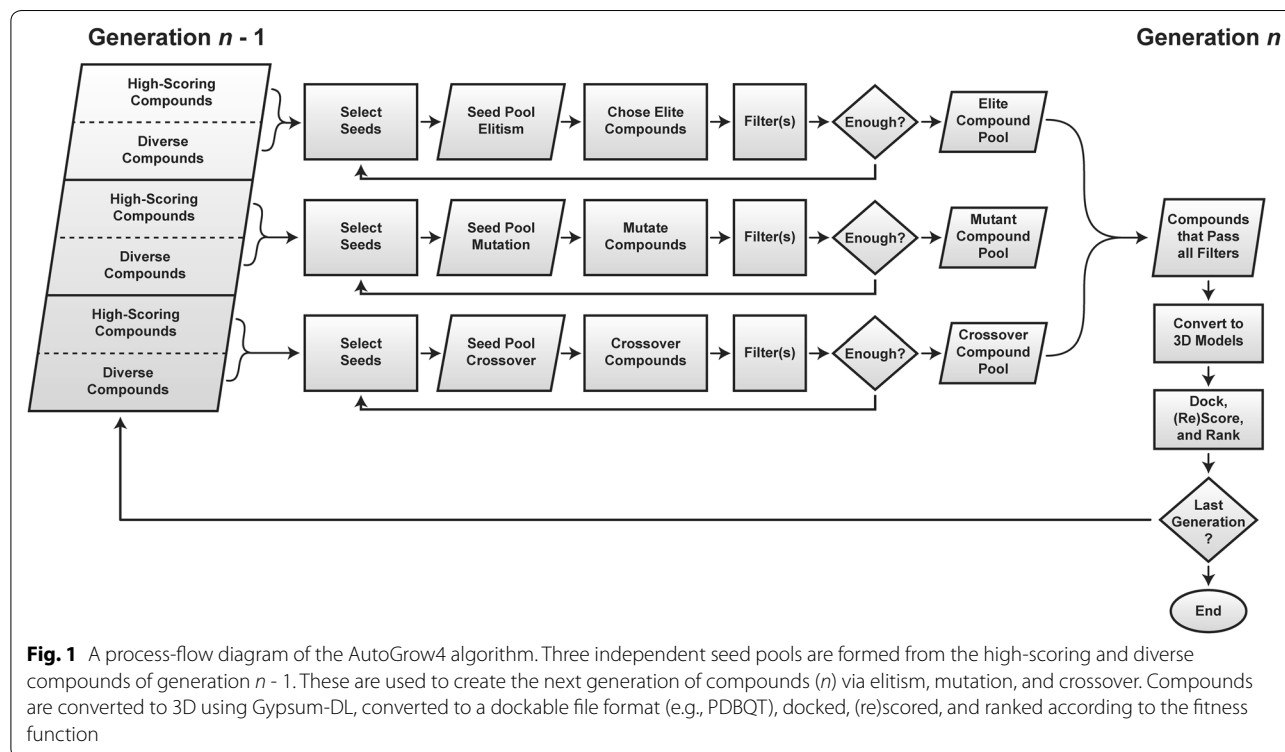


Fig. 2 Compound lineages from a lead-optimization run seeded with PARPi fragments, chosen to illustrate crossover and mutation operations. QVina2 was used for docking. To simplify presentation, we omit some intermediate steps such as changes in protonation states. **a** Olaparib recreated via crossover from two source fragments. The largest commonly shared substructure is highlighted in purple. **b** A high-scoring compound derived from a E7449 (known inhibitor) fragment illustrates the mutation operator

time. Reassessment may identify better poses than previous attempts.

Second, AutoGrow4 decouples the elitism operator from seed-population selection. In previous AutoGrow versions, the compounds that advanced via elitism were the same compounds used to seed the mutation and crossover operators (see below). With these two processes now decoupled, the user can independently control the number of compounds that advance via elitism vs. other operators.

Population generation via mutation

The mutation operator performs an *in silico* chemical reaction to generate an altered child compound derived from a parent (Fig. 2b). AutoGrow4's file-naming scheme allows the user to easily trace any mutant compound's lineage.

The AutoGrow4 mutation operator is improved over previous versions of AutoGrow. AutoGrow 3.1.3 used AutoClickChem [22] to perform *in silico* reactions on 3D compound models, which required an extensive codebase to perform substructure searches, molecular alignments, and *in silico* reactions [5, 22]. In contrast, AutoGrow4 uses SMARTS-reaction notation, together with RDKit, to perform chemical mutations much faster.

This improved approach allowed us to easily add new reaction sets. Aside from the 36 click-chemistry reactions that were already possible with AutoClickChem (the AutoClickChemRxn set), AutoGrow4 also draws on a second library of 58 reactions published by Hartenfeller et al. (the RobustRxn set) [23]. We merged these two sets to form a third with 94 reactions (the AllRxn set). All sets were manually inspected, extensively unit tested, and adjusted where chemical modifications were necessary. The tutorial included with the AutoGrow4 download describes how users can create and incorporate their own custom reaction sets.

Seventy-nine of AutoGrow4's 94 default reactions require two reactants. In these cases, one of the reactants is taken from a previous generation, and the other is taken from one of AutoGrow4's complementary molecular-fragment libraries. To create these libraries,

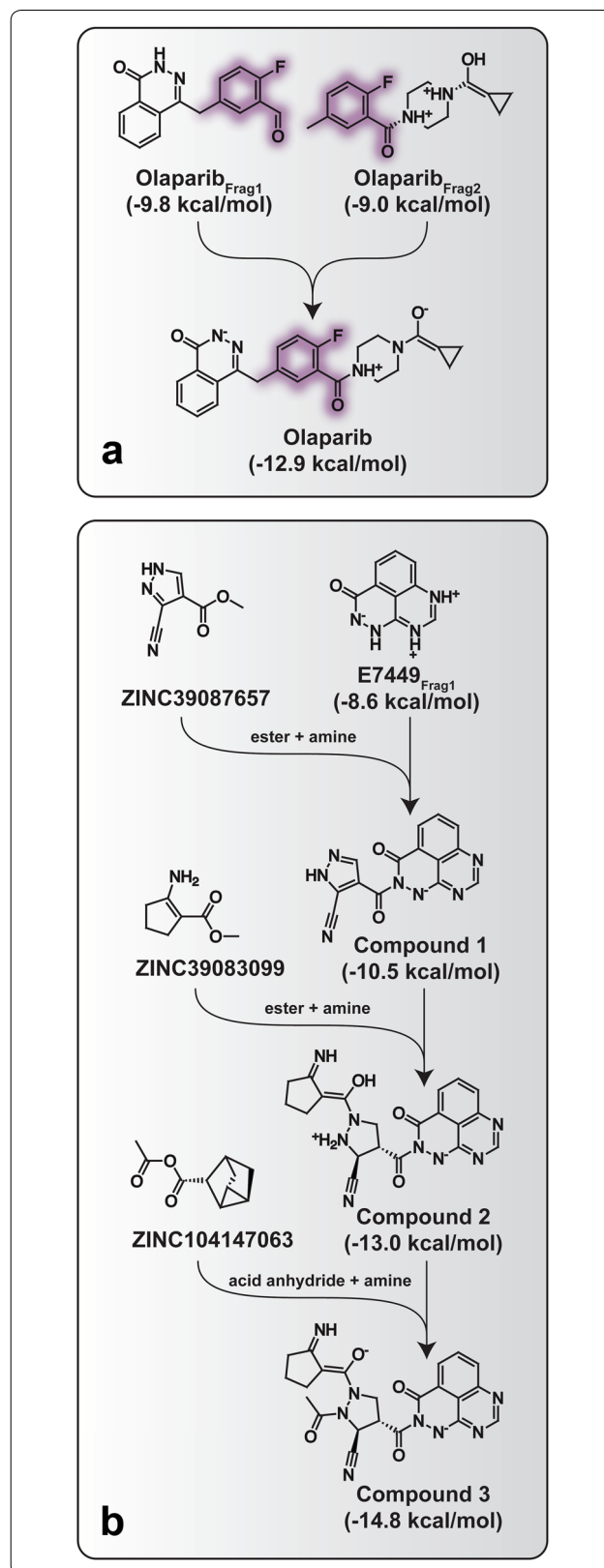


Table 1 The default AutoGrow4 molecular filters

Name	logP	HD; HA	MW	MR	Atoms	RotB	R	N; O; X	PSA	Sub
Lipinski [72]	≤ 5.0	≤ 5; ≤ 10	≤ 500							
Lipinski* [5]	≤ 5.0	≤ 5; ≤ 10	≤ 500							
Ghose [54]	-0.4 to 5.6		160–480	40–130	20–70					
Ghose* [5]	-0.4 to 5.6		160–500	40–130	20–70					
Vandewaterbeemd [107]			< 450						< 90	
Mozziconacci [108]						≤ 15	≤ 6	≥ 1; ≥ 1; ≤ 7		
BRENK [56]										+
NIH [109, 110]										+
PAINS [111]										+

Lipinski allows for one violation. Lipinski* is a stricter version that allows for no violations. Ghose* is a more lenient version of Ghose that allows compounds with molecular weights up to 500 Da

HD hydrogen-bond donor; HA hydrogen-bond acceptor; MW molecular weight (Da); MR molar refractivity ($\text{m}^3 \text{mol}^{-1}$); Atoms atom count; RotB rotatable bonds; R rings; N, O, and X nitrogen, oxygen, and halogen atoms, respectively; PSA polar surface area (\AA^2); Sub substructure searching

we downloaded a set of 19,274,338 commercially available small molecules from the Zinc15 database [24] on December 19, 2019. Each compound had a molecular weight (MW) less than 250 Da and a predicted octanol–water partition coefficient (logP) less than 5.0. We further filtered the compounds using the Lipinski* filter, which allows for no violations (Table 1). To ensure that all compounds could participate in at least one of AutoGrow4's default in silico reactions, we discarded those molecules that did not possess any appropriate functional groups. We then divided the remaining compounds into functional-group categories. For each category, we retained at most the 5000 compounds with the lowest MW. These complementary libraries are included in the AutoGrow4 download in the SMILES (SMI) format. A customization option also allows users to provide their own complementary molecule libraries.

