Supplementary Information
Parallel Tempered Genetic Algorithm Guided by Deep Neural Networks for Inverse Molecular Design

AkshatKumar Nigam,[1,2,3,6] Robert Pollice,[2,3,6] and Alán Aspuru-Guzik[2,3,4,5,*]

[1] *Department of Computer Science, Stanford University, USA.*
[2] *Department of Computer Science, University of Toronto, Canada.*
[3] *Department of Chemistry, University of Toronto, Canada.*
[4] *Vector Institute for Artificial Intelligence, Toronto, Canada.*
[5] *Lebovic Fellow, Canadian Institute for Advanced Research (CIFAR), 661 University Ave, Toronto, Ontario M5G, Canada.*
[6] *Equal Contributions.*

S1. EXTENDED BACKGROUND

Variational autoencoders (VAEs), generative adversarial networks (GANs), Markov decision processes (MDPs) and reinforcement learning (RL) approaches belong to the group of deep generative models as they all rely on DNNs for molecular generation. VAEs convert discrete representations into continuous latent spaces and vice versa. This latent space allows for both gradient-based optimization and vicinity-based sampling for generating favorable molecules [1]. Since their first demonstration in chemistry [2], they have received widespread attention, notably being applied to the design of nanoporous crystalline reticular materials [3], the optimization of binding affinities for drug discovery [4], the exploration of inorganic materials [5] and scaffold-based drug design [6]. Common implementations of VAEs for inverse molecular design include CVAE [2], GVAE [7] and SD-VAE [8], which operate on molecular string representations. Alternatively, JT-VAE [9], CGVAE [10] and DEFactor [11] use matrix representations of molecular graphs. The framework of VAEs can be generalized to encoder-decoder architectures where the output of the decoder does not correspond to the input of the encoder. This has been used in translation tasks for performing structural modifications. VJTNN [12] follows this approach by implementing a graph-to-graph translation.

GANs are characterized by joint adversarial training of a generator and a discriminator DNN. The generator proposes molecular structures from a high-dimensional latent space, while the discriminator is tasked with distinguishing the proposed structures from a reference dataset. The generator and discriminator are trained as competing networks. For inverse design, ORGAN [13] and ORGANIC [14] were the first implementations of a GAN for molecular design that were trained on molecular string representations with RL. Follow up work includes the use of adjacency matrices [15, 16]. RL approaches in general rely on learning how to take actions in an environment to maximize a cumulative reward. Accordingly, the goal is to use DNNs to predict the best action based on a given state. GCPN [17] and PGFS [18] use matrix representations of molecular graphs, while MolDQN [19] and REINVENT [20] use string-based representations of molecular graphs. GCPN relies on an MDP for generating new molecules, starting from an initial set of structures. MolDQN utilizes an action space of elementary modifications chosen by domain experts. Similarly, PGFS also implements an expert-derived action space but in the form of reaction templates to mimic forward synthesis for molecular generation. In contrast, REINVENT and MRNN [21] use recurrent neural networks (RNNs) [22] to propose new molecules.

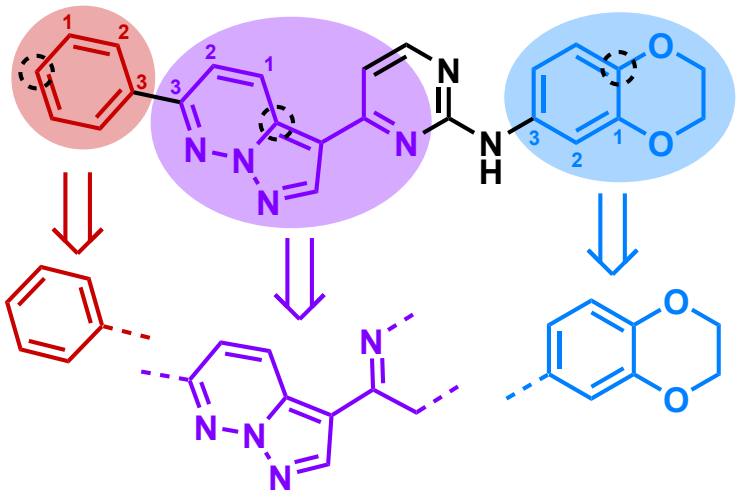
MDPs require explicit actions from a decision maker and knowledge of the current state to predict future states. Monte Carlo tree search (MCTS) is a method to sample decision processes such as MDPs that has gained significant popularity in ML due to its use in AlphaGo in combination with DNNs [23]. This also inspired its application for inverse molecular design. Particular implementations include ChemTS [24] that relies on string-based representations of molecular graphs and an RNN for modeling the MCTS. Alternative approaches are unitMCTS [25] and MARS [26], which both rely on matrix representations of molecular graphs. While unitMCTS implements MCTS for modeling the molecular generation process, MARS uses annealed Markov chain Monte Carlo sampling. Finally, Flow-based generative models (FGMs) are based on normalized flow to model probability distributions by using a sequence of invertible functions acting on an input to explicitly model the log-likelihood. They can be used as an alternative to

* Correspondence to: alan@aspuru.com

VAEs to generate continuous latent spaces with the advantage that only one direction of the transformation needs to be trained due to its inherent invertibility. One model relying on FGMs is GraphAF [27], which combines the advantages of FGMs with the ones of autoregressive models and uses an RL framework for optimization of molecular properties.

S2. EXTRACTING FRAGMENTS FROM DATASETS

In JANUS, fragments can be extracted in an automated manner based on molecular subgraphs within a given radius of a central atom indicating the distance in number of edges from that atom. They are based on a reference dataset and can be used as optional method to bias the genetic operators. The corresponding methodology is illustrated in Supplementary Figure 1. The molecule depicted is taken from the training datasets of the imitated inhibition tasks.



Supplementary Figure 1. Schematic illustration of the extraction of circular fragments from molecules with a radius of 3 from the circled atoms.

S3. SELECTION OF POPULATION MEMBERS FOR GENETIC OPERATORS

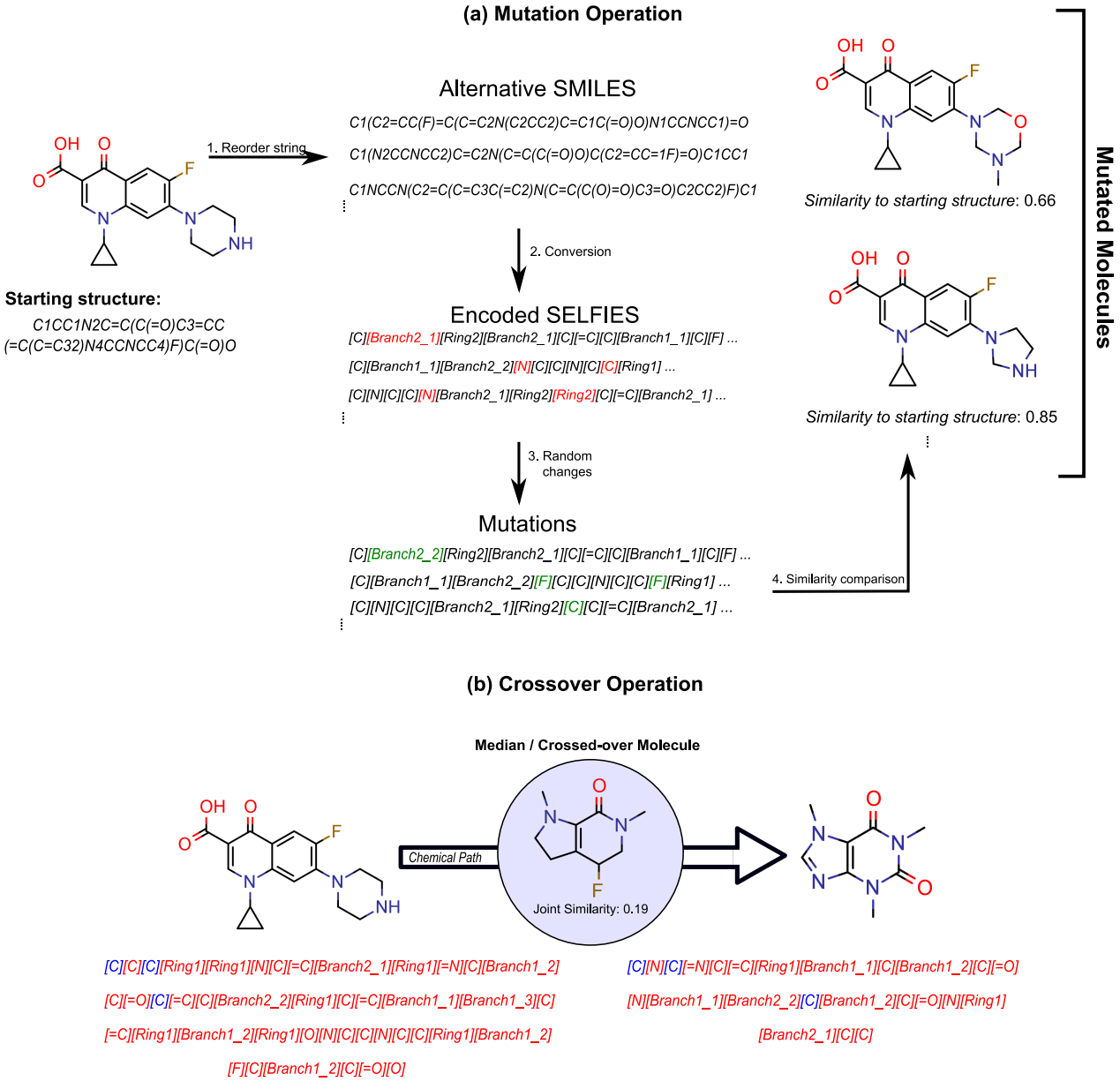
The molecules that are selected from the exploration population for proceeding to the genetic operators are not solely chosen based on fitness but also based on a parameter F_{25} that essentially determines the temperature of the population. At higher temperature, molecules of lower fitness are more likely to proceed to the next generation. Inspired by Fermi-Dirac statistics [28, 29], we evaluate the relative frequency p_i for molecule i to be selected according to the following formula:

$$p_i = \left(3^{\frac{F_{50}-F_i}{F_{50}-F_{25}}} + 1\right)^{-1} \tag{1}$$

In that equation, F_{50} corresponds to the fitness of the n^{th} molecule when the population is sorted by decreasing fitness and n molecules need to be selected in total. This fitness corresponds to a relative frequency of 0.50 for being selected. The parameter F_{25} is the fitness value that is to be assigned a relative frequency of 0.25 for being selected. The lower it is, the higher the effective temperature of the system and the more likely molecules of very low fitness are propagated to the next generation increasing the extent of exploration. Unless noted otherwise, the fitness of the $(n+1)^{th}$ molecule is chosen for F_{25} in each generation. The impact of the F_{25} parameter on the performance of JANUS was tested in the unconstrained optimization of the penalized log P and the corresponding details can be found in Section S9.

S4. GENETIC OPERATORS IN JANUS

Supplementary Figure 2 illustrates the algorithms used as genetic operators in JANUS. They are based on the STONED algorithm, a set of string modifications of SELFIES that have recently been shown to be extremely efficient for molecular generation [30].



Supplementary Figure 2. Overview of the mutation and crossover operations implemented in JANUS. (a) A combination of SMILES reordering and random SELFIES modification yields a diverse set of mutated molecules. (b) Chemical paths between two SELFIES are generated by gradually changing the SELFIES characters of the starting molecule to the corresponding SELFIES characters at the exact same position of the target molecule. The order these characters are changed is random providing a large number of possible paths between the two molecules. The molecule with highest joint similarity with respect to both starting and target molecules is selected as child structure between the two.

S5. NEURAL NETWORK FEATURES

For training of neural network models, i.e, the classifier or the property predictor, in JANUS, we use 51 descriptors collected from RDKit as input features. They are described in more detail in the following list.

