Directly Optimizing for Synthesizability in Generative Molecular Design using Retrosynthesis Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Synthesizability in generative molecular design remains a pressing challenge. Existing methods to assess synthesizability span heuristics-based methods, retrosynthesis models, and synthesizability-constrained molecular generation. The latter has become increasingly prevalent and proceeds by defining a set of permitted actions a model can take when generating molecules, such that all generations are anchored in "synthetically-feasible" chemical transformations. To date, retrosynthesis models have been mostly used as a post-hoc filtering tool as their inference cost remains prohibitive to use directly in an optimization loop. In this work, we show that with a sufficiently sample-efficient generative model, it is straightforward to directly optimize for synthesizability using retrosynthesis models in goal-directed generation. Under a heavily-constrained computational budget, our model can generate molecules satisfying a multi-parameter drug discovery optimization task while being synthesizable, as deemed by the retrosynthesis model.

1 Introduction

Generative molecular design for drug discovery has recently seen a surge of experimental validation, with many candidate molecules progressing into clinical trials[1]. However, the synthesizability of generated designs remains a pressing challenge. Regardless of how "good" generated molecules are, they must be synthesized and experimentally validated to be of use, and work has shown that many generative models propose molecules for which finding a viable synthetic route for, is *at the very least* not straightforward[2,3]. Existing works tackle synthesizability in generative molecular design either by heuristics[4,5], learning synthetic complexity from reaction corpus[6], retrosynthesis models which predict synthetic routes[7–18], or enforce a notion of synthesizability directly in the generative process[19–29].

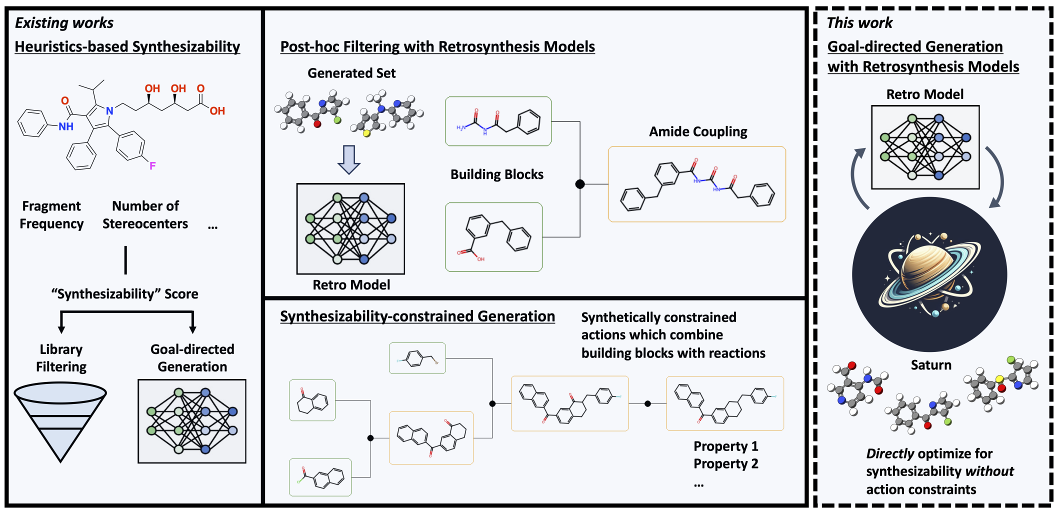
Recently, synthesizability-constrained generative models[19–29] have become increasingly prevalent. A typical metric to *quantify* synthesizability is whether a retrosynthesis model can solve a route for the generated molecules[28,29]. It is common practice to apply retrosynthesis models during post-hoc filtering due to their inference cost[2,3].

On the other hand, sample efficiency is also a pressing challenge, which concerns with how many oracle calls (computational predictions of molecular properties) are required to optimize an objective function. When these oracle calls are computationally expensive, such as in binding affinity predictions, there is a practical limit to an *acceptable* oracle budget for real-world model deployment. The Practical Molecular Optimization (PMO) benchmark[30] highlighted the importance of sample efficiency and since then, more recent works have explicitly considered an oracle budget[31–40].

Recently, Saturn[40], which is a language-based molecular generative model leveraging the Mamba[41] architecture, displayed state-of-the-art sample efficiency compared to 22 models. In this work, we build on Saturn and show that with a *sufficiently sample-efficient* model, one can treat retrosynthesis models as an oracle[42] and directly optimize for generating molecules where synthesis routes can be solved for (Fig. 1). We compare to the recent Reaction-GFlowNet (RGFN)[29] model and show that Saturn can optimize their proposed multi-parameter optimization (MPO) task to generate molecules with good docking scores (to predict binding affinity) and is synthesizable (as deemed by a retrosynthesis model) with 1/400th the oracle budget (1,000 calls instead of 400,000).