Population generation via crossover

The crossover operator merges two compounds from previous generations into a new compound. Like the previous version of AutoGrow (3.1.3) [5], the AutoGrow4 crossover operator finds the largest substructure that the two parent compounds share and generates a child by randomly combining their decorating moieties (Fig. 2a). AutoGrow4 embeds information about the lineage of each crossover in the compound file name, allowing users to easily examine any compound's evolution.

AutoGrow 3.1.3 used LigMerge [25] to perform crossovers. LigMerge requires computationally expensive geometric calculations to merge 3D molecular models. In contrast, AutoGrow4 uses the RDKit Python library [26] to generate child compounds from SMILES strings of the parents. This change dramatically reduces the

computational cost of compound generation and greatly simplifies the AutoGrow4 codebase.

Molecular filtration

AutoGrow4 uses common molecular filters to remove generated compounds with undesirable physical and chemical properties (e.g., poor predicted solubility, high biological reactivity, etc.). These compounds are eliminated before docking to avoid wasting computational resources (Fig. 1). If too few compounds pass the user-specified filter(s), AutoGrow4 automatically returns to the mutation and crossover operators to generate more candidate molecules (Fig. 1).

AutoGrow4 includes the nine predefined molecular filters shown in Table 1. Users can combine any of these filters in series. The new modular codebase also makes it easy for users to add their own custom filters that assess other molecular properties.

Conversion of SMILES to 3D PDB

AutoGrow4 uses the open-source program Gypsum-DL [11] to convert the SMILES representations of all new molecules into 3D models for docking. For each input SMILES, Gypsum-DL generates one or more 3D models with alternate ionization, tautomeric, chiral, cis/trans isomeric, and ring-conformational forms [11, 27]. The user can specify the pH range to use for protonation as well as the maximum number of molecular forms (variants) that Gypsum-DL should produce per input SMILES.

Assessing fitness

AutoGrow4 uses two metrics to assess the fitness of each solution. These fitness scores are used to select seeds for the next generation (see below). The primary fitness

metric assesses how well each compound is predicted to bind the target receptor. By default, AutoGrow4 uses the docking score of the top-scoring pose. AutoGrow4 has broader docking-program support than previous versions. It supports not only AutoDock Vina [21] but also QuickVina 2.1 (QVina2) [10], a Vina-based program that runs about twofold faster. AutoGrow4 can also rescore docked poses using the NNScore1 and NNScore2 scoring functions [28, 29]. The score associated with each compound can optionally be divided by the number of non-hydrogen ligand atoms, a metric called ligand efficiency [30]. AutoGrow4's modular, plugin-based architecture also enables long-term expandability and user customization. With the appropriate plugin code, virtually any method for docking and/or reassessing ligand binding can be integrated into AutoGrow4, including quantitative structure–activity relationship (QSAR) approaches, ligand-similarity evaluations, etc.

Aside from this primary fitness metric, AutoGrow4 also calculates a secondary fitness metric called the diversity score. The diversity score is optionally used to select seed compounds that are structurally unique compared to those of the previous generation. By seeding a new generation with a population comprised of molecules separately selected for target binding and molecular diversity, AutoGrow4 delays population convergence while still refining for the desired binding affinity.

To determine the diversity score, AutoGrow4 uses RDKit [26, 31, 32] to calculate the Morgan fingerprint of each population compound after successful docking. The similarity s of two compounds mol_A and mol_B is given by

$$s(F_A, F_B) = \frac{2|F_A \cap F_B|}{|F_A| + |F_B|} \quad (1)$$

where F_A and F_B are the fingerprint bit-strings of mol_A and mol_B , respectively. The value of s ranges from 0.0 (completely different) to 1.0 (perfectly matched). The diversity score d of a given molecule mol_M measures its uniqueness relative to the other molecules in its generation. The score is calculated by

$$d(mol_M) = \sum_{N \neq M}^n s(F_M, F_N) \quad (2)$$

where the summation is over all mol_N within the generation except mol_M .

Compound ranking and seed selection

AutoGrow4 implements Ranking, Roulette, and Tournament selection strategies [33] to choose which compounds will seed the next generation. Each strategy has its advantages and disadvantages. A Ranking selector simply chooses the best-scoring solutions. It quickly

identifies local optima but often produces inbred populations of highly similar compounds [33]. In extreme cases, population convergence can cause AutoGrow4 to perpetually recreate very similar compounds each generation, without substantial improvement in fitness. A Roulette selector assigns each solution to a metaphorical roulette wheel, where the size of each area is weighted by fitness. By incorporating randomness into each generation, Roulette selection reduces the chances of becoming trapped in local optimum. But it gives all potential solutions—even the most unfit—an opportunity to advance [33]. Lastly, a Tournament selector randomly chooses a subpopulation of solutions and then selects the fittest solutions from that subpopulation. A Tournament selector incorporates more randomness than a Ranking selector while reducing the risk of selecting unfit solutions [33].

AutoGrow4 performs three subselections per generation, one for elitism, mutation, and crossover operations, respectively (Fig. 1). When using a stochastic (i.e., Roulette or Tournament) selector, these subselections are independent (i.e., each subselection picks a different but potentially overlapping set of compounds). When using a deterministic (i.e., Ranking) selector, the three subselections are identical. In all cases, each subselection separately selects compounds based on their primary (binding) and/or secondary (diversity) scores. The user can also specify the number of compounds selected based on each score type.

Benchmark AutoGrow runs

Protein preparation

We tested AutoGrow4 against the PARP-1 catalytic domain. We first obtained the 4R6E structure from the Protein Data Bank (PDB) [18, 34] and removed all atoms but those of the chain A protein (4R6E:A). We used the PDB2PQR server (2.1.1, default settings) [35, 36] to add hydrogen atoms and optimize the hydrogen-bond network (pH 7). We then converted the resulting PQR file back to the PDB format using OpenBabel (2.3.1) [37].

To define the location of the binding pocket, we selected five protein residues that flank the bound, crystallographic niraparib ligand: E763, I872, G888, T907, and E988. We used the Scoria Python library [38] to calculate a bounding box that encompasses these residues. We expanded the width, length, and height of this box by a few Å to ensure that it entirely surrounded the pocket. The docking box ultimately had dimensions 25.0 Å × 16.0 Å × 25.0 Å, centered on the binding pocket. We used this PARP-1 PDB file and docking box for all AutoGrow runs (see published tutorial).

Comparison benchmark runs

AutoGrow 3.1.3 provides a set of 117 PDB-formatted small molecules with naphthalene substructures as source compounds [5]. To compare AutoGrow4 and AutoGrow 3.1.3, we converted these compounds to SMILES using OpenBabel 2.3.1 [37] and RDKit [26]. They are included as an SMI file in the AutoGrow4 download.

We ran the AutoGrow4 and AutoGrow 3.1.3 benchmarks on the same hardware: 12-core Xeon E5-2643v4 3.40 GHz Broadwell nodes with 512 GB RAM, provided by the University of Pittsburgh's Center for Research Computing (CRC). We also closely matched the AutoGrow4 and AutoGrow 3.1.3 settings in terms of processor counts, population sizes, mutation reaction sets, seed molecules, and population-size/seed-size ratios. In all cases, we subjected the evolving molecules to the Ghose* and Lipinski* filters (Table 1). The complete settings are provided in Additional file 1: JSON 1 and 2.

AutoGrow 3.1.3 has some notable limitations, requiring several additional considerations. AutoGrow 3.1.3 is not Python 3 compatible, so we ran these benchmarks in a Python 2.7 environment. We also limited AutoGrow4 to one molecular variant per input SMILES, Vina 1.1.2 docking, the Ranking selector, and symmetric multiprocessing (SMP) because AutoGrow 3.1.3 does not consider alternate molecular forms, cannot use QVina2, does not implement the Roulette or Tournament selectors, and does not support message passing interface (MPI) multiprocessing [5].

All benchmark AutoGrow4 and AutoGrow 3.1.3 runs were repeated independently 24 times.

Large-scale de novo PARPi run

We generated a sizable, chemically diverse source library for general use when performing de novo AutoGrow4 runs. We started with the same large set of Lipinski*-filtered Zinc15 molecules described above, which was previously used to generate the default complementary small-molecule libraries required for the mutation operator. We discarded those compounds that lacked functional groups capable of participating in at least one of AutoGrow4's 94 default reactions. The remaining compounds were grouped by MW (≤ 100 Da, 100 Da $<$ MW ≤ 150 Da, 150 Da $<$ MW ≤ 200 Da, and 200 Da $<$ MW ≤ 250 Da). To keep the source library reasonably sized while maintaining chemical diversity, we randomly discarded excess compounds in each MW category that had overrepresented functional groups. Ultimately, at most 100 compounds remained for each functional group in each weight range. These four source-library sets, which collectively comprise 24,595 molecules, are included in the AutoGrow4 download.

The large-scale de novo run described in "Results and discussion" was seeded with small molecules from the 100 Da $<$ MW ≤ 150 Da category. This run used MPI multiprocessing, QVina2 docking, and the Ranking selector in a Python 3.7 environment. It ran for thirty generations on ten MPI-enabled CRC computer nodes with 28-core Broadwell Processors and 64 GB RAM/node, networked with Intel's Omni-Path communication architecture. Evolving molecules were subjected to the Ghose, Lipinski*, and PAINS filters (Table 1). In the first generation, AutoGrow4 generated 40, 500, and 500 compounds via elitism, mutation, and crossover, respectively. In subsequent generations, it generated 500, 2500, and 2500 compounds via elitism, mutation, and crossover, respectively. The complete settings are provided in Additional file 1: JSON 3. We used BlendMol [39] to generate figures of representative docked molecules.