- The ratio of the number of carbon atoms to the total number of atoms within a molecule (1 feature)
- The ratio of the number of hydrogen atoms to the total number of atoms within a molecule (1 feature)
- The ratio of the number of nitrogen atoms to the total number of atoms within a molecule (1 feature)
- The ratio of the number of sulfur atoms to the total number of atoms within a molecule (1 feature)
- The ratio of the number of oxygen atoms to the total number of atoms within a molecule (1 feature)
- The ratio of the number of chlorine atoms to the total number of atoms within a molecule (1 feature)
- The ratio of the number of bromine atoms to the total number of atoms within a molecule (1 feature)
- The ratio of the number of fluorine atoms to the total number of atoms within a molecule (1 feature)
- A set of RDKit descriptors, titled: "RingCount", "HallKierAlpha", "BalabanJ", "NumAliphaticCarbocycles","NumAliphaticHeterocycles", "NumAliphaticRings","NumAromaticCarbocycles","NumAromaticHeterocycles", "NumAromaticRings","NumHAcceptors","NumHDonors","NumHeteroatoms", "NumRadicalElectrons", "NumSaturatedCarbocycles","NumSaturatedHeterocycles", "NumSaturatedRings","NumValenceElectrons" (17 features in total)
- The total number of single, double, triple and aromatic bonds within a molecule (4 features)
- Total number of rings in a molecule (1 feature)
- The number of rings ranging from size 3 to 20 within a molecule and the total number of consecutive double bonds within rings (19 features)
- The number of triple bonds within rings (1 feature)
- The number of consecutive double bonds within a molecule (1 feature)

S6. FEATURE COMPARISON OF JANUS AND GA+D

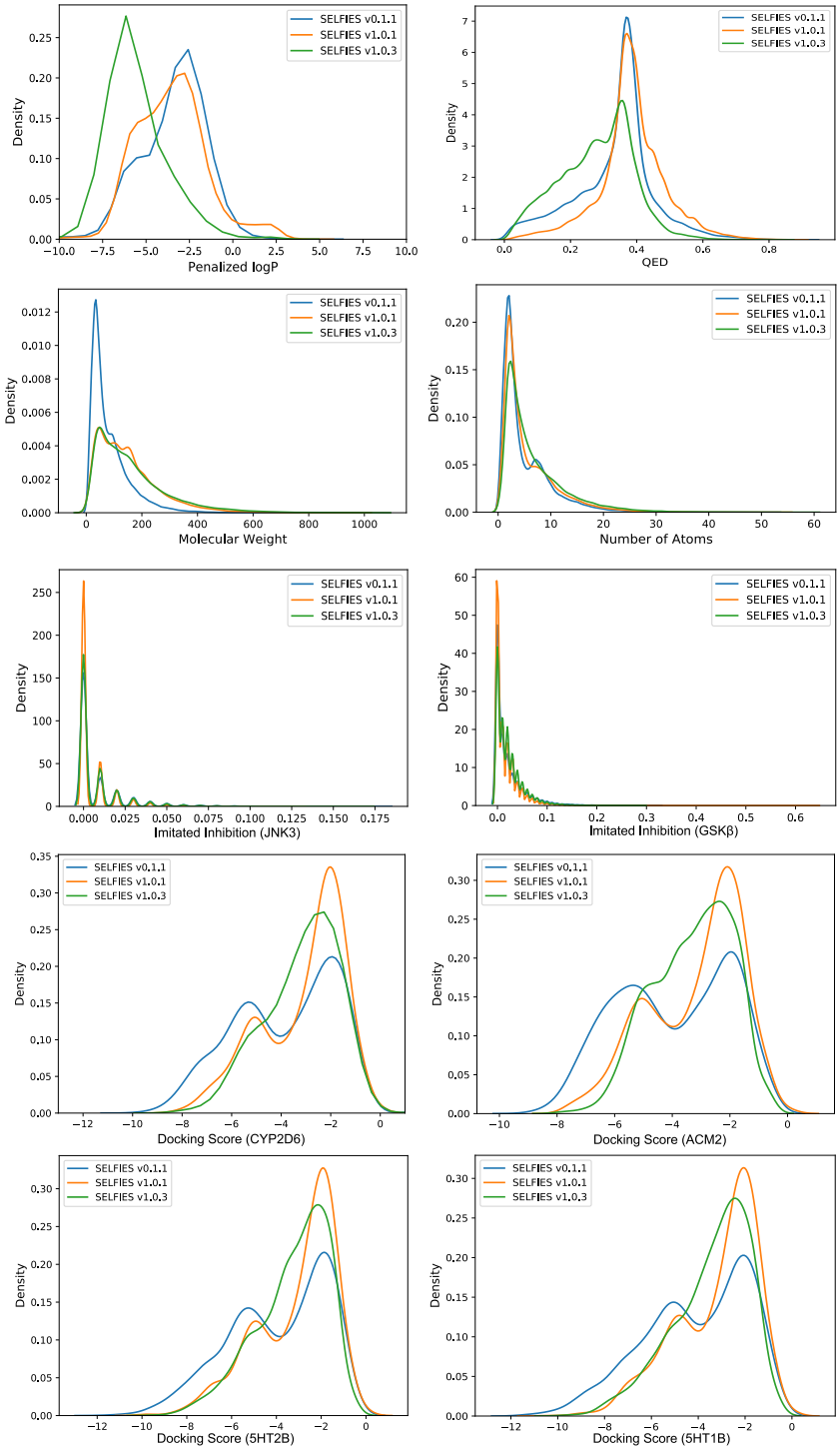
Supplementary Table 1 compares the features of GA+D [31], a previously published genetic algorithm based on SELFIES [32], and JANUS. It demonstrates several advances of JANUS over GA+D but also shows that the current version of JANUS does not use fitness augmentation via a discriminator.

Supplementary Table 1. Feature comparison of JANUS and GA+D [31], two GAs relying on SELFIES [32]. ✓ and ✗ indicate the presence and absence of a feature, respectively. ○ indicates the feature to be implemented but optional.

Feature	**GA+D [31]**	**JANUS**
Mutation	✓	✓
Crossover	✗	✓
Similarity Selection Pressure	✗	✓
Neural Network Selection Pressure	✗	○
Parallel Populations	✗	✓
Fragment Extraction	✗	○
Discriminator Fitness Augmentation	○	✗

S7. PROPERTY DISTRIBUTIONS OF RANDOM SELFIES

The properties of molecules generated from random SELFIES depend on the specific version of the alphabet used. The distribution densities of various properties used in the course of this work for various version of SELFIES are illustrated in Supplementary Figure 3. Importantly, this does not only affect completely randomly generated structures but also randomly modified structures generated via point mutations.



Supplementary Figure 3. Property distribution densities of molecules obtained from random SELFIES using various versions.

S8. ADJUSTABLE PARAMETERS IN JANUS

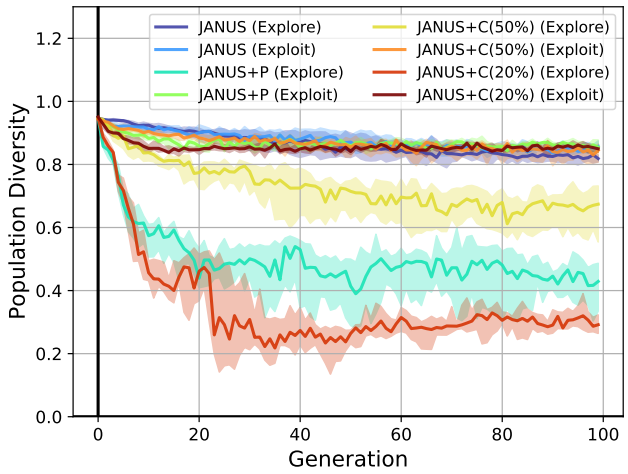
Supplementary Table 2 provides an overview and an explanation of all the parameters that a user can readily define for every molecular design run with JANUS.

Supplementary Table 2. Description of parameters that users can specify in the file *params_init.py* prior to runnning JANUS (`https://github.com/aspuru-guzik-group/JANUS/blob/main/params_init.py`.)

Parameter	**Description**
$params\_['generations']$	Number of generations (iterations) to run JANUS.
$params\_['generation\_size']$	The number of molecules that are kept within a generation. It also corresponds to the number of fitness calculations done for the exploration and exploitation components of JANUS, respectively.
$params\_['start\_population']$	The location of the text file containing SMILES that are used to seed JANUS for Generation 0.
$params\_['num\_exchanges']$	Number of molecules that are exchanged between the exploration and exploitation components of JANUS.
$params\_['use\_fragments']$	An option to generate fragments and use them when performing mutations. Fragments are generated using the SMILES provided for the starting population. The list of generated fragments is stored in the file './DATA/fragments_selfies.txt'.
$params\_['use\_NN\_classifier']$	An option to use a classifier for sampling. If set to true, the trained model is saved at the end of every generation in './RESULTS/'.
$calc\_prop(smi)$	Given a SMILES string (smi), a user can specify an algorithm within this function for calculating the fitness value.

S9. UNCONSTRAINED PENALIZED LOG P OPTIMIZATION

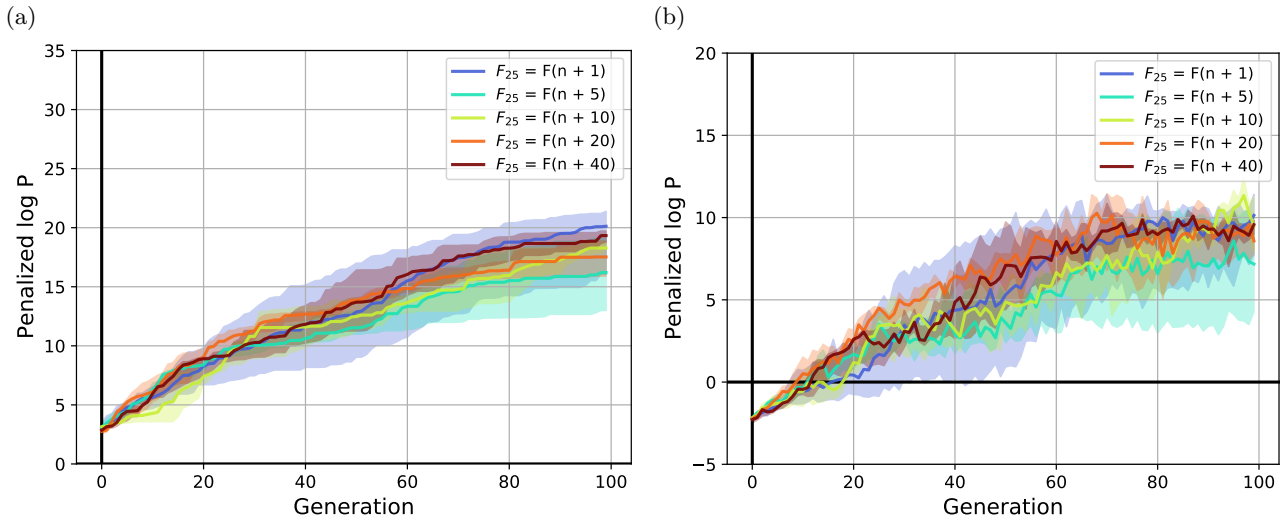
In addition to the results discussed in the main text, we also investigated the generational propagation of diversities of both the explorative and the exploitative populations in the maximization of the unconstrained penalized logarithm of the octanol-water partition coefficient scores. The corresponding results using four different kinds of selection pressure are depicted in Supplementary Figure 4. First, we observe that the diversities in the explorative populations are never higher than in the exploitative populations. With additional selection pressure in the explorative populations from neural networks, the diversities are significantly lower compared to the exploitative populations. This is particularly pronounced when additional selection pressure is applied using a property predictor or a classifier that only accepts the best 20% of molecules as both of them only accept a narrow range of molecules. As the optimizations progress, the diversities tend to decrease which suggests that the search space is at least partially focused on the best-performing molecules. However, it only decreases marginally for the exploitative populations. Nevertheless, at least in the experiments investigated here, the diversities remain high in both the explorative and the exploitative populations as long as the additional selection pressure allows for some leeway. However, the results also suggest that JANUS could still be improved further by increasing the diversities in the explorative population. At the same time, the diversities in the exploitative population could probably be decreased leading to more efficient local searches.