We strictly emphasize that we neither claim to solve synthesizability nor claim our model *guarantees* synthesizability. Rather, the take-home message of this work is that if one wants to optimize certain properties, then they should be included in the MPO objective function. If the downstream metric is whether a retrosynthesis tool can solve a route for the generated molecules[2,28,29], then the tool itself should be part of the objective function (given that the model is not synthesizability-constrained).

2 Related Work



Existing works
Heuristics-based Synthesizability
Fragment Frequency
Number of Stereocenters
...
"Synthesizability" Score
Library Filtering
Goal-directed Generation
Post-hoc Filtering with Retrosynthesis Models
Generated Set
Retro Model
Building Blocks
Amide Coupling
Synthesizability-constrained Generation
Synthetically constrained actions which combine building blocks with reactions
Property 1
Property 2
...
This work
Goal-directed Generation with Retrosynthesis Models
Retro Model
Saturn
Directly optimize for synthesizability *without* action constraints

Figure 1: Overview of algorithmic methods to handle synthesizability in generative molecular design.

Synthesizability Metrics. Quantifying and defining synthesizability is non-trivial and early metrics assess *molecular complexity* rather than synthesizability explicitly. Exemplary works include the Synthetic Accessibility (SA) score[4] and SYnthetic Bayesian Accessibility (SYBA)[5] which are based on the frequency of chemical groups in databases. The Synthetic Complexity (SC) score[6] is trained on Reaxys data to measure molecular complexity and implicitly considers the number of synthetic steps required to make a target molecule. There is a correlation between these scores and whether retrosynthesis tools can solve a route[43]. The recent Focused Synthesizability (FS) score[44] incorporated domain-expert *preferences*[45] to assess synthesizability.

Retrosynthesis Models. Given a target molecule, retrosynthesis models propose viable synthetic routes by combining commercial building blocks (starting reagents) with reaction templates (coded patterns that map chemical reaction compatibility) or template-free approaches (learned patterns from data). Exemplary examples include the first work applying Monte Carlo tree search (MCTS) for retrosynthesis[10], SYNTHIA[46,47], AiZynthFinder[11–13], ASKCOS[15], Eli Lilly's LillyMol retrosynthesis model[48], Molecule.one's M1 platform[49], and IBM RXN[16,50,51]. We further highlight surrogate models including Retrosynthesis Accessibility (RA) score[14] and RetroGNN[17] trained on the output of retrosynthesis models for faster inference. Note that these models output a score rather than synthetic routes.

Synthesizability-constrained Molecular Generation. More recently, molecular generative models have been designed with a notion of synthesizability, for example by enforcing transformations from a set of permitted reaction templates. Expansion methods include SYNOPSIS[19], Design of Genuine Structures (DOGS)[20], and RENATE[52]. Other models include MOLECULE CHEF[21], Synthesis Directed Acyclic Graph (DAG)[22], ChemBO[23], SynNet[26], and SyntheMol[27]. Models that also use reinforcement learning (RL) include Policy Gradient for Forward Synthesis (PGFS)[24], Reaction-driven Objective Reinforcement (REACTOR)[25], and LibINVENT[53]. Recent works have equipped GFlowNets[54] with reaction templates, including SynFlowNet[28] and RGFN[29]. Finally, very recent work proposes a new paradigm of "projecting" unsynthesizable molecules into similar, but synthesizable analogs[55].

Goal-directed Generation with Synthesizability Metrics. An alternative to synthesizability-constrained molecular generation is to task molecular generative models to also optimize for synthesizability metrics[44], with common ones being SA score[2,3]. Although SA score assesses molecular complexity, it is correlated with whether AiZynthFinder can solve a route[43]. Generally, more confidence is placed on the output of retrosynthesis models in assessing synthesizability and this is reflected in works that assess model performance on whether generated molecules have a solved route[28,29,55]. In this work, we propose to directly incorporate retrosynthesis tools as an oracle in the MPO objective function and show that generated molecules satisfy MPO objectives.

We end this section by reinforcing that quantifying synthesizability is non-trivial and neither reaction templates nor retrosynthesis tools *guarantee* synthesizability. Notably, reaction templates depend on the granularity of their definition, for instance with the inclusion or omission of incompatibilities which affects the false positive rate of matching reagents[56,57]. A concrete example of this is the original paper reporting Enamine REAL which is a "make-on-demand" commercial database with a stated ~80% synthesis success rate[58]. Recently, SyntheMol[27] which enforces reaction templates during molecular generation, ordered 70 compounds from Enamine REAL with 58 successful syntheses (~83%). We wish to emphasize that the point of drawing attention to this is strictly to support our statement that neither reaction templates, make-on-demand libraries (often generated by reaction templates), nor retrosynthesis tools *guarantee* synthesizability. "Make-on-demand libraries" are a remarkable resource.