PARPi lead-optimization runs

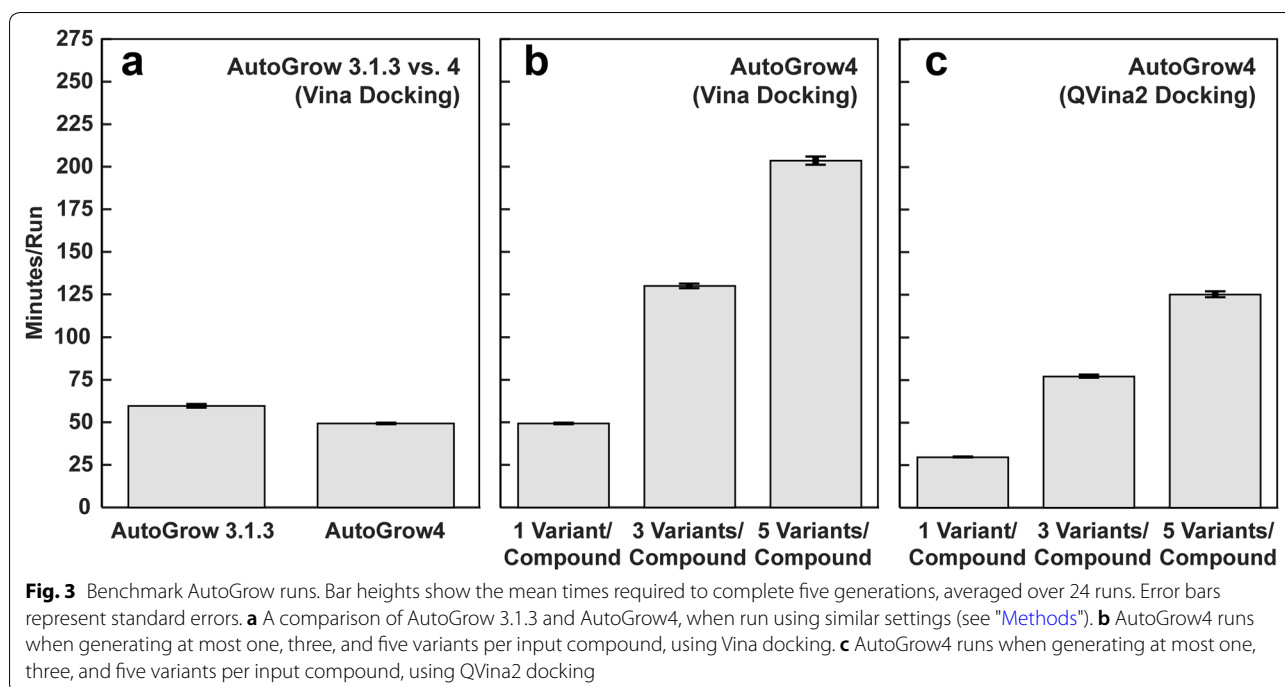
To show how AutoGrow4 can be used for lead optimization, we generated a focused source library of 94 seed molecules. This library includes eleven known PARPi, identified using <http://www.clinicaltrials.gov>. It also includes 83 PARPi molecular fragments derived from those eleven via Breaking of Retrosynthetically Interesting Chemical Substructures (BRICS) decomposition [7].

We used AutoGrow4 with this small source library to predict new PARP-1 ligands similar to known inhibitors. To focus computational effort on the chemistry space near known PARPi, we ran AutoGrow4 for only five generations, but with demanding docking-exhaustiveness, variant-per-molecule, and population-size settings. Evolving molecules were subjected to the Ghose, Lipinski*, and PAINS filters (Table 1). The complete settings are provided in Additional file 1: JSON 4. The settings of the first generation were conservative because the small source library had a limited number of reactive functional groups. We thus limited the first generation's size to 40, 500, and 500 molecules derived using the elitism, mutation, and crossover operators, respectively. Subsequent generations were larger. They included 250, 2500, and 2500 molecules derived using the elitism, mutation, and crossover operators, respectively.

These lead-optimization runs were repeated six independent times using Python 3.7 on the same MPI-enabled nodes available through the CRC. We again used BlendMol [39] to generate figures of representative docked molecules.

PARPi-like compounds: AutoGrow4 optimization vs. similarity-based screening

To compare AutoGrow4 lead optimization to a more traditional similarity-based virtual screening (VS) approach



[40, 41], we generated a library of small molecules that are structurally similar to known PARPi. We downloaded the structures of ~2200 known PARPi from the BindingDB database [42, 43] on March 14, 2020. Many of these compounds were close analogues, so we used a Tanimoto-based clustering algorithm [44] (Tanimoto cutoff: 0.65) to group the compounds into 40 clusters. We then selected one molecule per cluster to construct a set of structurally unique known PARPi.

For each of these PARPi, we next downloaded at most 250 compounds from PubChem [45] with corresponding Tanimoto coefficients greater than 0.80 (8444 unique molecules). To maximize structural diversity and reduce the number of compounds, we again applied the clustering algorithm (Tanimoto cutoff: 0.2), yielding a set of 4631 PARPi-like molecules that (1) were not in the original PARPi set and (2) could be docked into PARP-1 with QVina2.

Results and discussion

Poly(ADP-ribose) polymerase 1

We used PARP-1, a protein critical for DNA repair, as a test system to demonstrate AutoGrow4 utility. DNA is under constant threat of damage by ionizing radiation, UV radiation, and reactive oxygen species [16, 46]. Base excision repair (BER) is a crucial pathway for repairing single-strand DNA (ssDNA) breaks, and non-homologous end joining (NHEJ) and homologous recombination (HR) are critical pathways for repairing double-strand

DNA (dsDNA) breaks [16, 46–49]. When PARP-1 recognizes sites of DNA damage, it begins to convert NAD⁺ molecules into a negatively charged network of poly(ADP-ribose) (PAR) chains [16]. These PAR chains are covalently attached to nearby proteins, including PARP-1 itself, through a process known as PARylation [16]. This PARylation signal recruits DNA repair proteins (e.g., NHEJ and BER) [16, 50].

Defects in the *BRCA 1* and *2* genes, prevalent in breast and ovarian cancers, cause loss of HR repair function [16]. HR-defective cancer cells rely heavily on NHEJ and BER to compensate, so much so that further loss of NHEJ and BER is lethal [16]. PARPi capable of disrupting BER and NHEJ signaling are thus potential therapeutics for treating HR-defective breast-cancer cells [16, 17]. In contrast, non-cancerous cells survive PARPi exposure because their HR-repair mechanisms are intact [16, 17]. HR is also most active during the S to M transition, so actively dividing tumor cells are especially vulnerable [16, 51, 52]. Four PARPi are FDA approved (olaparib, rucaparib, niraparib, and talazoparib), and several more are in clinical trials [12–15].

Comparison Benchmark runs

To compare AutoGrow4 and its predecessor AutoGrow 3.1.3, we ran both programs 24 times using similar settings. On average, AutoGrow4 completes five generations

1.21 times faster than AutoGrow 3.1.3 (59.64 vs 49.34 min/run, see Fig. 3a).

We also compared AutoGrow4 performance under different user-parameter conditions. AutoGrow4 has been outfitted with many new features not available in previous versions. Of note, it uses Gypsum-DL [11] to generate alternate ionization, tautomeric, chiral, cis/trans isomeric, and ring-conformational variants of each input SMILES string. Figure 3b shows that AutoGrow4 run times vary roughly linearly with the specified maximum number of variants per input molecule (Additional file 1: JSON 2). It is worth noting that in practice Gypsum-DL often produces fewer variants than the maximum specified. In these benchmarks, specifying a maximum of three and five produced 2.5 and 3.9 variants per compound.

Finally, we assessed AutoGrow4 performance when docking with QVina2 (Fig. 3c). Compared to the other steps in our algorithm, docking is particularly time consuming. AutoGrow4's ability to dock with QVina2 in addition to Vina is a notable improvement over previous versions. When set to generate at most five variants per compound, AutoGrow4 runs 1.6 times faster when docking with QVina2 vs. Vina (125.04 vs. 203.52 min/run, respectively; Fig. 3b, c).

Large-scale de novo PARPi run

Predicted ligands

We performed an extensive PARP-1 run to show how AutoGrow4's parallelization and multiprocessing capabilities enable large-scale de novo design. These runs used a large, diverse library of seed molecules and fragments to produce high-scoring compounds such as Compound 4, which has a QVina2-predicted binding affinity of -16.7 kcal/mol (Fig. 4). This predicted ligand has a 1*H*-naphtho[2,3-*d*][1,2,3]triazole substructure that forms π - π stacking interactions with the PARP-1 Y907, H862, and Y896 residues. Interestingly, two of these residues (H862 and Y896) belong to the PARP-1 catalytic triad, which is conserved in PARP-1 through PARP-6 [53]. Additional hydrogen bonds form between the compound's cyclic nitrogen atoms and the backbone atoms of G863, R865, and R878. An electrostatic interaction with D766 is also possible, depending on the protonation states of D766 and the compound tetrazolidine substructure.

A caution regarding chemical properties

Though longer AutoGrow4 runs can produce compounds with remarkable scores, we generally recommend multiple independent runs with fewer generations. Longer runs have several disadvantages. First, the evolving compounds increasingly take on chemical properties that