Supplementary Figure 4. Progress of the median-of-median diversities in each generation across 15 independent runs for both the explorative and the exploitative populations of JANUS with four variations of selection pressure in the maximization of the penalized logarithm of the octanol-water partition coefficient (penalized log P). The semi-transparent areas depict the diversity intervals between the corresponding first and third quartiles of each generation.

In addition to the experiments discussed in the main text, we also tested the impact of the F_{25} parameter on the performance of JANUS in the unconstrained optimization of the penalized log P. The corresponding results are depicted in Supplementary Figure 5. Notably, the parameter F_{25} is the fitness value in a given generation that is to be assigned a relative frequency of 0.25 for being selected. In this experiment, F_{50} corresponds to the fitness of the 50^{th} molecule when the exploration population of size 250 is sorted by decreasing fitness, as 50 molecules need to be selected in total. The default value of F_{25} is chosen to be the fitness of the $(\text{n}+1)^{\text{th}}$ molecule in each generation, which is referred to here as F(n+1). Effectively, this is a very low "temperature" of the population as essentially only the most fit molecules will be selected for genetic operations. The additional values of the F_{25} parameter tested are the fitnesses of the $(\text{n}+5)^{\text{th}}$, $(\text{n}+10)^{\text{th}}$, $(\text{n}+20)^{\text{th}}$ and $(\text{n}+40)^{\text{th}}$ molecule, respectively, and they are referred to as F(n+5), F(n+10), F(n+20) and F(n+40) in Supplementary Figure 5. The corresponding results demonstrate that, at least in the parameter range tested, F_{25} does not seem to have any significant influence. Hence, we decided to not investigate its influence on the performance of JANUS any further and used the default setting for F_{25} in all subsequent experiments.

Furthermore, we performed an ablation experiment to test the impact of the use of crossover in the genetic operators. The comparison of the performance of JANUS with and without crossover on the unconstrained penalized log P optimization task is illustrated in Supplementary Figure 6. The results demonstrate that the use of crossover in the genetic operators has a very significant influence on the performance of JANUS. The impact is particularly pronounced in the median-of-medians fitness in each generation of the exploration population (cf. Supplementary Figure 6(b)).

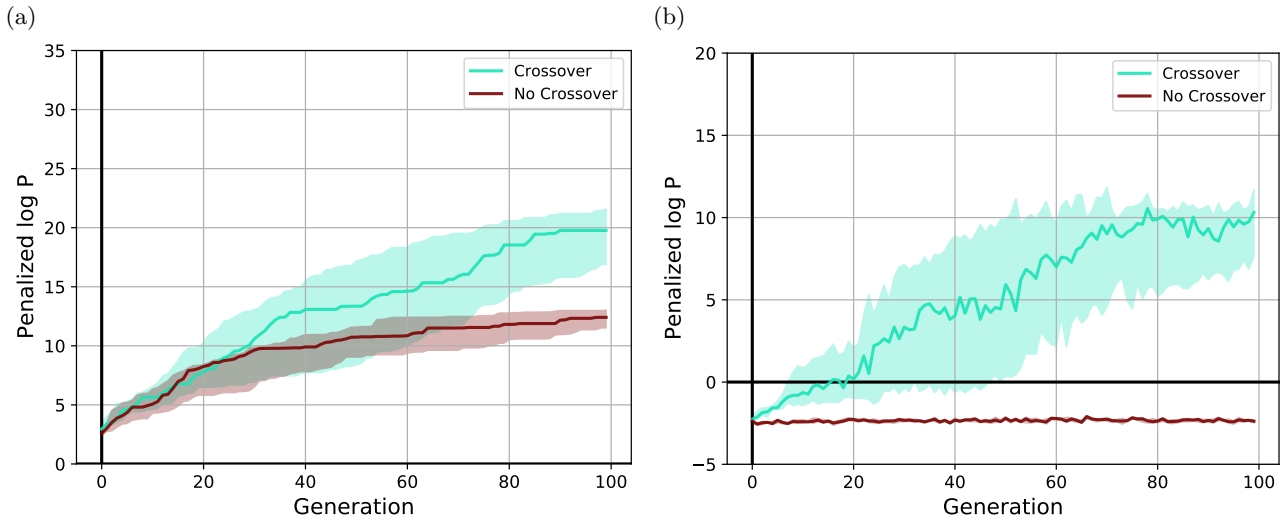


Supplementary Figure 5. Optimization progress of JANUS with four variations of F_{25} parameters in the maximization of the penalized logarithm of the octanol-water partition coefficient (penalized log P). (a) Progress of the median of the highest fitness in each generation across 6 independent runs. (b) Progress of the median-of-medians fitness in each generation of the exploration population across 6 independent runs. The semi-transparent areas in both (a) and (b) depict the fitness intervals between the corresponding first and third quartiles of each generation

.

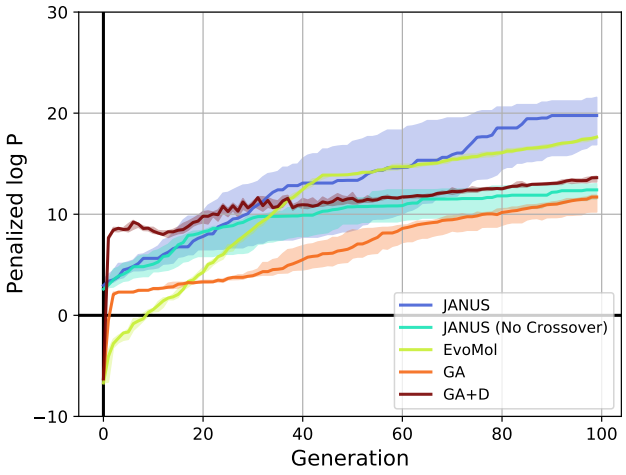
It shows that the majority of molecules generated in the exploration population via mutation does not have a high fitness. The main reason for this is that the molecules generated are not filtered or ordered based on their similarity to the parent molecule but simply selected randomly. However, the majority of molecules generated by mutation differ significantly from the parent as has been established in previous work [30]. Hence, the majority of molecules generated by mutation have not much in common with their parents anymore. In contrast, molecules generated via crossover are similar to both parent molecules as that is explicitly required from them. These results suggest that the performance of JANUS could be enhanced even further by improving the mutated structures generated and work in that regard is ongoing in our group.

Finally, we compared the performance of JANUS with and without crossover in the unconstrained penalized log P optimization task against alternative GA-based molecular design algorithms described previously in the literature, namely GA [31], GA+D [31] and EvoMol [33]. Notably, GA is the same algorithm as GA+D but without the use of a discriminator. It serves as an important ablation study to test the impact of the parallel populations in JANUS as GA differs from JANUS without crossover only by the absence of parallel populations. Overall, there are several points to be noted. First, parallel populations lead to a significant performance improvement across the entire run of 100 generations. Secondly, the discriminator in GA+D significantly impacts the performance as well demonstrating that it will be an important future improvement of JANUS. Finally, EvoMol differs significantly in its trajectory characteristics from JANUS as it is almost deterministic across 10 runs. This is indicated by its almost invisible semi-transparent area depicting the corresponding first and third best fitness quartiles in each generation across 10 runs. This suggests EvoMol to have only a limited stochastic exploration of the chemical space. Nevertheless, EvoMol shows very steady and reliable improvement of the penalized log P in these simulations demonstrating comparable performance to JANUS without additional selection pressure at intermediate numbers of generations.



Supplementary Figure 6. Comparison of the optimization progress of JANUS with and without crossover as genetic operators in the maximization of the penalized logarithm of the octanol-water partition coefficient (penalized log P). (a) Progress of the median of the highest fitness in each generation across 10 independent runs. (b) Progress of the median-of-medians fitness in each generation of the exploration population across 10 independent runs. The semi-transparent areas in both (a) and (b) depict the fitness intervals between the corresponding first and third quartiles of each generation

.



Supplementary Figure 7. Comparison of the optimization progress of JANUS with and without crossover as genetic operators against GA, GA+D and EvoMol, three approaches described in the literature [31, 33], in the maximization of the penalized logarithm of the octanol-water partition coefficient (penalized log P) via the progress of the median of the highest fitness in each generation across 10 independent runs. The semi-transparent areas depict the fitness intervals between the corresponding first and third quartiles of each generation

.

S10. CONSTRAINED PENALIZED LOG P OPTIMIZATION

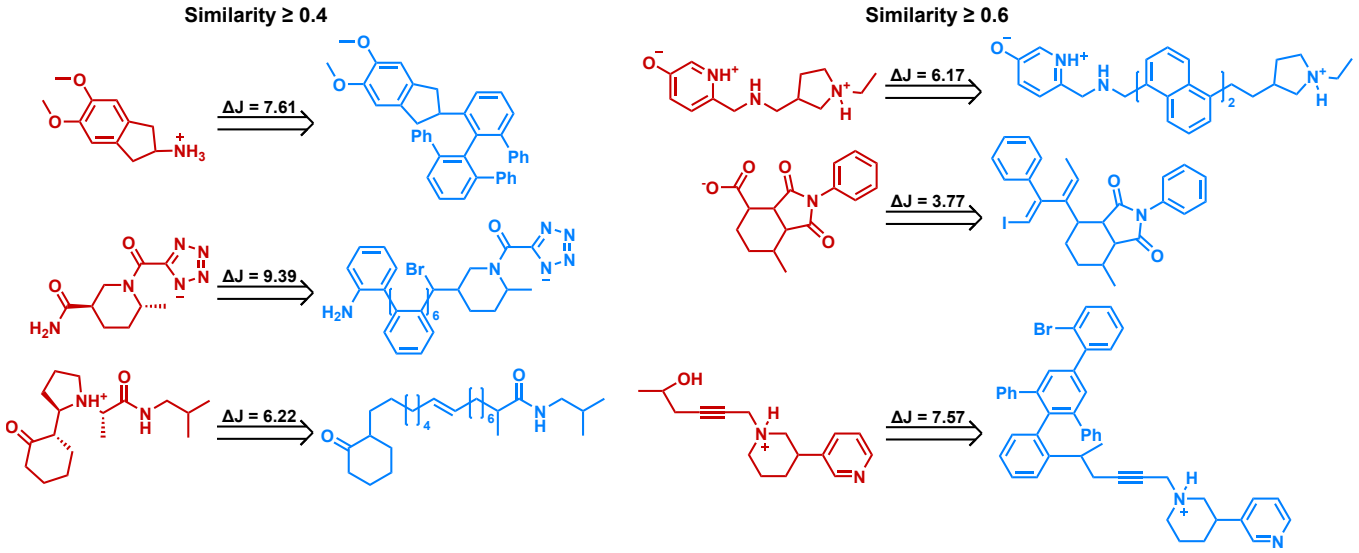
For the constrained penalized log P optimization benchmark, we use the setup employed by Jin et al. [9]. The goal is to improve the penalized log P values of select molecules while maintaining similarity (δ) constraints to them. This task has two subtask, one is to maintain a similarity of at least 0.4 to the original structure, the other is to maintain a similarity of at least 0.6. The tasks are assessed by the improvement in penalized log P values and by success, which is measured by the percentage of structures that were successfully improved in terms of penalized log P within the respective similarity constraints. In total, the benchmark task is to perform this constrained optimization for 800 molecules selected from the ZINC dataset. For each molecule, we run JANUS independently, and the generations are

seeded by the starting structure. Each experiment is run for at most 10 generations. The corresponding results are summarized in Table 3.