3 Methods

In this section, we describe in detail the experimental design and highlight caveats in the results. Firstly, we use Saturn[40] as the generative model which uses RL for goal-directed generation and has high sample efficiency. For details of the model, we refer to the original work[40]. In Saturn, we newly implement AiZynthFinder[11–13] and QuickVina2-GPU-2.1[59–61] (for docking) as oracles to match the case study in RGFN[29].

The RGFN work assesses the synthesizability of generated molecules using the quantitative estimate of drug-likeness (QED)[62], SA score[4], and whether AiZynthFinder[11–13] can solve a route. The authors state that the latter better estimates synthesizability. This statement is supported by recent work highlighting how AiZynthFinder predictions can augment medicinal chemists' decision-making, which led to real-world impact in commercial drug discovery projects[63]. The RGFN work features three case studies where the objective function is either to optimize a proxy model for docking scores, optimize a proxy model for biological activity classification, or optimize QuickVina2-GPU-2.1[59–61] docking scores directly. We choose to compare our model on the latter task because proxy models, while offering faster inference, suffer from domain out-of-applicability if generated molecules deviate too far from the training data. This was also stated by the authors and was the motivation for designing the docking case study[29].

Experimental Caveats. As the code for RGFN[29] is not released, we implement their oracle function ourselves. In Appendix C, we describe the steps we took to reproduce their case study faithfully. Here, we instead highlight caveats that make the comparison not exactly apples to apples for transparency.

1. **Pre-training:** Saturn is pre-trained with either ChEMBL 33[64] or ZINC[65]. These datasets, containing bio-active molecules, inherently bias the learned distribution to already known synthesizable entities[2]. On the other hand, RGFN defines a state space based on reaction templates and building blocks. We note, however, that these are common pre-training datasets that many generative models in literature are pre-trained with.

2. **Quantifying Synthesizability:** RGFN handles synthesizability by combining building blocks with reaction templates. By contrast, Saturn is handling synthesizability by optimizing AiZynthFinder. It could be the case that RGFN's reaction templates represent

"true" synthesizability better in some cases. We note, however, that RGFN also evaluates synthesizability using AiZynthFinder, and in principle, any building blocks and templates used in RGFN could be added to a retrosynthesis model.

3. **Docking Results Filtering**. RGFN filters the best generated molecules by whether they pass PoseBusters[66] checks (plausible physicality). We do not consider this as this is essentially an artefact of the oracle. Provided a more accurate oracle, failure naturally decreases. We note that QuickVina2-GPU-2.1 outputs almost all pass the PoseBusters checks (see Appendix H in the RGFN[29] work). Subsequently, RGFN filters molecules to be dissimilar (< 0.4 Tanimoto similarity) to known aggregators based on the Aggregation Advisor dataset[67] as the *final set*. We also do not consider this. RGFN reports statistics *before* this final aggregator filtering and *these* are the results we compare to (Table 1 in the RGFN[29] work).

4. **Objective Function.** The RGFN work (which also reports results for GraphGA[68], SyntheMol[27], and FGFN[69]), defines the objective function to only optimize for docking score, but assesses generated molecules also by their QED and SA scores. It is unclear the performance of these models if the objective function were modified to also enforce these properties.

Still, despite these caveats, the message we convey is that if one wants to optimize for downstream metrics, then they should be included in the MPO objective function. This is often impractical because certain oracles are computationally expensive and generative models are not efficient enough to directly optimize them. Provided a model *is* sufficiently sample-efficient, generative models can be tasked to optimize *anything* (this does *not* mean that it will *always* be able to optimize the objective under the budget).

Experimental Setup. Following the RGFN[29] work, the case study is to generate synthesizable molecules with good docking scores (using QuickVina2-GPU-2.1[59–61]) to ATP-dependent Clp protease proteolytic subunit (ClpP). The objective function is:

$$R_{RGFN}(x) = Docking\ Score(x) \tag{1}$$

where x is a generated molecule. In Saturn, we apply reward shaping so that $R_{RGFN}(x) \in [0, 1]$. As the purpose of this short paper is to convey that retrosynthesis models can be directly optimized as an oracle, we further define two objective functions:

$$R_{\text{All MPO}}(x) = (\text{Docking Score}(x) \times \text{QED}(x) \times \text{SA Score}(x) \times \text{AiZynthFinder}(x))^{\frac{1}{4}} \in [0, 1] \tag{2}$$

$$R_{Double\ MPO}(x) = (Docking\ Score(x) \times AiZynthFinder(x))^{\frac{1}{2}} \in [0, 1] \tag{3}$$

See Appendix H for reward shaping details to normalize Eq. 2 and 3 ∈ [0, 1] and the exponential term which is from the product aggregator that outputs the final reward. The rationale for $R_{All\ MPO}$ (Eq. 2) is because RGFN evaluates generated molecules also by their QED, SA score, and whether AiZynthFinder can solve a route. Since these are the downstream metrics, we include them in the objective function. The rationale for $R_{Double\ MPO}$ is to illustrate a contrast in optimization difficulty as $R_{All\ MPO}$ is inherently more challenging. Still, we show in the Results section that both objective functions can be optimized.