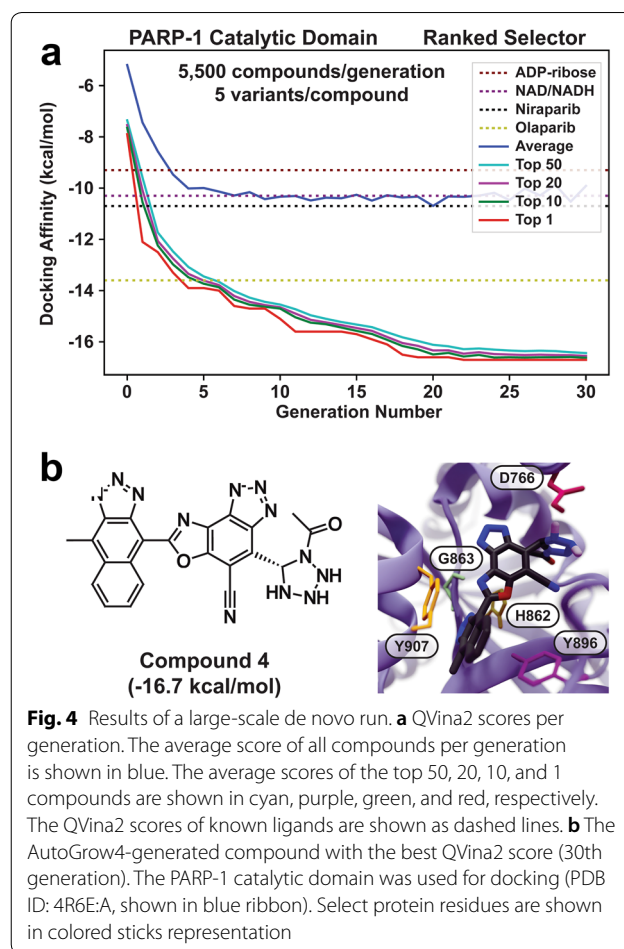
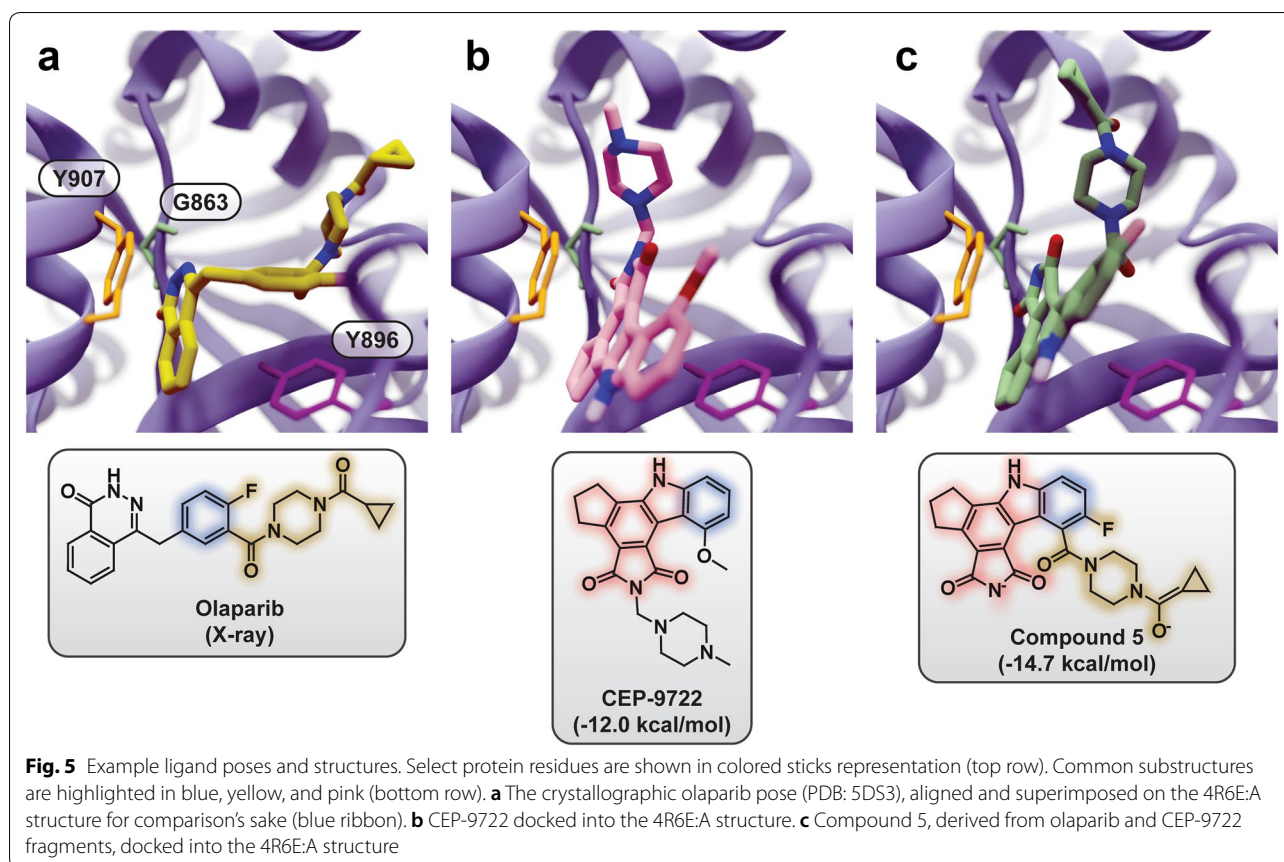


Fig. 4 Results of a large-scale de novo run. **a** QVina2 scores per generation. The average score of all compounds per generation is shown in blue. The average scores of the top 50, 20, 10, and 1 compounds are shown in cyan, purple, green, and red, respectively. The QVina2 scores of known ligands are shown as dashed lines. **b** The AutoGrow4-generated compound with the best QVina2 score (30th generation). The PARP-1 catalytic domain was used for docking (PDB ID: 4R6E:A, shown in blue ribbon). Select protein residues are shown in colored sticks representation

are artefactually favored by the fitness function and/or ligand-creation operations. For example, Compound 4 (Fig. 4b), one of the best-scoring compounds produced by the extensive PARP-1 de novo run, has a molecular weight (MW) near the 480 Da maximum that the applied Ghose filter permits (478.1 Da) [54]. The Vina scoring function is known to favor larger molecules [55], perhaps explaining in part this apparent evolutionary tendency towards increased MW. Filters that place tighter restrictions on MW, as well as ligand-efficiency rescoring [30], can mitigate this bias.

Second, longer runs can lead to the accumulation of undesirable moieties. The 24th-generation compound shown in Additional file 1: Figure S1, one of the highest scoring compounds from our large-scale de novo run, provides a good example. This compound possesses azo and ethyne moieties, which belong to a broad category of substructures thought to be mutagenic, pharmacokinetically unfavorable, reactive, and/or likely to interfere with typical high-throughput screening approaches [56]. This challenge, typical of longer runs, can be mitigated by



applying the appropriate filter(s) (e.g., the BRENK filter [56]).

Third, compound synthesizability, also a critical chemical property, similarly tends to diminish in later generations as the accumulation of mutation and crossover events causes the population to drift from the source molecules.

A caution regarding homogeneity and convergence

Long runs also suffer from population homogeneity and premature convergence. In this scenario, the fitness scores of existing molecules are so good that new compounds generated via mutation and crossover can rarely outcompete them [57]. Compound fitness thus tends to improve quickly in the earliest generations but stalls in later generations despite consuming comparable computational resources. For example, the average docking score of the top 50 molecules in our large-scale de novo run improved -6.09 kcal/mol from generation zero to five (-7.36 kcal/mol to -13.45 kcal/mol, respectively), but it only improved another -2.99 kcal/mol by generation 30 (-16.44 kcal/mol). The populations began to converge by generation 20, with only minor subsequent improvements in fitness (Fig. 4a).

AutoGrow4 uses several strategies to avoid population convergence and homogeneity. First, its sizable libraries of diverse seed molecules encourage the exploration of a large subset of chemistry space, at least in early generations. Second, it considers both primary (binding) and secondary (diversity) scores when selecting molecules for elitism, mutation, and crossover operations (Fig. 1). By seeding each generation with a combination of well docked and unique compounds, AutoGrow4 aims to search more of chemistry space while still maintaining a selective pressure for reasonable predicted ligands. Finally, AutoGrow4 provides different selection strategies (e.g., Roulette and Tournament) that may delay convergence. Despite these measures, multiple independent runs of fewer generations are typically more computationally efficient.

PARPi lead-optimization runs

AutoGrow4 applied to lead optimization

We performed six short PARP-1 runs to show how AutoGrow4 is a useful tool for lead optimization. These runs used 94 known PARPi and PARPi fragments as seeds rather than a large library of diverse molecular fragments. Our ultimate goal was to evolve molecules

that are chemically similar to known ligands, but with improved docking scores. To perform a narrow but thorough search of the chemistry space centered around the initial leads, we ran each AutoGrow4 run for only five generations. But we used large population sizes (see "Methods") and increased the QVina2 *exhaustiveness* parameter to 25 to improve the chances of finding optimal docked poses.

By the third generation, the average QVina2 score of the top-20 compounds across all six runs (the grand mean) already matched the score of the best-scoring known PARPi, olaparib (-13.6 kcal/mol, AstraZeneca). By the fifth generation, the grand mean of the top-50 compounds (-14.0 kcal/mol) was better than the olaparib score (Additional file 1: Figure S2).

Of all the molecules generated during the six PARPi lead-optimization runs, compound 5 had one of the best QVina2 scores (Fig. 5 and Additional file 1: Figure S3, -14.7 kcal/mol). We focused subsequent analysis on this molecule rather than the best-scoring compound (Fig. 2b) because compound 5 was derived from two source-library PARPi fragments (Olaparib_{Frag3} and CEP-9722_{Frag1}), the result of a crossover in the first generation (Additional file 1: Figure S3). It thus provides an excellent example of AutoGrow4-guided lead optimization.

Compound 5 participates in π - π stacking interactions with the PARP-1 Y907 and Y896 residues. One of its carbonyl oxygen atoms also forms a hydrogen bond with G863 (Fig. 5c). These interactions are typical of the binding modes of known PARPi [18–20, 58–66] such as the crystallographic olaparib pose (Fig. 5a) [19] and a docked CEP-9722 pose (Cephalon, Fig. 5b).

In constructing compound 5, AutoGrow4 attached an olaparib-derived piperazine moiety at a different position than the CEP-9722 piperazine, but the docked poses of both compounds orient their respective piperazines similarly. Interestingly, this orientation differs from that of the olaparib piperazine (Fig. 5). We note that the crystallographic poses of several other PARPis position piperazine moieties at alternate locations [19, 66–68]. For example, the crystal structure of the potent PARPi EB₄₇ [69, 70] bound to PARP16 (PDB ID: 6HXR) places a piperazine substructure near that of our compound 5 docked pose. Notably, EB₄₇ was not among the source-library compounds used for the PARP-1 lead-optimization runs.

AutoGrow4 vs. other docking techniques for lead optimization

Similarity-based virtual screening (VS) is a popular technique for in silico ligand optimization [40, 41]. One first generates a library of compounds that are chemically similar to known ligands. VS is then used to prioritize the compounds in hopes of ultimately identifying molecules

that bind with better predicted affinities than those of the known ligands. Two methods for identifying chemically similar compounds are common: substructure and similarity searching.

In the substructure scheme, the compound library consists of molecules that share substructures in common with known ligands. The generation-1 compounds of each AutoGrow4 lead-optimization run form such a library because (1) they are derived from the PARPi fragments/molecules of generation 0, and (2) both the mutation and crossover operators generate compounds that share substructures in common with parent molecules. The progress made from generation 1 to generation 5 thus illustrates how AutoGrow4 optimization can improve scores beyond substructure-based VS alone. As shown in Additional file 1: Figure S2, the grand-mean docking score of the top 50 compounds from generation 1 (~1% of all compounds screened that generation) was -12.3 kcal/mol. Following five generations of AutoGrow4 optimization, that grand-mean score improved to -14.0 kcal/mol.