It should be noted that JANUS achieved state-of-the-art performance in this benchmark with a lower number of generations than the GA approach developed by Nigam et al. [31] which was run for 20 generations. Similarly, JANUS also outperforms GEGL despite a much lower number of property evaluations, as GEGL was run for 50 generations with a generation size of 16,384 molecules to obtain the corresponding results [34]. Particular pairs of initial structures and structures with maximized constrained penalized log P generated by JANUS are depicted in Supplementary Figure 8. They show that JANUS finds a few very effective modifications that improve the penalized log P while maintaining a sufficient similarity. In particular, keeping less polar substructures but replacing more polar parts with extensive aliphatic or aromatic moieties is very effective in the examples provided.

Supplementary Table 3. Comparison on constrained improvement of penalized log P of specific molecules.

Method	$\delta \geq 0.4$		$\delta \geq 0.6$	
	Improvement	**Success**	**Improvement**	**Success**
JTVAE [9]	0.84 ±1.45	83.6%	0.21 ± 0.71	46.4%
GCPN [17]	2.49 ± 1.30	**100%**	0.79 ± 0.63	**100%**
MMPA [35]	3.29 ± 1.12	-	1.65 ± 1.44	-
DEFactor [11]	3.41 ± 1.67	85.9%	1.55 ± 1.19	72.6%
VJTNN [12]	3.55 ± 1.67	-	2.33 ± 1.17	-
GA [31]	5.93 ± 1.41	**100%**	3.44 ± 1.09	99.8%
GEGL [34]	7.87 ± 1.81	**100%**	4.43 ± 1.53	**100%**
JANUS	**8.34 ± 3.17**	**100%**	**5.29 ± 2.33**	**100%**



Supplementary Figure 8. Selection of structures from the constrained penalized log P optimization task. Red structures are the starting structures, blue structures are the best structures generated by JANUS.

S11. IMITATED INHIBITION

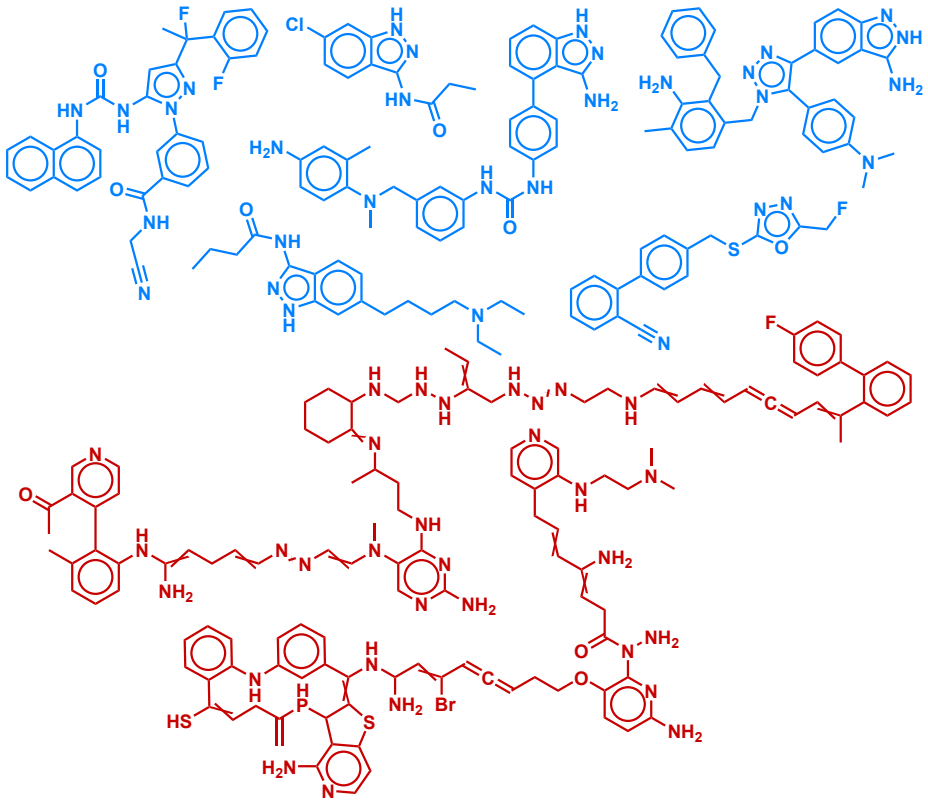
For this task, JANUS is initiated by molecules classified as inhibitors that are provided as a reference dataset in the benchmark. Across multiple generations, a list of molecules that fulfill all the criteria within each experiment is maintained. The mutation and crossover operations are carried out using one random structure selected from this list to replace a molecule from the population. Diversity and novelty of the 5,000 molecules generated are calculated based on the following expressions:

$$\text{Diversity} = 1 - \frac{2}{n(n-1)} \sum_{X,Y} \text{sim}(X, Y) \tag{2}$$

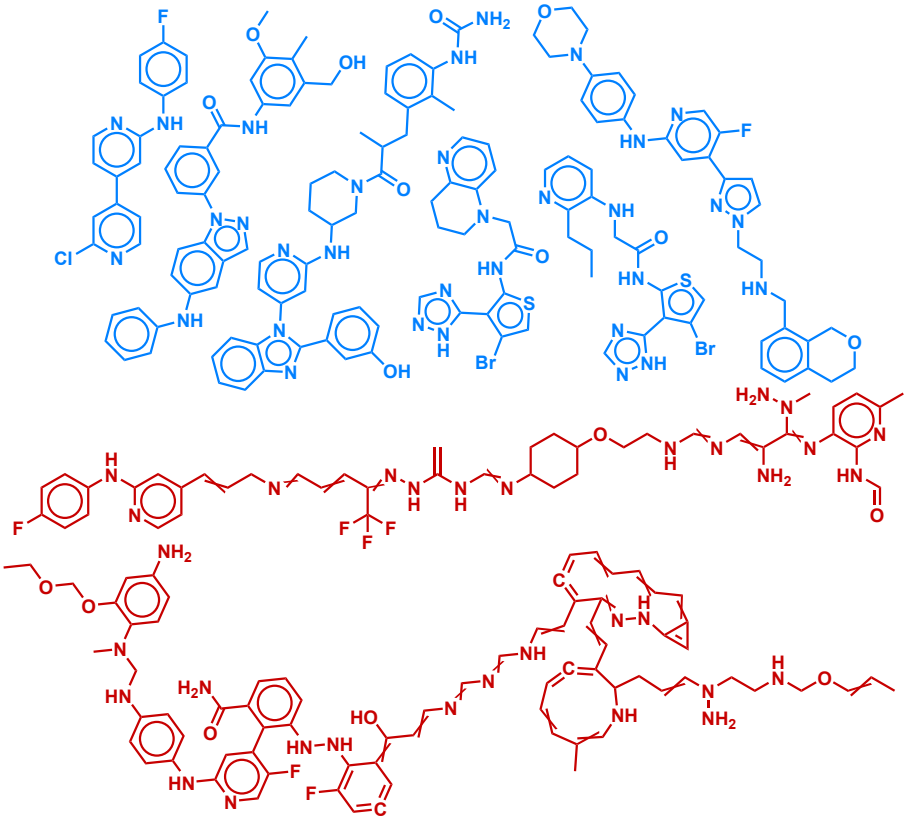
$$\text{Novelty} = \frac{1}{n} \sum_{G} 1\left[\text{sim}(G, G_{\text{SNN}}) < 0.4\right] \tag{3}$$

The expression $\text{sim}(X, Y)$ computes the pairwise molecular similarity for all n structures calculated as the Tanimoto distance of the Morgan fingerprint (calculated with a radius of size 3 and a 2048 bit size) [36]. In addition, for a given compound G fulfilling the requirements of the benchmark, G_{SNN} refers to the molecule from the reference dataset that is closest to G in terms of molecular similarity.

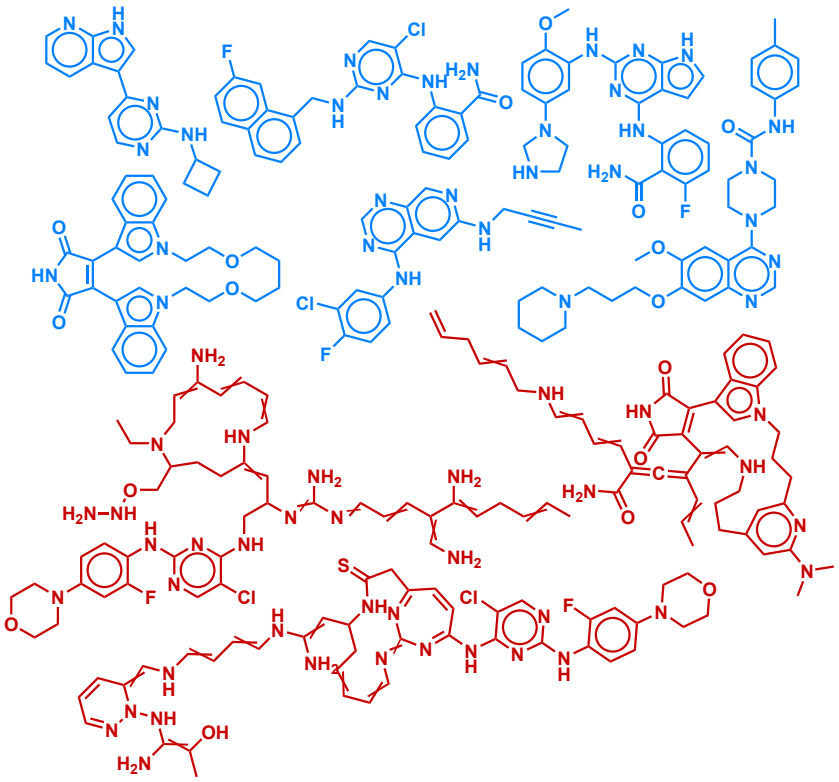
Supplementary Figures 9 – 12 illustrate some structures produced by JANUS+C(50%) (cf. Table 3 of the main text) in the imitated inhibition tasks. Blue structures are considered reasonable, red structures are considered problematic based on expert opinion. It should be noted that some very large and unstable structures are considered favorable inhibitors.



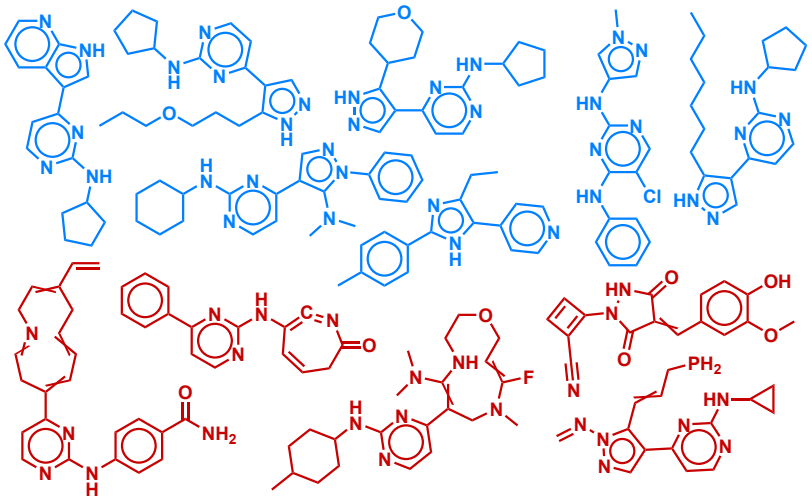
Supplementary Figure 9. Selection of structures from the imitated inhibition task optimizing for GSK3β inhibition. Blue structures are considered reasonable, red structures are considered problematic in terms of stability based on expert opinion.



Supplementary Figure 10. Selection of structures from the imitated inhibition task optimizing for JNK3 inhibition. Blue structures are considered reasonable, red structures are considered problematic in terms of stability based on expert opinion.