All Saturn experiments are run across 10 seeds (0-9 inclusive) with 1,000 oracle calls. We note this is 1/400th of the oracle budget of the RGFN work (400,000 calls). We compare with RGFN and also GraphGA[68], SyntheMol[27], and Fragment-based GFlowNet (FGFN)[69]. We do not run these models ourselves and take the results from the RGFN work.

Metrics. Following the RGFN[29] work, a **Mode** is defined as a molecule with docking score < -10. **Discovered Modes** denotes the set of generated Modes that also possess Tanimoto similarity < 0.5 to every other mode. We note that Modes with > 0.5 Tanimoto similarity with other Modes are still valuable, as given a pair of "similar" molecules, there can be a clear preference if for example, one of the molecules contains an undesired substructure. For Saturn results, we additionally report **Yield** which denotes the total number of unique molecules generated with docking score < -10.

4 Results and Discussion

We devise three experiments: optimizing only docking score (following the RGFN[29] work), jointly optimizing docking and AiZynthFinder, and lastly, showing that AiZynthFinder can *still* be directly optimized as an oracle even if none of the molecules in the training data for the generative model can be solved by AiZynthFinder. We construct a custom dataset for this last case study.

4.1 Experiment 1: Optimizing only docking score leads to unreasonable molecules

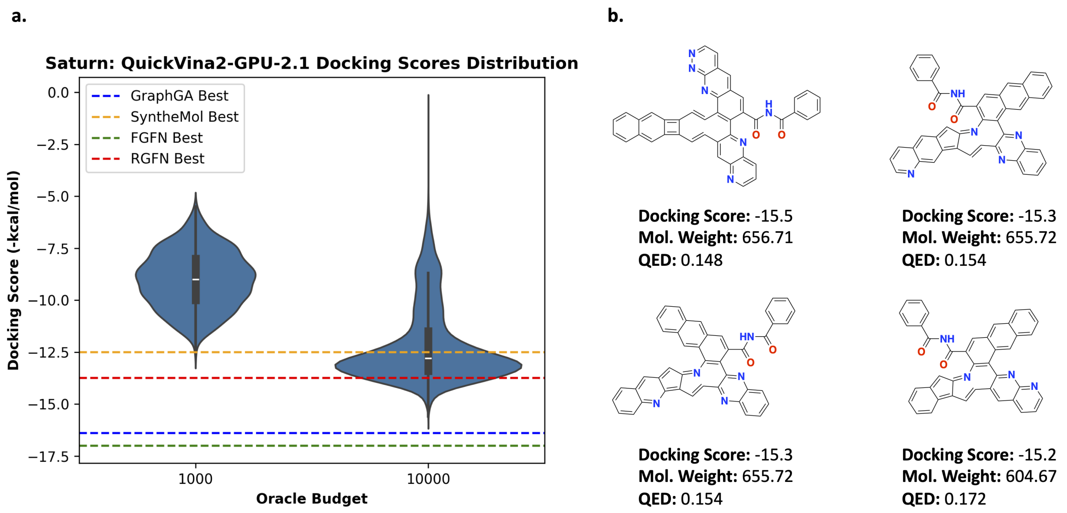


Figure 2: Experiment 1: Optimizing only docking scores. **a.** Distribution of docking scores at varying oracle budgets. The best docking score across comparison methods (taken from the RGFN[29] work) are annotated as dotted lines. **b.** Example lipophilic molecules generated by Saturn with the best docking scores.

We first present results for the R_{RGFN} (Eq. 1) objective function which only optimizes for docking scores against ClpP. It is generally not advised to optimize this in isolation because docking oracles can be highly exploitable, such that lipophilic (lots of carbon atoms and high logP) molecules (promiscuous binders with solubility issues[70]) receive good docking scores. We show that with 10,000 oracle calls, Saturn (trained on ChEMBL 33[64]) generates molecules with approximately the same best QuickVina2-GPU-2.1[59–61] docking scores compared to GraphGA[68], SyntheMol[27], FGFN[69], and RGFN[29] which were run with 400,000 oracle calls (40x higher budget). We perform one replicate here as we only want to convey that the objective function is highly exploitable. Fig. 2a shows the distribution of docking scores at varying oracle budgets. We illustrate how the docking oracle can be exploited in Fig. 2b which shows the best molecules generated by Saturn. Although possessing good docking scores, they are lipophilic with high molecular weight and low QED. Consequently, these are not meaningful molecules. Table 1 in the RGFN[29] work shows that the best generated molecules across various models also have low QED: GraphGA (~0.32), FGFN (~0.22), and RGFN (~0.23), suggesting that they are also exploiting the docking oracle. We note that SyntheMol has slightly higher QED (~0.45).