In the similarity-searching scheme, the compound library consists of molecules that are structurally similar in their entirety to known ligands, per a whole-molecule metric such as the Tanimoto coefficient [71]. To compare this in silico optimization method to our GA approach, we generated a library of 4631 PARPi-like molecules. These compounds were processed with Gypsum-DL [11] and docked into PARP-1 with QVina2 [10] using the same parameters used in the AutoGrow4 lead-optimization runs.

The average docking score of the top-50 compounds from this similarity library (~1% of all unique compounds docked) was -12.7 kcal/mol. In contrast, five generations of AutoGrow4 optimization produced compound sets with top-50 average scores around -14.0 kcal/mol. This comparison of course has its limitations. Generating the similarity library using stricter Tanimoto cutoffs would have likely improved the average scores, though at the expense of structural diversity. In contrast, had we run the AutoGrow4 lead-optimization runs for additional generations, our GA method would have likely identified compounds with improved docking scores. But the comparison nevertheless contextualizes the AutoGrow4 approach.

AutoGrow4 operators and molecular weight

In evaluating AutoGrow4, we also carefully studied the program's tendency to evolve compounds with increasing MW. The AutoGrow4 mutation and crossover operators could in theory drive this tendency. On the other hand, larger ligands often form more molecular interactions with their targets, so the observed MW increases over

Table 2 The protein–ligand interactions of the 100 best-docked compounds from the large-scale de novo run, per BINANA

	D766	D770	H862	G863	R865	R878	Y907
Cation- π	0	0	98	0	0	0	0
Hydrogen bond	3	0	1	100	56	47	1
Electrostatic	100	41	3	0	0	9	0
T-stacking	0	0	71	0	0	0	0
π - π	0	0	100	0	0	0	100

Infrequent interactions (< 10%) are excluded. Values are given as percents

time may reflect physiochemical reality. The Vina scoring function may also explain the tendency towards greater MW, given its known bias in favor of larger molecules [55].

To determine the role AutoGrow4 operators play in the observed tendency, we first explored the impact of the mutation operator on MW. From among all the operations performed during the six PARPi lead-optimization runs, we identified 55,683 mutation events involving an AutoGrow4-generated reactant. On average, each of these operations increased MW by 28% (66.0 Da). But in 11% of cases, the MW decreased. To illustrate how this is possible, consider the transesterification of phenyl benzoate and methanol. The resulting products (methyl benzoate and phenol) both have MWs less than the phenyl benzoate reactant.

We next explored the impact of the crossover operator on MW. We identified 50,169 crossover events from the six PARPi lead-optimization runs that involved two AutoGrow4-generated parent compounds. On average, the MW of the resulting child compound was only 5% larger (11.5 Da) than the average MW of the two parents. In 43% of cases, the MW decreased because the child molecule inherited a low-weight set of decorating moieties from the parents.

These results show that the mutation and crossover operators may promote some compound growth, but they often reduce compound size as well. Users who wish to limit AutoGrow's tendency towards larger-MW molecules may be interested in the included molecular filters that place tighter restrictions on MW (e.g., Lipinski [72], Ghosh [54]). We also recommend source (generation 0) populations comprised of small molecular fragments to maximize the number productive AutoGrow4 generations executed before running into MW filter cutoffs. Increasing the number of crossover operations per generation may also effectively control MW, given that crossovers are more likely to reduce MW than are mutations. Finally, users can instruct AutoGrow4 to rescore docked molecules by ligand efficiency [30], which normalizes docking scores by the number of compound heavy atoms and so penalizes larger molecules.

Identifying critical protein–ligand interactions

Beyond de novo generation and lead optimization, AutoGrow4 provides a systematic way of identifying pharmacologically important protein–ligand interactions. Cataloguing the most common interactions among top-scoring AutoGrow4-generated compounds can inform subsequent experiments ranging from QSAR drug design to site-directed mutagenesis.

The large-scale de novo run provides many useful examples of compounds with high predicted affinities. We identified the 100 compounds with the best docking scores from among the hundreds of thousands of docking events performed over 30 generations of evolution. We then used the BINANA 1.1.2 algorithm [73] to automatically characterize the protein–ligand interactions of each best-docked pose. Four interactions were present in all 100 docked poses: two separate π - π stacking interactions with Y907 and H862, an electrostatic interaction with D766, and a hydrogen-bond interaction with G863 (Table 2). Several other interactions were prevalent, though not universal: an electrostatic interaction with D770 (41%), a hydrogen-bond interaction with R865 (56%), and electrostatic and hydrogen-bond interactions with R878 (9% and 47%, respectively) (Table 2). The crystallographic poses of known PARPi (e.g., olaparib) participate in many of the same interactions seen among the top AutoGrow4 compounds [19]. Our large-scale de novo run—which was seeded with random molecular fragments not necessarily related to known PARPi—thus serves as a blind-study validation of AutoGrow4's ability to identify pharmacologically important catalytic-pocket interactions.

Given that many of the AutoGrow4 generated compounds have better docking scores than known PARPi, the in silico compounds provide insight into future optimization strategies. For example, all 100 of the top-scoring AutoGrow4-generated compounds (large-scale de novo run) form electrostatic interactions with D766, but olaparib does not. Adding a positively charged moiety to the olaparib piperazine might enable an additional interaction with D766.

Table 3 A comparison of several de novo design programs

Program	FOSS	Docking options	MPI enabled	OS
AutoGrow4	Yes	Vina/QVina2/customizable	Yes	Linux/macOS/ Windows (via Docker)
MoleGear [92]	No	Autodock and Vina	Yes	Unspecified
GANDI [96]	Yes	DAIM/SEED/FFLD	Yes	Linux
de novo DOCK [94]	Yes	DOCK	Yes	Linux/macOS
REINVENT [100]	Yes	N/A	Unspecified	Linux/macOS
LigDream [101]	Yes	N/A	Unspecified	Unspecified

FOSS stands for “free and open source software”

These results reinforce the critical role Y907 plays in high-affinity binding. Many known co-crystallized PARPi participate in a π - π stacking interaction with Y907 [18–20, 58–66]. The top-100 compounds (per the docking score) produced in both the large-scale de novo and lead-optimization runs are all predicted to interact with Y907, suggesting that this interaction may be broadly critical regardless of the chemical scaffold. Unfortunately, interactions with Y907 also raise concerns for the future of orthosteric PARPi. When the receptor tyrosine kinase c-Met phosphorylates Y907, PARP-1 catalytic activity increases and PARPi binding affinity weakens [74]. c-Met phosphorylation thus provides a potential mechanism for PARPi resistance [74]. Consequently, there is a need for improved PARPi that do not rely on any interaction with Y907. Based on our AutoGrow4 results, we hypothesize that developing catalytic-pocket inhibitors that do not interact with Y907 will be difficult. A better strategy may be to target other (allosteric) pockets or to pursue cocktail treatments that inhibit both PARP-1 and c-Met.

Comparison with other programs

Over the years, a number of programs have been developed to assist with de novo drug design [75–91]. A comprehensive review is beyond the scope of this article, but a few programs, summarized in Table 3, warrant specific mention. MoleGear is a recently published algorithm that also takes an evolutionary approach to de novo drug design [92]. It provides a graphical user interface and allows users to dock compounds with either AutoDock [93] or AutoDock Vina [21]. But despite its recent publication, MoleGear does not appear to be publicly available, and the program is closed source.

In contrast, de novo DOCK [94] is an open-source algorithm that is integrated into the DOCK6 docking program itself [95]. To produce novel compounds, it uses an iterative fragment-growth method that is based on the DOCK6 anchor-and-grow search algorithm [94]. The method first identifies core components of a given

compound, referred to as anchors, and then expands that anchor layer by layer via fragment addition. Though de novo DOCK is a powerful program, AutoGrow4 has several advantages. First, AutoGrow4 is not tied to a specific docking program. Users can choose between Vina and QVina2 docking by default, and AutoGrow4's plugin-based architecture makes it easy to incorporate other docking programs as well. Second, AutoGrow4 uses high-yielding in silico chemical reactions to generate compounds via mutation. In contrast, de novo DOCK does not provide a reaction-based mutation operator.

The free and open-source program GANDI takes a different approach to de novo design [96]. It first docks and scores molecular fragments using the DAIM [97], SEED [98], and FFLD [99] programs. It then joins promising fragments via a molecular linker taken from a pre-defined look-up table. GANDI uses a GA that employs a parallel-model approach, often referred to as an island model [96]. It evolves multiple populations separately, only occasionally swapping molecules between them. GANDI's fragment-and-linker approach, though effective, does limit the search space to compounds that can be generated using a pre-defined set of linkers. In contrast, AutoGrow4 effectively allows any linker regions to evolve with the rest of the compound.

Recent efforts have also used machine learning for de novo design. For example, the open-source program REINVENT [100] uses recurrent neural networks and reinforcement learning to generate de novo compounds. LigDream, another example, uses a convolutional neural network [101] and focuses training instead on the 3D shapes of known ligands. Machine-learning approaches such as these are effective, but they must often be trained on preexisting ligands. In contrast, AutoGrow4 can generate compounds in the absence of known inhibitors (see the large-scale de novo run above).

Conclusions

AutoGrow4 is a powerful program for hit discovery and lead optimization, particularly when paired with expert knowledge about the target pocket and any known ligands. We view AutoGrow4 as an open-source tool for hypothesis generation. It effectively narrows the vast scope of all possible compounds to a subset of candidate ligands. Expert users must then apply their own biological and chemical understanding to properly interpret the results and to ensure that the generated compounds are chemically feasible.

AutoGrow4 is available free of charge under the terms of the open-source Apache License, version 2.0. A copy of the latest version can be downloaded from <http://durrantlab.com/autogrow4>, and an archived copy is provided as Additional file 2. AutoGrow4 is compatible with both Python 2.7 and 3.7. Users must separately install the following third-party Python-library dependencies: RDKit [26], numpy [102, 103], scipy [104], matplotlib [105], and func_timeout (available via *pip*). If the mpi4py Python package is installed [106], AutoGrow4 can leverage multiple processors using MPI on MPI-enabled clusters. Finally, to convert structures from the PDB to the PDBQT format for use with Vina and QVina2, AutoGrow4 requires either AutoDock MGLTools or Open Babel [37, 93].