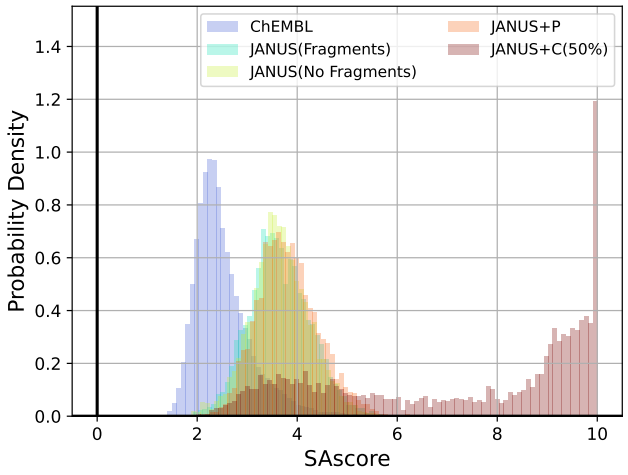
Supplementary Figure 11. Selection of structures from the imitated inhibition task optimizing for both GSK3β and JNK3 inhibition. Blue structures are considered reasonable, red structures are considered problematic in terms of stability based on expert opinion.



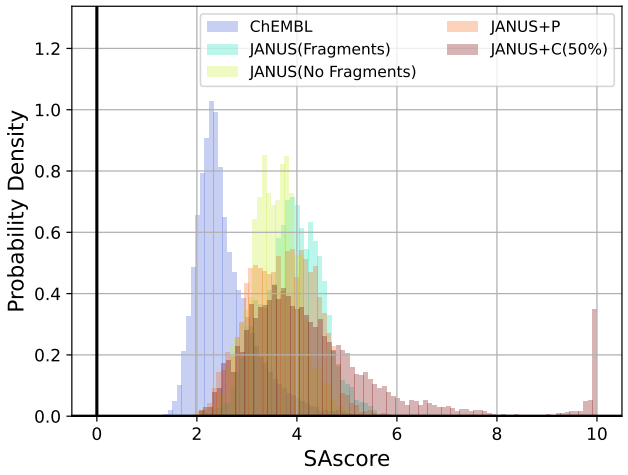
Supplementary Figure 12. Selection of structures from the imitated inhibition task optimizing for both GSK3β and JNK3 inhibition as well as QED and the SAscore. Blue structures are considered reasonable, red structures are considered problematic in terms of stability based on expert opinion.

Supplementary Figures 13 – 15 provide the property histograms for the structures generated by JANUS in the imitated inhibition set of benchmarks with respect to the SAscore [37], the SCScore [38] and the RAscore [39]. Additionally, the corresponding distributions of the reference molecules [35] taken from the ChEMBL database [40, 41] are added for comparison.

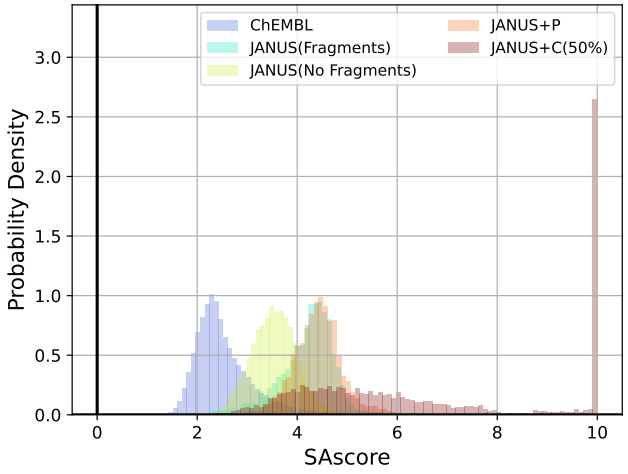
(a) GSK3β



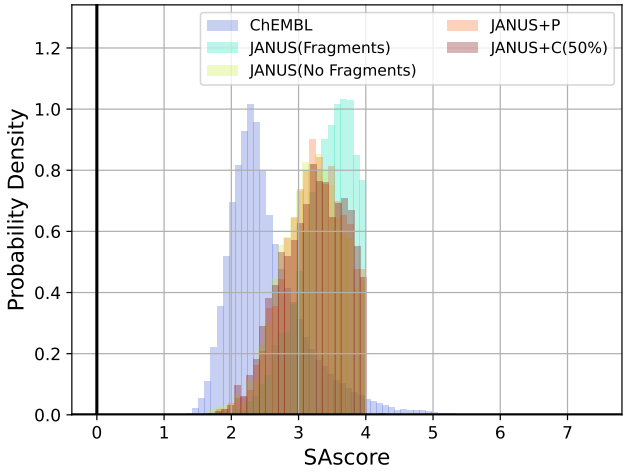
(b) JNK3



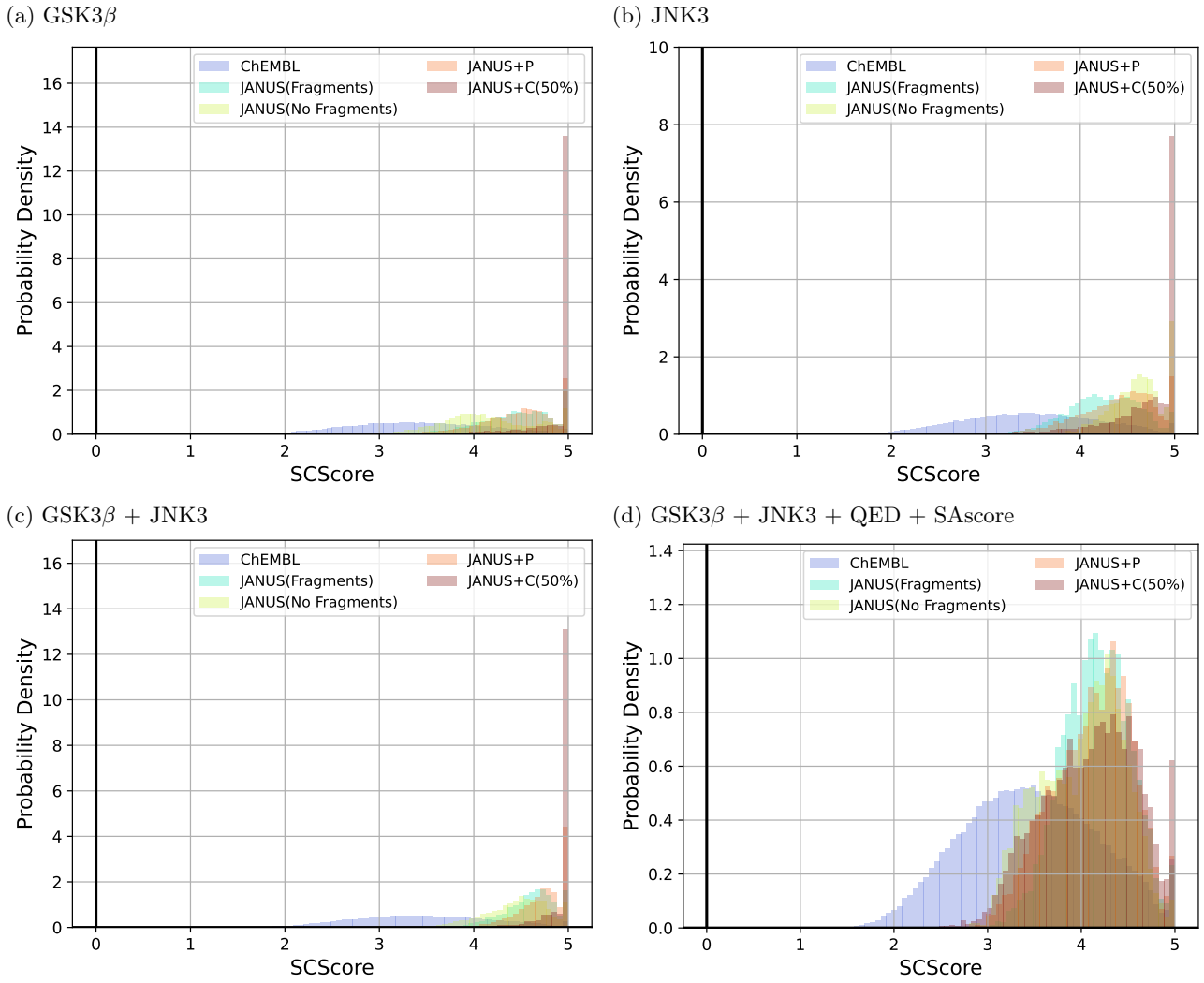
(c) GSK3β + JNK3



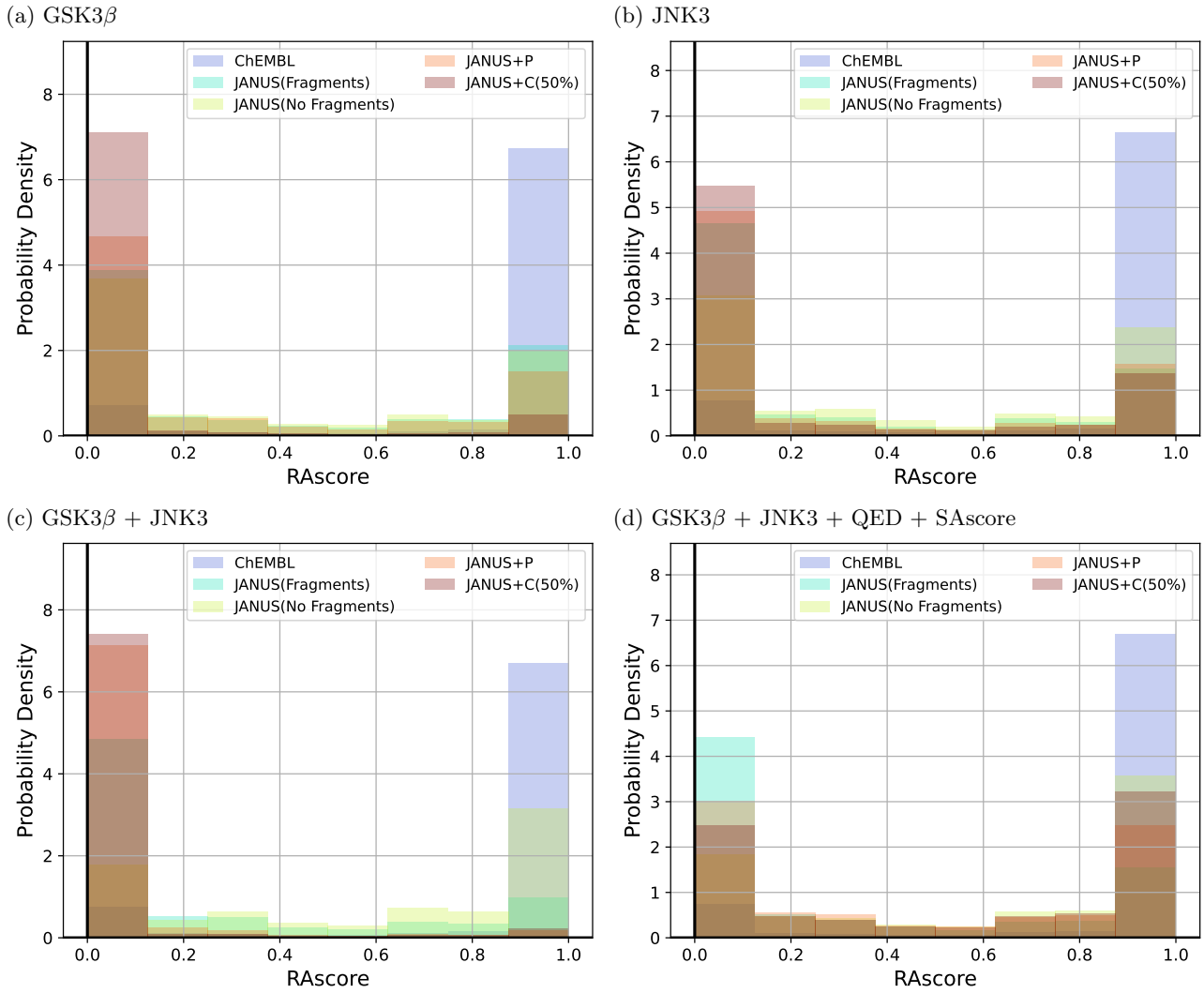
(d) GSK3β + JNK3 + QED + SAscore



Supplementary Figure 13. Histograms based on the SAscore of the molecules that fulfilled all the respective benchmark conditions generated by JANUS with three variations of selection pressure and two different types of mutations in the four imitated inhibition tasks (a-d). The training dataset provided by the authors of the benchmark and taken from the ChEMBL database (labelled ChEMBL) was used to estimate the reference synthesizability scores.