4.2 Experiment 2: Directly optimizing for synthesizability using AiZynthFinder

In the previous section, we have shown that generative models can exploit docking oracles. Yet, docking scores can be valuable as they can be *correlated* with better binding affinity[71] and should be optimized in combination with oracles that modulate physico-chemical properties. In this section, we run Saturn with the $R_{All\ MPO}$ (jointly maximize QED, minimize SA score, minimize docking score, and is AiZynthFinder solvable) and $R_{Double\ MPO}$ (jointly minimize docking score and is AiZynthFinder solvable) objective functions.

Quantitative Results. Table 1 shows the Saturn results and also results taken from RGFN's[29] work. **As stated previously, since the comparison to RGFN is not apples to apples, we focus our**

Table 1: Synthesizability metrics for top-k Modes (**molecules with docking score < -10**). Results are taken from the RGFN[29] paper (it was not stated how many replicates the models were run for). Mol. weight, QED, and SA score results are for the top-500 Modes. AiZynth results are for the top-100 Modes. NR denotes "not reported". All Saturn experiments were run across 10 seeds (0-9 inclusive). The mean and standard deviation are reported. Both Yield and Modes are reported. The number after the configuration denotes the number of successful replicates out of 10 (Modes $\geq$ 1). For Saturn, none of the configurations found 100 Modes in 1,000 oracle calls so the metrics are reported for however many Modes were found.
^{a} Less than 400,000 oracle calls as SyntheMol[27] roll-outs took time. The RGFN authors decided to match the wall time instead.

Method	Modes (Yield)	Mol. weight (↓)	QED (↑)	SA score (↓)	AiZynth (↑)	Oracle calls (Wall time)
Previous work		top-500	top-500	top-500	top-100	
GraphGA[68]	NR	521.0 ± 31.8	0.32 ± 0.07	4.14 ± 0.51	0.00	400,000 (NR)
SyntheMol[27]	NR	458.2 ± 60.7	0.45 ± 0.16	2.86 ± 0.56	0.56	100,000^a (72h)
FGFN[69]	NR	548.6 ± 42.9	0.22 ± 0.03	2.94 ± 0.54	0.25	400,000 (NR)
RGFN[29]	NR	526.2 ± 37.6	0.23 ± 0.04	2.83 ± 0.22	0.65	400,000 (72h)
$R_{All\ MPO}$ **(ours)**	4 objectives (Docking, QED, SA, AiZynth)					
Saturn-ChEMBL (10)	4 ± 1 (5 ± 3)	367.7 ± 15.7	0.70 ± 0.13	2.11 ± 0.19	0.91 ± 0.11	1,000 (2.9h ± 34m)
Saturn-GA-ChEMBL (10)	7 ± 6 (10 ± 9)	373.3 ± 20.9	0.67 ± 0.09	2.08 ± 0.23	0.82 ± 0.17	1,000 (2.1h ± 24m)
Saturn-ZINC (9)	6 ± 3 (8 ± 10)	368.7 ± 27.6	0.79 ± 0.08	2.15 ± 0.22	0.87 ± 0.19	1,000 (2.1h ± 29m)
Saturn-GA-ZINC (10)	7 ± 4 (10 ± 7)	382.8 ± 27.9	0.71 ± 0.08	2.10 ± 0.15	0.85 ± 0.17	1,000 (2.0h ± 26m)
$R_{Double\ MPO}$ **(ours)**	2 objectives (Docking, AiZynth)					
Saturn-ChEMBL (10)	49 ± 19 (175 ± 94)	442.3 ± 26.2	0.36 ± 0.05	2.36 ± 0.17	0.84 ± 0.06	1,000 (2.0h ± 31m)
Saturn-GA-ChEMBL (10)	43 ± 19 (99 ± 54)	436.1 ± 17.2	0.39 ± 0.04	2.35 ± 0.13	0.77 ± 0.07	1,000 (1.7h ± 16m)
Saturn-ZINC (10)	24 ± 17 (71 ± 64)	414.0 ± 20.2	0.52 ± 0.11	2.30 ± 0.25	0.90 ± 0.07	1,000 (1.9h ± 22m)
Saturn-GA-ZINC (10)	30 ± 11 (64 ± 28)	408.1 ± 12.4	0.46 ± 0.05	2.19 ± 0.10	0.86 ± 0.07	1,000 (1.6h ± 16m)