Installation instructions for AutoGrow4 and its dependencies are provided in the AutoGrow4 tutorial. AutoGrow4 runs on Linux and macOS. We strongly encourage use of the AutoGrow4 Docker container (Docker, Inc.), included in the download, which automatically installs all dependencies and further enables use on Windows.

Supplementary information

Supplementary information accompanies this paper at <https://doi.org/10.1186/s13321-020-00429-4>.

Additional file 1. The Additional file includes detailed descriptions of the AutoGrow 3.1.3 and AutoGrow4 parameters used in the benchmark and PARP-1 runs. It also includes Figures S1, S2, and S3, referenced in the text.

Additional file 2. An archive of the AutoGrow4 source code. See <http://durrantlab.com/autogrow4> for the latest version.

Acknowledgements

We would like to thank Patrick J. Ropp for useful discussions and programming tips; Erich Hellemann for discussions and help with SMARTS reactions; Pauline Spiegel for manuscript editing and tutorial testing; Yuri Kochnev for compiling QVina2 for macOS; Kevin C. Cassidy for help with molecular rendering; and Harrison Green for help developing AutoGrow4 accessory scripts. We would like to thank the University of Pittsburgh's Center for Research Computing for providing helpful computer resources. The default fragment libraries included with AutoGrow4 were derived from a subset of the ZINC database (<https://zinc.docking.org/>). We thank ZINC for allowing us to distribute these fragment libraries to AutoGrow4 users.

Availability and requirements

Project name: AutoGrow4
Project home page: <http://durrantlab.com/autogrow4>
Operating systems: Linux, macOS, and Windows (via Docker)
Programming language: Python 2/3
Other requirements: RDKit, NumPy, SciPy, func_timeout, mpi4py (optional), MGLTools (optional), Open Babel (optional)
License: Apache License, Version 2.0

Authors' contributions

JOS and JDD conceived of and planned the study. JOS rewrote the AutoGrow4 codebase and performed the computational validation. JOS and JDD wrote the manuscript and prepared figures together. All authors discussed the results and contributed to the final manuscript. All authors read and approved the final manuscript.

Funding

None.

Availability of data and materials

All data and materials used to validate AutoGrow4 are included in the AutoGrow4 download.

Competing interests

The authors declare that they have no competing interests.

Received: 3 February 2020 Accepted: 31 March 2020

Published online: 17 April 2020

References

- Ertl P (2003) Cheminformatics analysis of organic substituents: identification of the most common substituents, calculation of substituent properties, and automatic identification of drug-like bioisosteric groups. *J Chem Inf Comput Sci*. <https://doi.org/10.1021/ci0255782>
- Bohacek RS, McMartin C, Guida WC (1996) The art and practice of structure-based drug design: a molecular modeling perspective. *Med Res Rev*. [https://doi.org/10.1002/\(SICI\)1098-1128\(199601\)16:1<3::AID-MED1>3.0.CO;2-6](https://doi.org/10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6)
- Polishchuk PG, Madzhidov TI, Varnek A (2013) Estimation of the size of drug-like chemical space based on GDB-17 data. *J Comput Aided Mol Des*. <https://doi.org/10.1007/s10822-013-9672-4>
- Durrant JD, Amaro RE, McCammon JA (2009) Autogrow: a novel algorithm for protein inhibitor design. *Chem Biol Drug Des* 73(2):168–178. <https://doi.org/10.1111/j.1747-0285.2008.00761.x>
- Durrant JD, Lindert S, McCammon JA (2013) Autogrow 3.0: an improved algorithm for chemically tractable, semi-automated protein inhibitor design. *J Mol Graph Model* 44:104–112. <https://doi.org/10.1016/j.jmgm.2013.05.006>
- Sliwoski G, Kothiwale S, Meiler J, Lowe EW (2014) Computational methods in drug discovery. *Pharmacol Rev*. <https://doi.org/10.1124/pr.112.007336>
- Degen J, Wegscheid-Gerlach C, Zaliani A, Rarey M (2008) On the art of compiling and using 'drug-like' chemical fragment spaces. *ChemMedChem*. <https://doi.org/10.1002/cmdc.200800178>
- Yu W, MacKerell AD (2017) Chap. 5. Computer-aided drug design methods. In: Sass P (ed) *Antibiotics methods in molecular biology*, vol 1520. Springer, Berlin, pp 85–106
- Kawai K, Nagata N, Takahashi Y (2014) De novo design of drug-like molecules by a fragment-based molecular evolutionary approach. *J Chem Inf Model*. <https://doi.org/10.1021/ci400418c>
- Alhossary A, Handoko SD, Mu Y, Kwok CK (2015) Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics* 31(13):2214–2216. <https://doi.org/10.1093/bioinformatics/btv082>
- Ropp PJ, Spiegel JO, Walker JL, Green H, Morales GA, Milliken KA, Ringe JJ, Durrant JD (2019) Gypsum-dl: an open-source program for preparing small-molecule libraries for structure-based virtual screening. *J Cheminform* 11(1):34. <https://doi.org/10.1186/s13321-019-0358-3>

12. Kim G, Ison G, McKee AE, Zhang H, Tang S, Gwise T, Sridhara R, Lee E, Tzou A, Philip R et al (2015) Fda approval summary: olaparib monotherapy in patients with deleterious germline BRCA-mutated advanced ovarian cancer treated with three or more lines of chemotherapy. *Clin Cancer Res* 21(19):4257–4261
13. Balasubramaniam S, Beaver JA, Horton S, Fernandes LL, Tang S, Horne HN, Liu J, Liu C, Schrieber SJ, Yu J, Song P, Pierce W, Robertson KJ, Palmby TR, Chiu HJ, Lee EY, Philip R, Schuck R, Charlab R, Banerjee A, Chen XH, Wang X, Goldberg KB, Sridhara R, Kim G, Pazdur R (2017) FDA approval summary: rucaparib for the treatment of patients with deleterious BRCA mutation-associated advanced ovarian cancer. *Clin Cancer Res* 23(23):7165–7170. <https://doi.org/10.1158/1078-0432.CCR-17-1337>
14. Ison G, Howie LJ, Amiri-Kordestani L, Zhang L, Tang S, Sridhara R, Pierre V, Charlab R, Ramamoorthy A, Song P, Li F, Yu J, Manheng W, Palmby TR, Ghosh S, Horne HN, Lee EY, Philip R, Dave K, Chen XH, Kelly SL, Janoria KG, Banerjee A, Eradiri O, Dinin J, Goldberg KB, Pierce WF, Ibrahim A, Kluetz PG, Blumenthal GM, Beaver JA, Pazdur R (2018) FDA approval summary: niraparib for the maintenance treatment of patients with recurrent ovarian cancer in response to platinum-based chemotherapy. *Clin Cancer Res* 24(17):4066–4071. <https://doi.org/10.1158/1078-0432.CCR-18-0042>
15. Hoy SM (2018) Talazoparib: first global approval. *Drugs* 78(18):1939–1946. <https://doi.org/10.1007/s40265-018-1026-z>
16. Bitler BG, Watson ZL, Wheeler LJ, Behbakht K (2017) PARP inhibitors: clinical utility and possibilities of overcoming resistance. *Gynecol Oncol* 147(3):695–704. <https://doi.org/10.1016/j.ygyno.2017.10.003>
17. Patel AG, Sarkaria JN, Kaufmann SH (2011) Nonhomologous end joining drives poly(ADP-ribose) polymerase (PARP) inhibitor lethality in homologous recombination-deficient cells. *Proc Natl Acad Sci USA* 108(8):3406–3411. <https://doi.org/10.1073/pnas.1013715108>
18. Thorsell AG, Ekblad T, Karlberg T, Löw M, Pinto AF, Trésaugues L, Moche M, Cohen MS, Schüler H (2017) Structural basis for potency and promiscuity in poly(ADP-ribose) polymerase (PARP) and tankyrase inhibitors. *J Med Chem*. <https://doi.org/10.1021/acs.jmedchem.6b00990>
19. Dawicki-McKenna JM, Langelier M-F, DeNizio JE, Riccio AA, Cao CD, Karch KR, McCauley M, Steffen JD, Black BE, Pascal JM (2015) Parp-1 activation requires local unfolding of an autoinhibitory domain. *Mol Cell* 60(5):755–768. <https://doi.org/10.1016/j.molcel.2015.10.013>
20. Kinoshita T, Nakanishi I, Warizaya M, Iwashita A, Kido Y, Hattori K, Fujii T (2004) Inhibitor-induced structural change of the active site of human poly(ADP-ribose) polymerase. *FEBS Lett* 556(1–3):43–6. [https://doi.org/10.1016/s0014-5793\(03\)01362-0](https://doi.org/10.1016/s0014-5793(03)01362-0)
21. Trott O, Olson AJ (2009) Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem* 31(2):455–461. <https://doi.org/10.1002/jcc.21334>
22. Durrant JD, McCammon JA (2012) Autoclickchem: click chemistry in silico. *PLoS Comput Biol*. <https://doi.org/10.1371/journal.pcbi.1002397>
23. Hartenfeller M, Eberle M, Meier P, Nieto-Oberhuber C, Altmann KH, Schneider G, Jacoby E, Renner S (2011) A collection of robust organic synthesis reactions for in silico molecule design. *J Chem Inf Model* 51(12):3093–3098. <https://doi.org/10.1021/ci200379p>
24. Sterling T, Irwin JJ (2015) ZINC 15-ligand discovery for everyone. *J Chem Inf Model*. <https://doi.org/10.1021/acs.jcim.5b00559>
25. Lindert S, Durrant JD, McCammon JA (2012) LigMerge: a fast algorithm to generate models of novel potential ligands from sets of known binders. *Chem Biol Drug Des* 80(3):358–365. <https://doi.org/10.1111/j.1747-0285.2012.01414.x>
26. Landrum G RDKIT: open-source cheminformatics. <http://www.rdkit.org/>
27. Ropp PJ, Kaminsky JC, Yablonski S, Durrant JD (2019) Dimorphite-DL: an open-source program for enumerating the ionization states of drug-like small molecules. *J Cheminform* 11(1):14. <https://doi.org/10.1186/s13321-019-0336-9>
28. Durrant JD, McCammon JA (2010) NNScore: a neural-network-based scoring function for the characterization of protein–ligand complexes. *J Chem Inf Model* 50(10):1865–1871. <https://doi.org/10.1021/ci100244v>
29. Durrant JD, McCammon JA (2011) Nnscore 2.0: a neural-network receptor-ligand scoring function. *J Chem Inf Model* 51(11):2897–2903. <https://doi.org/10.1021/ci2003889>
30. Reynolds CH, Tounge BA, Bembek SD (2008) Ligand binding efficiency: trends, physical basis, and implications. *J Med Chem* 51(8):2432–2438
31. Landrum G (2011) Getting started with the RDKit in Python. <https://www.rdkit.org/docs/GettingStartedInPython.html>
32. Bajusz D, Rácz A, Héberger K (2015) Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *J Cheminform*. <https://doi.org/10.1186/s13321-015-0069-3>
33. Shukla A, Pandey HM, Mehrotra D (2015) Comparative review of selection techniques in genetic algorithm. In: 2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE) (February), pp 515–519. <https://doi.org/10.1109/ABLAZE.2015.7154916>
34. Berman HM, Battistuz T, Bhat TN, Bluhm WF, Bourne PE, Burkhardt K, Feng Z, Gilliland GL, Iype L, Jain S, Fagan P, Marvin J, Padilla D, Ravichandran V, Schneider B, Thanki N, Weissig H, Westbrook JD, Zardocki C (2002) The protein data bank. *Acta Crystallogr Sect D Biol Crystallogr*. <https://doi.org/10.1107/S0907444902003451>
35. Dolinsky TJ, Nielsen JE, McCammon JA, Baker NA (2004) Pdb2pqr: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucl Acids Res* 32(Web Server issue):665–667. <https://doi.org/10.1093/nar/gkh381>
36. Dolinsky TJ, Czodrowski P, Li H, Nielsen JE, Jensen JH, Klebe G, Baker NA (2007) Pdb2pqr: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucl Acids Res* 35(Web Server issue):522–525. <https://doi.org/10.1093/nar/gkm276>
37. O’Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR (2011) Open babel: an open chemical toolbox. *J Cheminform*. <https://doi.org/10.1186/1758-2946-3-33>
38. Ropp P, Friedman A, Durrant JD (2017) Scoria: a Python module for manipulating 3D molecular data. *J Cheminform*. <https://doi.org/10.1186/s13321-017-0237-8>
39. Durrant JD (2018) Blendmol: advanced macromolecular visualization in blender. *Bioinformatics* 35(13):2323–2325. <https://doi.org/10.1093/bioinformatics/bty968>
40. Alvesalo JK, Siiskonen A, Vainio MJ, Tammela PS, Vuorela PM (2006) Similarity based virtual screening: a tool for targeted library design. *J Med Chem* 49(7):2353–2356
41. de Souza Neto LR, Moreira-Filho JT, Neves BJ, Maidana RLBR, Guimarães ACR, Furnham N, Andrade CH, Silva FP Jr (2020) In silico strategies to support fragment-to-lead optimization in drug discovery. *Front Chem* 8:93
42. Chen X, Lin Y, Liu M, Gilson MK (2002) The binding database: data management and interface design. *Bioinformatics* 18(1):130–139. <https://doi.org/10.1093/bioinformatics/18.1.130>
43. Liu T, Lin Y, Wen X, Jorissen RN, Gilson MK (2007) BindingDB: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucl Acids Res* 35(Database issue):198–201. <https://doi.org/10.1093/nar/gkl999>
44. Butina D (1999) Unsupervised data base clustering based on daylight’s fingerprint and tanimoto similarity: a fast and automated way to cluster small and large data sets. *J Chem Inf Comput Sci* 39(4):747–750
45. Wang Y, Xiao J, Suzek TO, Zhang J, Wang J, Bryant SH (2009) Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucl Acids Res* 37(Web Server issue):623–33. <https://doi.org/10.1093/nar/gkp456>
46. Van Houten B, Santa-Gonzalez GA, Camargo M (2018) DNA repair after oxidative stress: current challenges. *Curr Opin Toxicol* 7:9–16. <https://doi.org/10.1016/j.cotox.2017.10.009>
47. Hegde ML, Hazra TK, Mitra S (2008) Early steps in the dna base excision/single-strand interruption repair pathway in mammalian cells. *Cell Res* 18(1):27
48. Hoeijmakers JH (2001) Genome maintenance mechanisms for preventing cancer. *Nature* 411(6835):366
49. Chang HH, Pannunzio NR, Adachi N, Lieber MR (2017) Non-homologous DNA end joining and alternative pathways to double-strand break repair. *Nat Rev Mol Cell Biol* 18(8):495
50. Couto CA-M, Wang H-Y, Green JCA, Kiely R, Siddaway R, Borer C, Pears CJ, Lakin ND (2011) PARP regulates nonhomologous end joining through retention of Ku at double-strand breaks. *J Cell Biol* 194(3):367–375. <https://doi.org/10.1083/jcb.201012132>