Supplementary Figure 14. Histograms based on the SCScore of the molecules that fulfilled all the respective benchmark conditions generated by JANUS with three variations of selection pressure and two different types of mutations in the four imitated inhibition tasks (a-d). The training dataset provided by the authors of the benchmark and taken from the ChEMBL database (labelled ChEMBL) was used to estimate the reference synthesizability scores.



Supplementary Figure 15. Histograms based on the RAscore of the molecules that fulfilled all the respective benchmark conditions generated by JANUS with three variations of selection pressure and two different types of mutations in the four imitated inhibition tasks (a-d). The training dataset provided by the authors of the benchmark and taken from the ChEMBL database (labelled ChEMBL) was used to estimate the reference synthesizability scores.

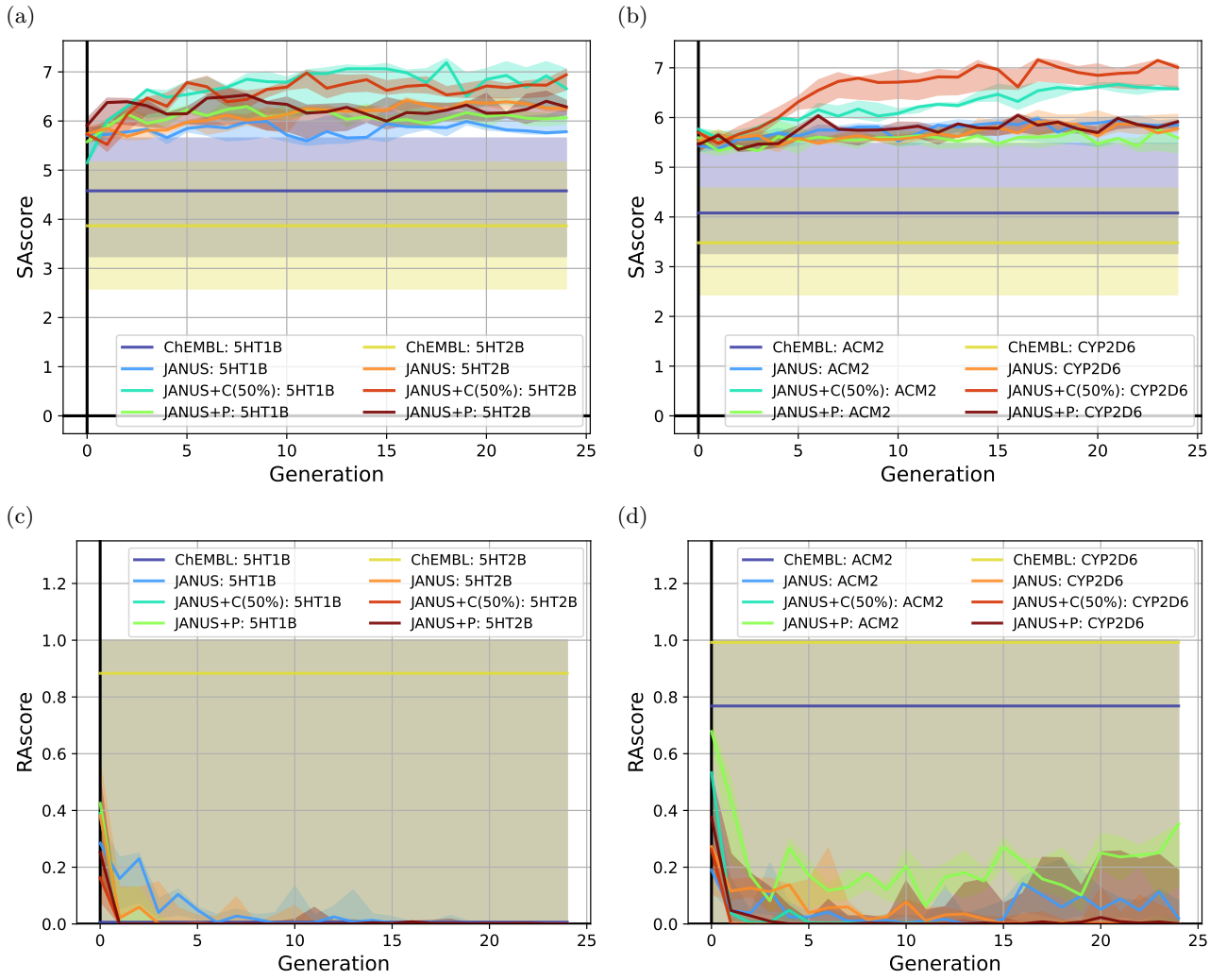
Supplementary Table 4 provides the areas of the bins in the RAscore histograms that are depicted in Supplementary Figure 15.

Supplementary Table 4. Areas of the bins in the RAscore histograms depicted in Supplementary Figure 15. They are based on the RAscore of the molecules that fulfilled all the respective benchmark conditions generated by JANUS with three variations of selection pressure and two different types of mutations in the four imitated inhibition tasks. The training dataset provided by the authors of the benchmark and taken from the ChEMBL database (labelled ChEMBL) was used to estimate the reference RAscore values.

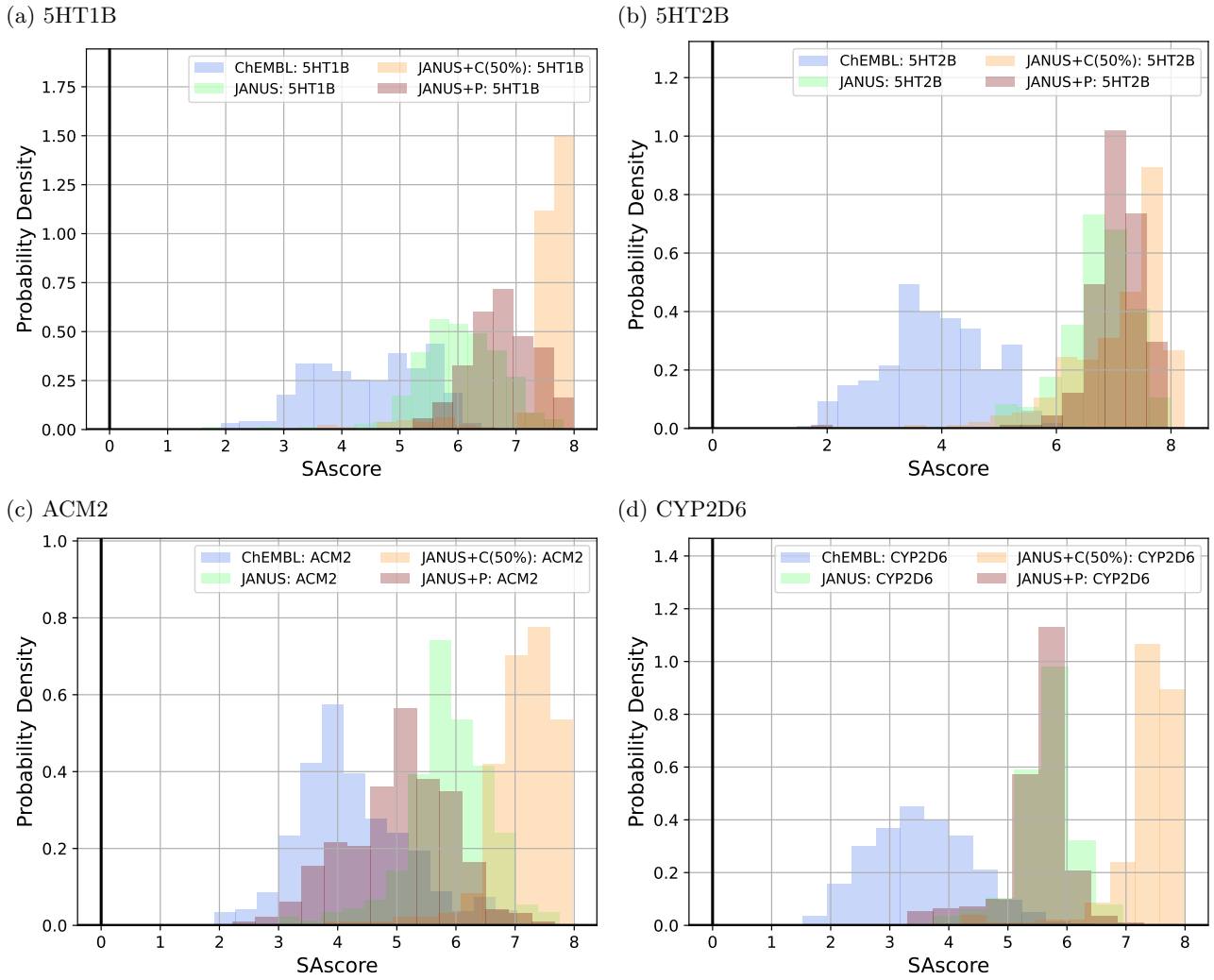
	RAscore Histogram Bin Intervals							
Method	[0,0.125]	(0.125,0.25]	(0.25,0.375]	(0.375,0.5]	(0.5,0.625]	(0.625,0.75]	(0.75,0.875]	(0.875,1]
GSK3β								
ChEMBL	0.088	0.014	0.010	0.008	0.008	0.014	0.017	0.841
JANUS (Fragments)	0.484	0.057	0.044	0.029	0.023	0.049	0.048	0.266
JANUS (No Fragments)	0.461	0.061	0.056	0.034	0.031	0.062	0.045	0.250
JANUS+P	0.582	0.054	0.050	0.027	0.018	0.043	0.039	0.187
JANUS+C(50%)	0.888	0.015	0.009	0.006	0.004	0.007	0.011	0.060
JNK3								
ChEMBL	0.097	0.013	0.010	0.008	0.007	0.014	0.019	0.830
JANUS (Fragments)	0.581	0.058	0.050	0.024	0.018	0.048	0.038	0.184
JANUS (No Fragments)	0.383	0.067	0.072	0.043	0.024	0.061	0.053	0.297
JANUS+P	0.615	0.047	0.039	0.020	0.016	0.035	0.030	0.197
JANUS+C(50%)	0.683	0.034	0.029	0.016	0.013	0.024	0.029	0.172
GSK3β + JNK3								
ChEMBL	0.093	0.013	0.010	0.008	0.008	0.014	0.018	0.836
JANUS (Fragments)	0.604	0.065	0.062	0.031	0.025	0.048	0.041	0.123
JANUS (No Fragments)	0.222	0.055	0.079	0.046	0.036	0.089	0.080	0.393
JANUS+P	0.893	0.031	0.021	0.008	0.006	0.011	0.008	0.023
JANUS+C(50%)	0.925	0.011	0.010	0.006	0.004	0.008	0.008	0.028
GSK3β + JNK3 + QED + SAscore								
ChEMBL	0.093	0.013	0.010	0.008	0.008	0.014	0.018	0.836
JANUS (Fragments)	0.553	0.063	0.049	0.032	0.021	0.044	0.047	0.193
JANUS (No Fragments)	0.228	0.058	0.055	0.036	0.030	0.071	0.075	0.446
JANUS+P	0.376	0.069	0.065	0.034	0.029	0.057	0.060	0.309
JANUS+C(50%)	0.309	0.058	0.049	0.030	0.027	0.058	0.066	0.404

S12. MOLECULAR DOCKING

Supplementary Figure 16 depicts the progress of the synthesizability metrics SAscore [37] and RAscore [39] for all molecules generated in each generation of the molecular docking benchmarks. Supplementary Figures 17 – 19 illustrate the histograms with respect to the synthesizability metrics SAscore, SCScore and RAscore based on the 250 best molecules in each individual run, which is precisely the subset of generated structures used to derive the benchmark scores provided in Table 4 in the main text.