discussion on Saturn. The central message of this section is that molecules satisfying the objective functions can be found within 1,000 oracle calls. An important note is that all RGFN results (top half of Table 1) report results for the top-500 (for Mol. weight, QED[62], SA score[4]) and top-100 (for AiZynthFinder[11–13]) Modes. Saturn does not find 100 Modes in all configurations with 1,000 oracle calls so the metrics are reported for however many Modes were found. Finally, we run Saturn with and without GraphGA-augmented experience replay[40,68] (see Appendix I for details) and pre-trained with both ChEMBL 33[64] and ZINC 250k[65] (see Appendix B for pre-training details). The purpose is to show that the MPO task can be optimized in 1,000 oracle calls using both popular pre-training datasets. We make the following observations: by including AiZynthFinder in the objective function, Saturn generates AiZynthFinder solvable molecules. Including QED and SA score in the objective function also optimizes these metrics (contrast $R_{All\ MPO}$ with $R_{Double\ MPO}$ results). Mol. weight is also implicitly minimized because it is a component of QED. $R_{Double\ MPO}$ finds notably more Modes than $R_{All\ MPO}$ because the optimization task is easier. In all cases, 1/400th the oracle budget is sufficient to find at least *some* molecules that optimize the objectives (and are AiZynthFinder solvable). The wall times are not 1/400th because AiZynthFinder is the slowest oracle, even with multi-threading (see Appendix D). Finally, we highlight that although the raw number of Modes generated when using the $R_{All\ MPO}$ objective function is relatively low (in 1,000 oracle calls), if AiZynthFinder *does* accurately predict "true" synthesizability, then these Modes are immediately actionable. Importantly, they satisfy every metric in the objective function (low docking score, high QED, low SA, and is AiZynthFinder solvable). In practice, one wants to identify a small set of *excellent* candidate molecules as fast as possible (oracle calls and/or wall time). See Appendix G for additional experiments, and particularly how *also* optimizing for QED is a considerably more difficult task.

Qualitative Results. Fig. 3 shows the docking pose for the generated molecules with the best docking score (no cherry-picking) across all Saturn configurations. In all cases, the pose conforms to the geometry of the binding cavity and the molecule itself is AiZynthFinder solvable (see Appendix F for the solved routes). Generated molecules using $R_{Double\ MPO}$ have better docking scores than

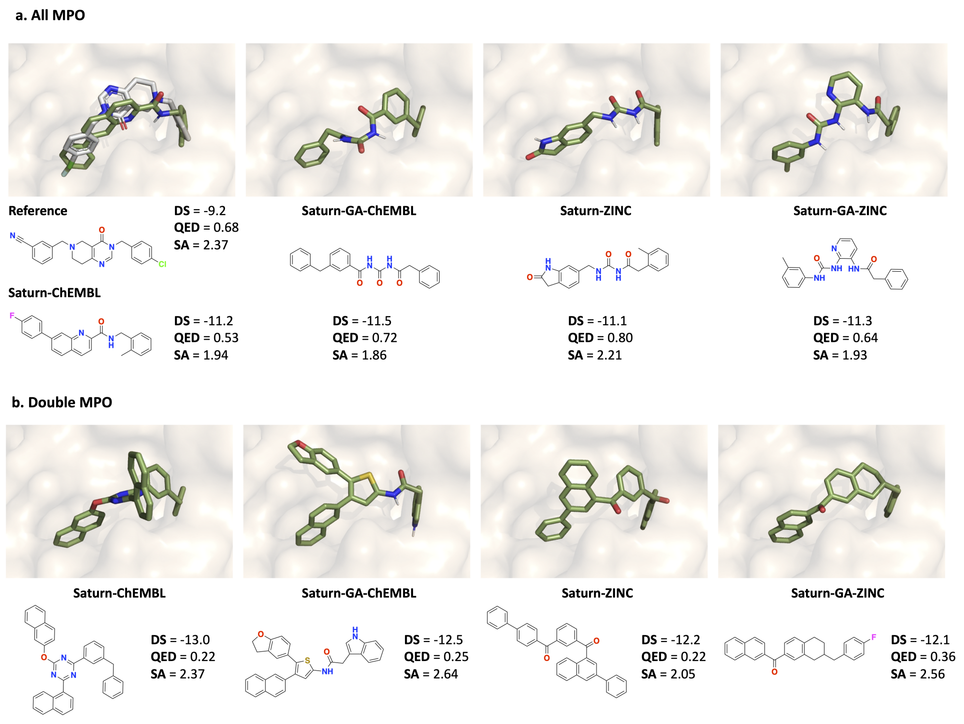


Figure 3: Docked pose of the reference ligand (PDB ID: 7UVU) and generated molecules with the best docking score (DS) across all Saturn configurations and across all 10 seeds (0-9 inclusive). The reference pose is in gray and all generated molecules are in green. **All molecules are AiZynthFinder solvable.** **a.** Molecules generated using $R_{All\ MPO}$. **b.** Molecules generated using $R_{Double\ MPO}$.