51. Mark M, Wendling O, Wynshaw-boris JMEA (2001) Early embryonic lethality in PARP-1 Atm double-mutant mice suggests a functional synergy in cell proliferation during development. *Microbiology* 21(5):1828–1832. <https://doi.org/10.1128/MCB.21.5.1828>
52. Ira G, Pelliccioli A, Balijja A, Wang X, Florani S, Carotenuto W, Liberi G, Bressan D, Wan L, Hollingsworth NM, Haber JE, Folani M (2004) DNA end resection, homologous recombination and DNA damage checkpoint activation require CDK1. *Nature* 431(7011):1011–1017. <https://doi.org/10.1038/nature02964>
53. Alemasova EE, Lavrik OI (2019) Poly (ADP-ribose) ation by PARP1: reaction mechanism and regulatory proteins. *Nucl Acids Res* 47(8):3811–3827
54. Ghose AK, Viswanadhan VN, Wendoloski JJ (1999) A knowledge-based approach in designing combinatorial or medicinal chemistry libraries for drug discovery. *J Comb Chem* 1(1):55–68. <https://doi.org/10.1021/cc9800071>
55. Chang MW, Ayeni C, Breuer S, Torbett BE (2010) Virtual screening for HIV protease inhibitors: a comparison of AutoDock 4 and Vina. *PLoS ONE*. <https://doi.org/10.1371/journal.pone.0011955>
56. Brenk R, Schipani A, James D, Krasowski A, Gilbert IH, Frearson J, Wyatt PG (2008) Lessons learnt from assembling screening libraries for drug discovery for neglected diseases. *ChemMedChem* 3(3):435–444. <https://doi.org/10.1002/cmdc.200700139>
57. Casas N (2015) Genetic algorithms for multimodal optimization: a review
58. Hattori K, Kido Y, Yamamoto H, Ishida J, Kamijo K, Murano K, Ohkubo M, Kinoshita T, Iwashita A, Mihara K, Yamazaki S, Matsuoka N, Teramura Y, Miyake H (2004) Rational approaches to discovery of orally active and brain-penetrable quinazolinone inhibitors of poly(ADP-ribose)polymerase. *J Med Chem* 47(17):4151–4. <https://doi.org/10.1021/jm0499256>
59. Ye N, Chen C-H, Chen T, Song Z, He J-X, Huan X-J, Song S-S, Liu Q, Chen Y, Ding J, Xu Y, Miao Z-H, Zhang A (2013) Design, synthesis, and biological evaluation of a series of benzo[de][1,7]naphthyridin-7(8h)-ones bearing a functionalized longer chain appendage as novel parp1 inhibitors. *J Med Chem* 56(7):2885–903. <https://doi.org/10.1021/jm301825t>
60. Lindgren AEG, Karlberg T, Thorsell A-G, Hesse M, Spjut S, Ekblad T, Andersson CD, Pinto AF, Weigelt J, Hottiger MO, Linusson A, Eloffsson M, Schüler H (2013) PARP inhibitor with selectivity toward ADP-ribosyltransferase ARTD3/PARP3. *ACS Chem Biol* 8(8):1698–703. <https://doi.org/10.1021/cb4002014>
61. Patel MR, Bhatt A, Steffen JD, Chergui A, Murai J, Pommier Y, Pascal JM, Trombetta LD, Fronczek FR, Talele TT (2014) Discovery and structure-activity relationship of novel 2,3-dihydrobenzofuran-7-carboxamide and 2,3-dihydrobenzofuran-3(2h)-one-7-carboxamide derivatives as poly(adp-ribose)polymerase-1 inhibitors. *J Med Chem* 57(13):5579–601. <https://doi.org/10.1021/jm5002502>
62. Aoyagi-Scharber M, Gardberg AS, Yip BK, Wang B, Shen Y, Fitzpatrick PA (2014) Structural basis for the inhibition of poly(adp-ribose) polymerases 1 and 2 by bmn 673, a potent inhibitor derived from dihydropyridophthalazinone. *Acta Crystallogr F Struct Biol Commun* 70(Pt 9):1143–9. <https://doi.org/10.1107/S2053230X14015088>
63. Papeo G, Posteri H, Borghi D, Busel AA, Caprera F, Casale E, Ciomei M, Cirila A, Corti E, D'Anello M, Fasolini M, Forte B, Galvani A, Isacchi A, Khvat A, Krasavin MY, Lupi R, Orsini P, Perego R, Pesenti E, Pezzetta D, Rainoldi S, Riccardi-Sirtori F, Scolaro A, Sola F, Zuccotto F, Felder ER, Donati D, Montagnoli A (2015) Discovery of 2-[1-(4,4-difluorocyclohexyl)piperidin-4-yl]-6-fluoro-3-oxo-2,3-dihydro-1h-isoindole-4-carboxamide (nms-p118): a potent, orally available, and highly selective PARP-1 inhibitor for cancer therapy. *J Med Chem* 58(17):6875–98. <https://doi.org/10.1021/acs.jmedchem.5b00680>
64. Fu L, Wang S, Wang X, Wang P, Zheng Y, Yao D, Guo M, Zhang L, Ouyang L (2016) Crystal structure-based discovery of a novel synthesized parp1 inhibitor (ol-1) with apoptosis-inducing mechanisms in triple-negative breast cancer. *Sci Rep* 6(1):3. <https://doi.org/10.1038/s41598-016-0007-2>
65. Chen X, Huan X, Liu Q, Wang Y, He Q, Tan C, Chen Y, Ding J, Xu Y, Miao Z, Yang C (2018) Design and synthesis of 2-(4,5,6,7-tetrahydrohienopyridin-2-yl)-benzimidazole carboxamides as novel orally efficacious poly(ADP-ribose)polymerase (PARP) inhibitors. *Eur J Med Chem* 145:389–403. <https://doi.org/10.1016/j.ejmech.2018.01.018>
66. Velagapudi UK, Langelier M-F, Delgado-Martin C, Diolaiti ME, Bakker S, Ashworth A, Patel BA, Shao X, Pascal JM, Talele TT (2019) Design and synthesis of poly (ADP-ribose) polymerase inhibitors: impact of adenosine pocket-binding motif appendage to the 3-oxo-2, 3-dihydrobenzofuran-7-carboxamide on potency and selectivity. *J Med Chem* 62:5330–5357
67. Wahlberg E, Karlberg T, Kouznetsova E, Markova N, Macchiarulo A, Thorsell A-G, Pol E, Frostell Å, Ekblad T, Öncü D et al (2012) Family-wide chemical profiling and structural analysis of PARP and tankyrase inhibitors. *Nat Biotechnol* 30(3):283
68. Upton K, Meyers M, Thorsell A-G, Karlberg T, Holeček J, Lease R, Schey G, Wolf E, Lucente A, Schüler H et al (2017) Design and synthesis of potent inhibitors of the mono (ADP-ribose)l transferase, PARP14. *Bioorg Med Chem Lett* 27(13):2907–2911
69. Jagtap PG, Southan GJ, Baloglu E, Ram S, Mabley JG, Marton A, Salzman A, Szabo C (2004) The discovery and synthesis of novel adenosine substituted 2, 3-dihydro-1h-isoindol-1-ones: potent inhibitors of poly (ADP-ribose) polymerase-1 (PARP-1). *Bioorg Med Chem Lett* 14(1):81–85
70. Jagtap P, Szabó C (2005) Poly (ADP-ribose) polymerase and the therapeutic effects of its inhibitors. *Nat Rev Drug Discov* 4(5):421
71. Tanimoto T (1957) IBM technical report series. Report
72. Lipinski CA, Lombardo F, Dominy BW, Feeney PJ (1997) Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv Drug Deliv Rev* 23(1–3):3–25. [https://doi.org/10.1016/S0169-409x\(96\)00423-1](https://doi.org/10.1016/S0169-409x(96)00423-1)
73. Durrant JD, McCammon JA (2011) BINANA: a novel algorithm for ligand-binding characterization. *J Mol Graph Model* 29(6):888–893. <https://doi.org/10.1016/j.jmgm.2011.01.004>
74. Du Y, Yamaguchi H, Wei Y, Hsu JL, Wang HL, Hsu YH, Lin WC, Yu WH, Leonard PG, Lee GR, Chen MK, Nakai K, Hsu MC, Chen CT, Sun Y, Wu Y, Chang WC, Huang WC, Liu CL, Chang YC, Chen CH, Park M, Jones P, Hortobagyi GN, Hung MC (2016) Blocking c-Met-mediated PARP1 phosphorylation enhances anti-tumor effects of PARP inhibitors. *Nat Med*. <https://doi.org/10.1038/nm.4032>
75. Wang R, Gao Y, Lai L (2000) Ligbuilder: a multi-purpose program for structure-based drug design. *J Mol Model* 6(7):498
76. Yuan Y, Pei J, Lai L (2011) Ligbuilder 2: a practical de novo drug design approach. *J Chem Inform Model* 51(5):1083–1091
77. Yuan Y, Pei J, Lai L (2020) Ligbuilder v3: a multi-target de novo drug design approach. *Front Chem* 8:142
78. Li Y, Zhao Y, Liu Z, Wang R (2011) Automatic tailoring and transplanting: a practical method that makes virtual screening more useful. *ACS Publications*, Washington
79. Li Y, Zhao Z, Liu Z, Su M, Wang R (2016) Autot&t v.2: an efficient and versatile tool for lead structure generation and optimization. *J Chem Inform Model* 56(2):435–453
80. Pearce BC, Langley DR, Kang J, Huang H, Kulkarni A (2009) E-novo: an automated workflow for efficient structure-based lead optimization. *J Chem Inform Model* 49(7):1797–1809
81. Fechner U, Schneider G (2006) Flux (1): a virtual synthesis scheme for fragment-based de novo design. *J Chem Inform Model* 46(2):699–707
82. Schürer SC, Tyagi P, Muskal SM (2005) Prospective exploration of synthetically feasible, medicinally relevant chemical space. *J Chem Inform Model* 45(2):239–248
83. Moore JW (2005) Maximizing discovery efficiency with a computationally driven fragment approach. *Curr Opin Drug Disco Dev* 8(3):355–364
84. Brown N, McKay B, Gilardoni F, Gasteiger J (2004) A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. *J Chem Inform Comput Sci* 44(3):1079–1087
85. Pierce AC, Rao G, Bemis GW (2004) Breed: generating novel inhibitors through hybridization of known ligands. application to CDK2, p38, and HIV protease. *J Med Chem* 47(11):2768–2775
86. Stultz CM, Karplus M (2000) Dynamic ligand design and combinatorial optimization: designing inhibitors to endothiapepsin. *Proteins Struct Funct Bioinform* 40(2):258–289
87. Pearlman DA, Murcko MA (1996) Concerts: dynamic connection of fragments as an approach to de novo ligand design. *J Med Chem* 39(8):1651–1663
88. Clark DE, Frenkel D, Levy SA, Li J, Murray CW, Robson B, Waszkowycz B, Westhead DR (1995) Pro ligand: an approach to de novo molecular