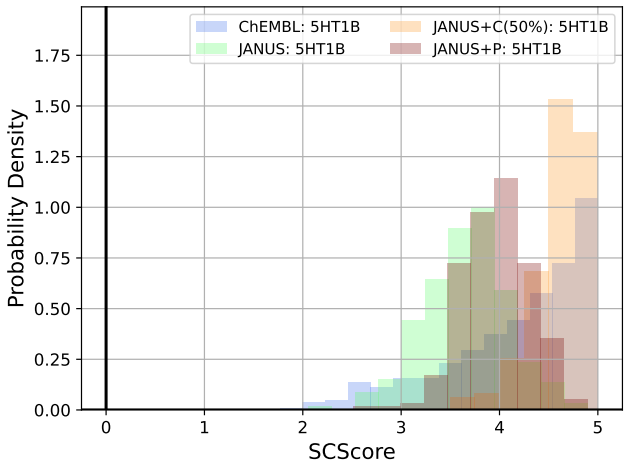


Supplementary Figure 16. Progress of the median-of-median synthesizability scores (a,b) SAscore and (b,c) RAscore of the molecules generated by JANUS with three variations of selection pressure in the minimization of the docking scores to the protein targets (a,c) 5HT1B and 5HT2B, and (b,d) ACM2 and CYP2D6. Progress is depicted via the median of the corresponding median synthesizability scores in each generation across 3 independent runs. The semi-transparent areas depict the synthesizability score intervals between the corresponding 10% and 90% quantiles of each generation. The training dataset provided by the authors of the benchmark taken from the ChEMBL dataset was used to estimate reference synthesizability scores.

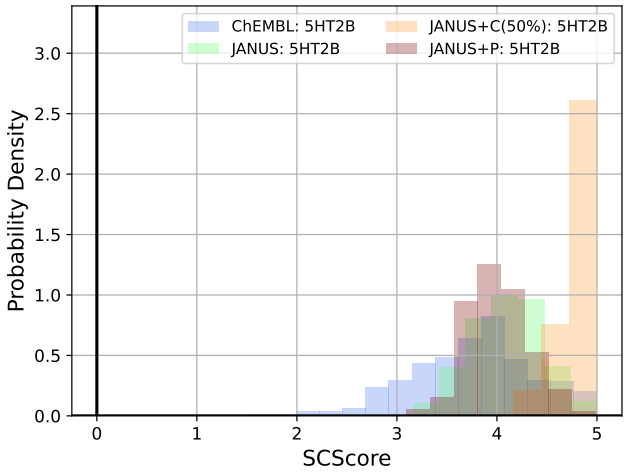


Supplementary Figure 17. Histograms based on the SAscores of the molecules generated by JANUS with three variations of selection pressure in the minimization of the docking scores to the protein targets (a) 5HT1B, (b) 5HT2B, (c) ACM2 and (d) CYP2D6. The training dataset provided by the authors of the benchmark and taken from the ChEMBL database (labelled ChEMBL) was used to estimate the reference synthesizability scores.

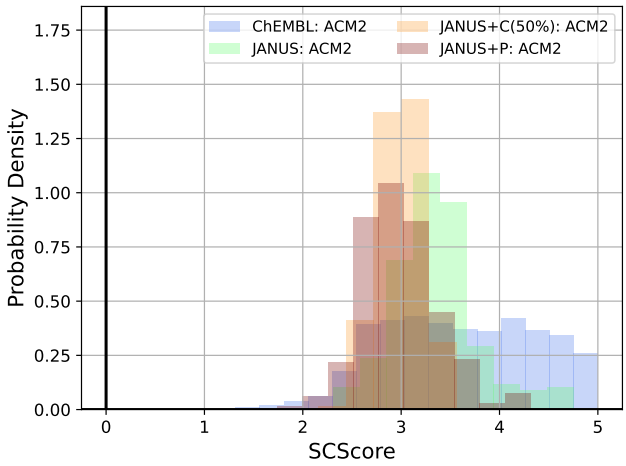
(a) 5HT1B



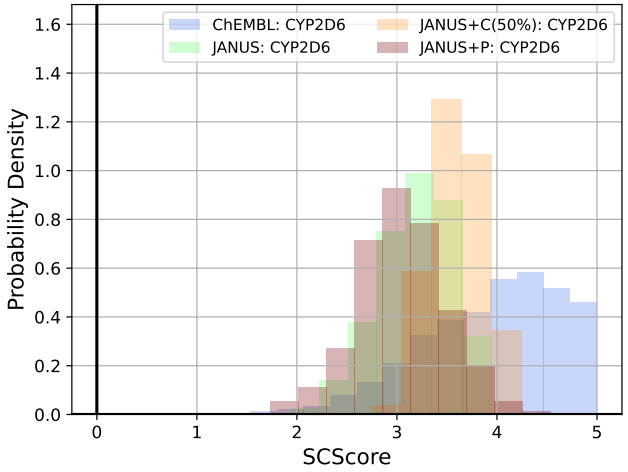
(b) 5HT2B



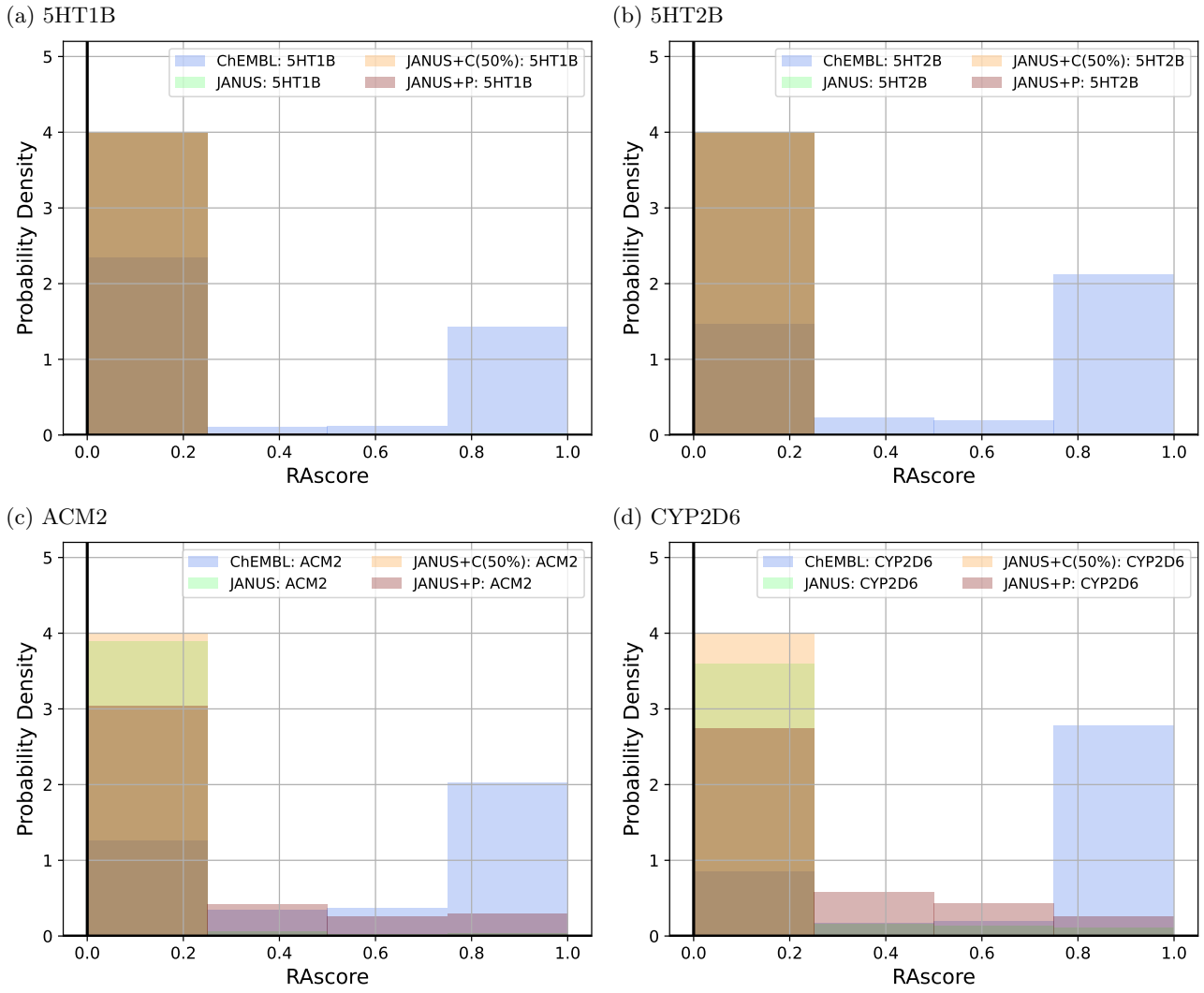
(c) ACM2



(d) CYP2D6

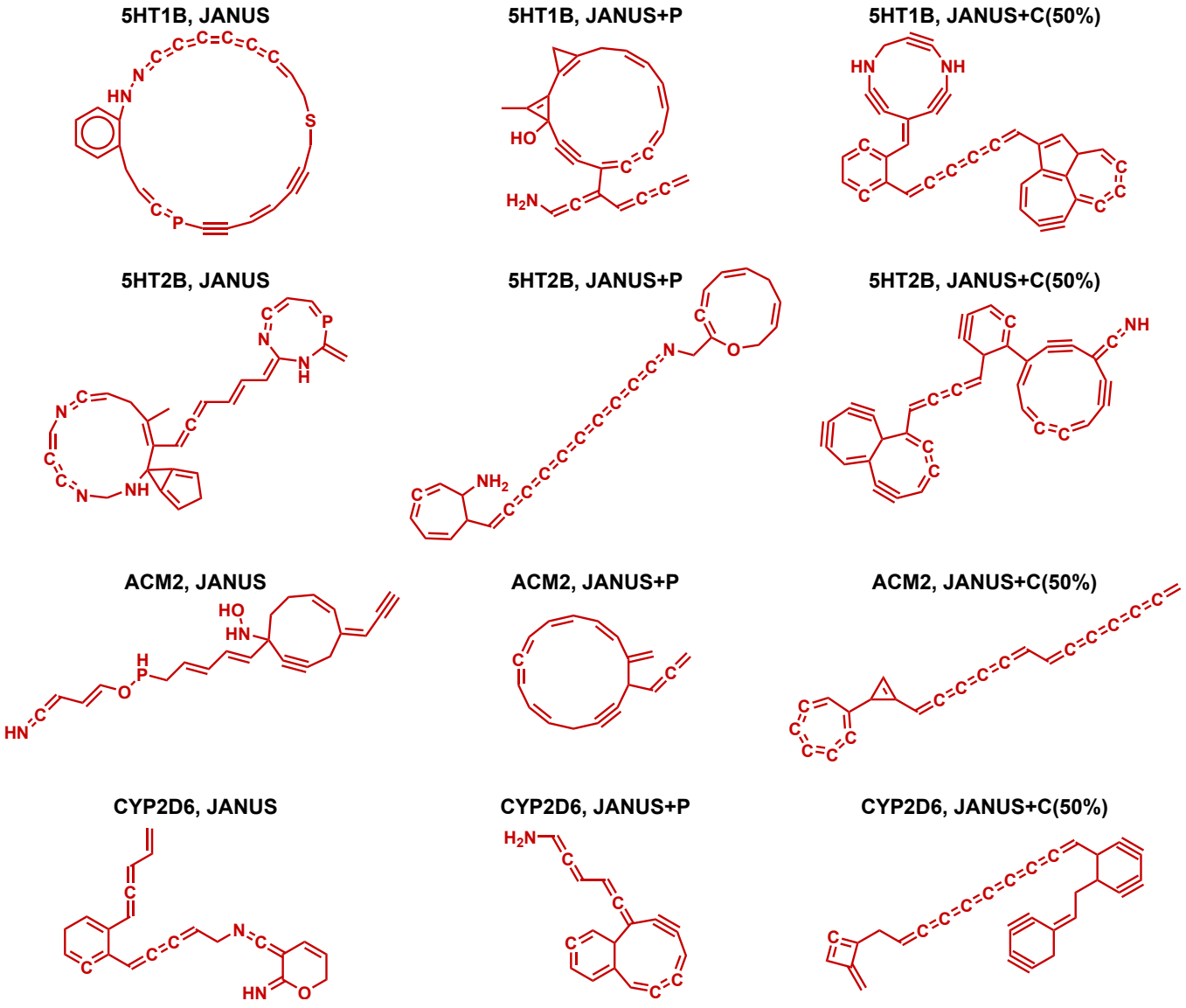


Supplementary Figure 18. Histograms based on the SCScores of the molecules generated by JANUS with three variations of selection pressure in the minimization of the docking scores to the protein targets (a) 5HT1B, (b) 5HT2B, (c) ACM2 and (d) CYP2D6. The training dataset provided by the authors of the benchmark and taken from the ChEMBL database (labelled ChEMBL) was used to estimate the reference synthesizability scores.



Supplementary Figure 19. Histograms based on the RAscores of the molecules generated by JANUS with three variations of selection pressure in the minimization of the docking scores to the protein targets (a) 5HT1B, (b) 5HT2B, (c) ACM2 and (d) CYP2D6. The training dataset provided by the authors of the benchmark and taken from the ChEMBL database (labelled ChEMBL) was used to estimate the reference synthesizability scores.

Supplementary Figure 20 shows for each of the specific docking experiments and JANUS variations that were conducted the proposed structure with the most favorable docking score.



Supplementary Figure 20. Structures with most favorable docking score in each of the specific experiments conducted using JANUS from the docking benchmarks.

S13. GUACAMOL

For the GuacaMol benchmark suite [42], we set the generation size to 10,000 and initiated the population with the top 10,000 molecules from the provided reference dataset. To reduce the computational requirements, we only used mutations as genetic operations. Moreover, no additional selection pressure via DNNs was applied. The local search within the exploitative population is only performed on the fittest member of the explorative population. The results of JANUS on all the benchmark tasks is summarized in Table 5 and compared to various generative models from the literature.

Supplementary Table 5. Comparison of JANUS against literature baselines for the GuacaMol benchmark suite [42]. The entry denoted as "JANUS" does not use additional selection pressure for the exploration population. The following abbreviations are used: redisc. for rediscovery, sim. for similarity.

Benchmark	**STONED** [30]	**SMILES GA** [42]	**SMILES LSTM** [42]	**CReM** [43]	**GB GA** [42]	**MSO** [44]	**EvoMol** [33]	**MolFinder** [45]	**GEGL** [34]	**GA+D** [46]	**JANUS** (here)
Celecoxib redisc.	0.556	0.732	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Troglitazone redisc.	0.543	0.515	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.700	**1.000**
Thiothixene redisc.	0.677	0.598	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.740	**1.000**
Aripiprazole sim.	0.716	0.834	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.580	**1.000**
Albuterol sim.	0.939	0.907	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.710	**1.000**
Mestranol sim.	0.769	0.790	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.570	**1.000**
$C_{11}H_{24}$	**1.000**	0.829	0.993	0.966	0.971	0.997	**1.000**	**1.000**	**1.000**	0.300	**1.000**
$C_9H_{10}N_2O_2PF_2Cl$	0.886	0.889	0.879	0.940	0.982	0.56	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Median molecules 1	0.351	0.334	0.438	0.371	0.406	0.437	**0.455**	0.412	**0.455**	0.270	0.434
Median molecules 2	0.395	0.380	0.422	0.434	0.432	0.395	0.417	**0.454**	0.437	0.270	0.416
Osimertinib MPO	0.863	0.886	0.907	0.995	0.953	0.966	0.969	0.945	**1.000**	0.780	0.967
Fexofenadine MPO	0.878	0.931	0.959	**1.000**	0.998	**1.000**	**1.000**	0.999	**1.000**	0.730	0.999
Ranolazine MPO	0.812	0.881	0.855	**0.969**	0.920	0.931	0.957	0.947	0.958	0.740	0.920
Perindopril MPO	0.629	0.661	0.808	0.815	0.792	0.834	0.827	0.816	**0.882**	0.550	0.817
Amlodipine MPO	0.738	0.722	0.894	0.902	0.894	0.900	0.869	**0.924**	**0.924**	0.620	0.905
Sitagliptin MPO	0.592	0.689	0.545	0.763	0.891	0.868	0.926	**0.948**	0.922	0.510	0.901
Zaleplon MPO	0.674	0.413	0.669	0.770	0.754	0.764	0.793	0.695	**0.834**	0.500	0.774
Valsartan SMARTS	0.864	0.552	0.978	0.994	0.990	0.994	0.998	0.999	**1.000**	0.690	0.993
Deco hop	0.968	0.970	0.996	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.660	**1.000**
Scafold hop	0.854	0.885	0.998	**1.000**	**1.000**	**1.000**	**1.000**	0.948	**1.000**	0.860	**1.000**
Total Score	14.704	14.396	17.340	17.919	17.983	18.086	18.211	18.087	**18.412**	12.040	18.126

[1] AkshatKumar Nigam, Robert Pollice, Matthew F. D. Hurley, Riley J. Hickman, Matteo Aldeghi, Naruki Yoshikawa, Seyone Chithrananda, Vincent A. Voelz, and Alán Aspuru-Guzik. Assigning confidence to molecular property prediction. *Expert Opinion on Drug Discovery*, 16(9):1009–1023, 2021.

[2] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[3] Zhenpeng Yao, Benjamín Sánchez-Lengeling, N Scott Bobbitt, Benjamin J Bucior, Sai Govind Hari Kumar, Sean P Collins, Thomas Burns, Tom K Woo, Omar K Farha, Randall Q Snurr, et al. Inverse design of nanoporous crystalline reticular materials with deep generative models. *Nature Machine Intelligence*, 3(1):76–86, 2021.

[4] Jacques Boitreaud, Vincent Mallet, Carlos Oliver, and Jerome Waldispuhl. Optimol: Optimization of binding affinities in chemical space for drug discovery. *Journal of Chemical Information and Modeling*, 60(12):5658–5666, 2020.

[5] Yashaswi Pathak, Karandeep Singh Juneja, Girish Varma, Masahiro Ehara, and U Deva Priyakumar. Deep learning enabled inorganic material generator. *Physical Chemistry Chemical Physics*, 22(46):26935–26943, 2020.

[6] Jaechang Lim, Sang-Yeon Hwang, Seokhyun Moon, Seungsu Kim, and Woo Youn Kim. Scaffold-based molecular design with a graph generative model. *Chemical Science*, 11(4):1153–1164, 2020.

[7] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1945–1954, 2017.

[8] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

[9] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2323–2332. PMLR, 10–15 Jul 2018.

[10] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained graph variational autoencoders for molecule design. *The Thirty-second Conference on Neural Information Processing Systems*, 2018.

[11] Rim Assouel, Mohamed Ahmed, Marwin H Segler, Amir Saffari, and Yoshua Bengio. Defactor: Differentiable edge factorization-based probabilistic graph generation. *arXiv preprint arXiv:1811.09766*, 2018.

[12] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations*, 2019.

[13] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[14] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L Guimaraes, and Alán Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). *ChemRxiv*, 2017.

[15] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[16] Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. Mol-cyclegan: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1):1–18, 2020.

[17] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pages 6410–6421, 2018.

[18] Sai Krishna Gottipati, Boris Sattarov, Sufeng Niu, Yashaswi Pathak, Haoran Wei, Shengchao Liu, Simon Blackburn, Karam Thomas, Connor Coley, Jian Tang, et al. Learning to navigate the synthetically accessible chemical space using reinforcement learning. In *International Conference on Machine Learning*, pages 3668–3679. PMLR, 2020.

[19] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

[20] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

[21] Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*, 2019.

[22] Danilo Mandic and Jonathon Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley, 2001.

[23] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[24] Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. Chemts: an efficient python library for de novo molecular generation. *Science and technology of advanced materials*, 18(1):972–976, 2017.

[25] Anand A Rajasekar, Karthik Raman, and Balaraman Ravindran. Goal directed molecule generation using monte carlo tree search. *arXiv preprint arXiv:2010.16399*, 2020.

[26] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. {MARS}: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations*, 2021.

[27] Chence Shi*, Minkai Xu*, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020.

[28] Paul Adrien Maurice Dirac. On the theory of quantum mechanics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 112(762):661–677, 1926.

[29] Enrico Fermi. Sulla quantizzazione del gas perfetto monoatomic. *Rendiconti Lincei, 3,*, 3:145–149, 1926.

[30] AkshatKumar Nigam, Robert Pollice, Mario Krenn, Gabriel dos Passos Gomes, and Alan Aspuru-Guzik. Beyond generative models: Superfast traversal, optimization, novelty, exploration and discovery (stoned) algorithm for molecules using selfies. *Chemical Science*, 2021.

[31] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alan Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *International Conference on Learning Representations*, 2020.

[32] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

[33] Jules Leguy, Thomas Cauchy, Marta Glavatskikh, Béatrice Duval, and Benoit Da Mota. Evomol: a flexible and interpretable evolutionary algorithm for unbiased de novo molecular generation. *Journal of Cheminformatics*, 12(1):1–19, 2020.

[34] Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. Guiding deep molecular optimization with genetic exploration. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12008–12021. Curran Associates, Inc., 2020.

[35] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*, pages 4849–4859. PMLR, 2020.

[36] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[37] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):8, 2009.

[38] Connor W. Coley, Luke Rogers, William H. Green, and Klavs F. Jensen. Scscore: Synthetic complexity learned from a reaction corpus. *Journal of Chemical Information and Modeling*, 58(2):252–261, 2018.

[39] Amol Thakkar, Veronika Chadimová, Esben Jannik Bjerrum, Ola Engkvist, and Jean-Louis Reymond. Retrosynthetic accessibility score (rascore) – rapid machine learned synthesizability classification from ai driven retrosynthetic planning. *Chem. Sci.*, 12:3339–3349, 2021.

[40] Anna Gaulton, Anne Hersey, Michał Nowotka, A. Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J. Bellis, Elena Cibrián-Uhalte, Mark Davies, Nathan Dedman, Anneli Karlsson, María Paula Magariños, John P. Overington, George Papadatos, Ines Smit, and Andrew R. Leach. The ChEMBL database in 2017. *Nucleic Acids Research*, 45(D1):D945–D954, 11 2016.

[41] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R Leach. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940, 11 2018.

[42] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[43] Pavel Polishchuk. Crem: chemically reasonable mutations framework for structure generation. *Journal of Cheminformatics*, 12(1):1–18, 2020.

[44] Robin Winter, Floriane Montanari, Andreas Steffen, Hans Briem, Frank Noé, and Djork-Arné Clevert. Efficient multi-objective molecular optimization in a continuous latent space. *Chemical science*, 10(34):8016–8024, 2019.

[45] Yongbeom Kwon and Juyong Lee. Molfinder: an evolutionary algorithm for the global optimization of molecular properties and the extensive exploration of chemical space using smiles. *Journal of cheminformatics*, 13(1):1–14, 2021.

[46] Kevin Maik Jablonka, Fergus Mcilwaine, Susana Garcia, Berend Smit, and Brian Yoo. A reproducibility study of" augmenting genetic algorithms with deep neural networks for exploring the chemical space". *arXiv preprint arXiv:2102.00700*, 2021.