$R_{All\ MPO}$, which is expected as the optimization task is easier. In the case of $R_{Double\ MPO}$, the best molecules have docking scores and QED values similar to the best molecules generated by RGFN[29] in 400,000 oracle calls (Fig. 2). We highlight that the molecules from $R_{Double\ MPO}$ possess extensive carbon rings and are likely exploiting the docking oracle.

4.3 Experiment 3: Directly optimizing for synthesizability using AiZynthFinder starting from an unsuitable training distribution

Generative models are pre-trained to model the training data distribution. The experiments thus far use Saturn[40] which has been trained on either ChEMBL 33[64] or ZINC 250k[65]. These datasets contain bio-active molecules and pre-trained models can already generate molecules that can be solved by a retrosynthesis model[2]. In this section, we pre-train Saturn on the fraction of ZINC 250k that *is not* AiZynthFinder[11–13] solvable. This model will be referred to as "Purged ZINC" (see Appendix E for details). The message we convey is that even with an unsuitable training distribution, both $R_{All\ MPO}$ and $R_{Double\ MPO}$ can *still* be optimized under a 1,000 oracle budget.

We showcase how curriculum learning (CL)[72] (decompose a complex optimization objective into sequential, simpler objectives) can be used by defining two phases of goal-directed generation. Firstly, "Purged ZINC" is tasked to minimize SA score[4] (500 oracle budget) as it is correlated with AiZynthFinder[43]. Fig. 4a shows the optimization trajectory and the resulting model is referred to as "Purged ZINC SA". The 500 oracle calls are not counted in the 1,000 oracle budget, as computing SA score is cheap (this process took 56 seconds). Next, to illustrate distribution learning, we sample 1,000 unique molecules from the "Normal ZINC" (trained on the full dataset), "Purged ZINC", and "Purged ZINC SA" models and run AiZynthFinder. Fig. 4b shows the fraction of molecules that are AiZynthFinder solvable. In 56 seconds, the "Purged ZINC" model can be fine-tuned to immediately generate molecules that are almost all solvable (see Appendix G for additional results and discussion). We note that "Purged ZINC" still generates molecules that are AiZynthFinder solvable due to stochastic generation and likely due to the use of SMILES randomization during