- design. 1. Application to the design of organic molecules. *J Comput Aided Mol Des* 9(1):13–32
89. Roe DC, Kuntz ID (1995) Builder v.2: improving the chemistry of a de novo design strategy. *J Comput Aided Mol Des* 9(3):269–282
90. Bohm H-J (1992) The computer program ludi: a new method for the de novo design of enzyme inhibitors. *J Comput Aided Mol Des* 6(1):61
91. Bohm HJ (1992) LUDI: rule-based automatic design of new substituents for enzyme inhibitor leads. *J Comput Aided Mol Des* 6(6):593–606
92. Chu He (2019) MoleGear: a java-based platform for evolutionary de novo molecular design. *Molecules* 24(7):1444. <https://doi.org/10.3390/molecules24071444>
93. Morris GM, Ruth H, Lindstrom W, Sanner MF, Belew RK, Goodsell DS, Olson AJ (2009) AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. *J Comput Chem*. <https://doi.org/10.1002/jcc.21256>
94. Allen WJ, Fochtman BC, Balias TE, Rizzo RC (2017) Customizable de novo design strategies for DOCK: application to HIVgp41 and other therapeutic targets. *J Comput Chem* 38(30):2641–2663. <https://doi.org/10.1002/jcc.25052>
95. Allen WJ, Balias TE, Mukherjee S, Brozell SR, Moustakas DT, Lang PT, Case DA, Kuntz ID, Rizzo RC (2015) DOCK 6: impact of new features and current docking performance. *J Comput Chem* 36(15):1132–1156. <https://doi.org/10.1002/jcc.23905>
96. Dey F, Caflich A (2008) Fragment-based de novo ligand design by multi-objective evolutionary optimization. Supporting Information. *J Chem Inf Model* 48(3):679–690
97. Kolb P, Caflich A (2006) Automatic and efficient decomposition of two-dimensional structures of small molecules for fragment-based high-throughput docking. *J Med Chem* 49(25):7384–7392
98. Majeux N, Scarsi M, Apostolakis J, Ehrhardt C, Caflich A (1999) Exhaustive docking of molecular fragments with electrostatic solvation. *Proteins Struct Funct Bioinform* 37(1):88–105
99. Budin N, Majeux N, Caflich A (2001) Fragment-based flexible ligand docking by evolutionary optimization. *Biol Chem* 382(9):1365–1372
100. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. *J Cheminform* 9(1):48. <https://doi.org/10.1186/s13321-017-0235-x>
101. Skalic M, Jiménez Luna J, Sabbadin D, De Fabritiis G (2019) Shape-based generative modeling for de-novo drug design. *J Chem Inf Model*. <https://doi.org/10.1021/acs.jcim.8b00263>
102. Ascher D, Dubois PF, Hinsen K, James JH, Oliphant T (1999) Numerical python, UCRL-MA-128569 edn. Lawrence Livermore National Laboratory, Livermore
103. Oliphant TE (2006) Guide to NumPy. Brigham Young University, Provo
104. Jones E, Oliphant T, Peterson P (2001) Others: SciPy: Open source scientific tools for python. <http://www.scipy.org>
105. Hunter JD (2007) Matplotlib: a 2D graphics environment. *Comput Sci Eng* 9(3):90
106. Dalcin L, Paz R, Storti M, D'Elia J (2008) MPI for python: performance improvements and MPI-2 extensions. *J Parallel Distrib Comput* 68(5):655–662. <https://doi.org/10.1016/j.jpdc.2007.09.005>
107. Van De Waterbeemd H, Camenisch G, Folkers G, Chretien JR, Raevsky OA (1998) Estimation of blood–brain barrier crossing of drugs using molecular size and shape, and H-bonding descriptors. *J Drug Target* 6(2):151–165. <https://doi.org/10.1177/0004563215595431>
108. Mozziconacci J, Arnoult E, Baurin N, Marot C, Morin-Allory L (2003) Preparation of a molecular database from a set of 2 million compounds for virtual screening applications: gathering, structural analysis and filtering. In: 9th electronic computational chemistry conference, World Wide Web
109. Jadhav A, Ferreira RS, Klumpp C, Mott BT, Austin CP, Inglese J, Thomas CJ, Maloney DJ, Shoichet BK, Simeonov A (2010) Quantitative analyses of aggregation, autofluorescence, and reactivity artifacts in a screen for inhibitors of a thiol protease. *J Med Chem* 53(1):37–51. <https://doi.org/10.1021/jm901070c>
110. Doveston RG, Tosatti P, Dow M, Foley DJ, Li HY, Campbell AJ, House D, Churcher I, Marsden SP, Nelson A (2015) A unified lead-oriented synthesis of over fifty molecular scaffolds. *Org Biomol Chem* 13(3):859–865. <https://doi.org/10.1039/c4ob02287d>
111. Baell JB, Holloway GA (2010) New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J Med Chem* 53(7):2719–2740. <https://doi.org/10.1021/jm901137j>

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions