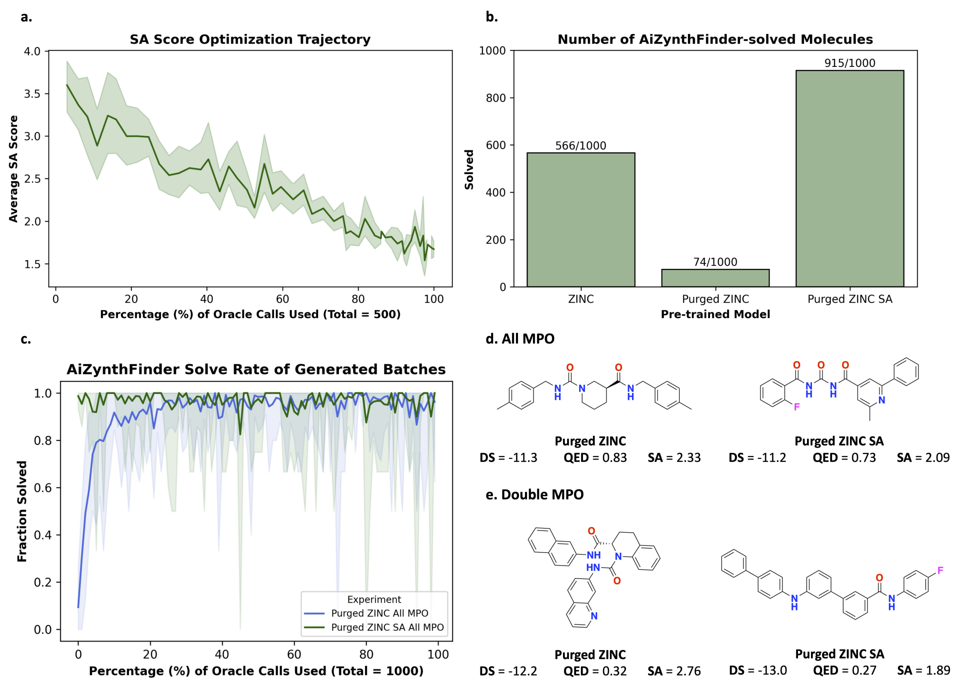


Figure 4: Correlation of SA score and AiZynthFinder solve rate and learning to generate AiZynthFinder solvable molecules. **a.** "Purged ZINC" is tasked to minimize SA score. The average SA score of the sampled batches are shown. **b.** AiZynthFinder solve rates for 1,000 molecules sampled from different models. **c.** All MPO task: fraction of generated molecules (without the GA activated) across all batches that are AiZynthFinder-solvable. Values are the mean and the shaded regions are the minimum-maximum across 10 seeds (0-9 inclusive). **d.** Example molecules generated from the "Purged ZINC" and "Purged ZINC SA" models with the best docking scores (DS).

training which enhances chemical space generalizability[73]. Next, we show how the "Purged ZINC" model can learn to generate molecules that are AiZynthFinder solvable during the course of RL (Fig. 4c). We contrast this with the "Purged ZINC SA" model which has almost 100% solve rate throughout the entire run. During the course of the run, some seeds occasionally generate batches that are not AiZynthFinder solvable (lower bound of shaded region), but this is not detrimental (see Appendix J for more details). Fig. 4d shows the molecules with the best docking score generated across all seeds. The property profiles are essentially the same as the runs with the normal ZINC model (Fig. 3).

Quantitative Results. Table 2 contrasts the results of the ClpP docking case study run across 10 seeds (0-9 inclusive) using the "Normal ZINC", "Purged ZINC", and "Purged ZINC SA" models. Despite an unsuitable training distribution, "Purged ZINC" can still generate Modes that are AiZynthFinder solvable, although the solve rate is slightly lower than "Normal ZINC". "Purged ZINC SA" was first fine-tuned to minimize SA score and already generated mostly AiZynthFinder solvable molecules (Fig. 4b). This process benefits both $R_{All\ MPO}$ (less so) and $R_{Double\ MPO}$ as the Yield and Modes found are higher. Next, we highlight that "Purged ZINC" and "Purged ZINC SA" wall times are longer. This is due to two reasons: firstly, we ran four experiments simultaneously on a single workstation, which shares resources but makes the total wall time to finish all the experiments faster. Secondly, "Purged ZINC SA" experiments took longer because the initial CL fine-tuning biases the model to generate more repeat molecules due to Saturn's[40] mechanism of local chemical space exploration. The effect is that it takes longer to exhaust the 1,000 *unique* oracle calls budget.

Overall, the property profiles of generated Modes are better than GraphGA[68], SyntheMol[27], FGFN[69], and RGFN[29]. Regardless of the starting model, both $R_{All\ MPO}$ and $R_{Double\ MPO}$ can be optimized within 1,000 oracle calls. Whether or not the output of AiZynthFinder represents "true" synthesizability (and the quality of the routes) is beyond the scope of this work. The message we convey in this section is that generating molecules that are solvable by a retrosynthesis model does not

Table 2: Synthesizability metrics for "Normal ZINC" (results from Table 1), "Purged ZINC", and "Purged ZINC SA". All experiments were run across 10 seeds (0-9 inclusive). The mean and standard deviation are reported. Both Yield and Modes are reported. The number after the configuration denotes the number of successful replicates out of 10 (Modes $\geq$ 1). The metrics are reported for however many Modes were found.

Method	Modes (Yield)	Mol. weight (↓)	QED (↑)	SA score (↓)	AiZynth (↑)	Oracle calls (Wall time)
$R_{All\ MPO}$	4 objectives (Docking, QED, SA, AiZynth)					
Normal ZINC						
Saturn (9)	6 ± 3 (8 ± 10)	368.7 ± 27.6	0.79 ± 0.08	2.15 ± 0.22	0.87 ± 0.19	1,000 (2.1h ± 29m)
Saturn-GA (10)	7 ± 4 (10 ± 7)	382.8 ± 27.9	0.71 ± 0.08	2.10 ± 0.15	0.85 ± 0.17	1,000 (2.0h ± 26m)
Purged ZINC						
Saturn (8)	5 ± 5 (9 ± 15)	354.4 ± 26.2	0.72 ± 0.15	1.99 ± 0.27	0.97 ± 0.05	1,000 (2.8h ± 72m)
Saturn-GA(10)	10 ± 3 (14 ± 5)	381.4 ± 15.6	0.68 ± 0.09	2.22 ± 0.24	0.77 ± 0.12	1,000 (2.4h ± 50m)
Purged ZINC SA						
Saturn (10)	9 ± 5 (16 ± 11)	365.9 ± 12.5	0.68 ± 0.09	1.97 ± 0.19	0.96 ± 0.10	1,000 (4.9h ± 54m)
Saturn-GA (10)	12 ± 6 (21 ± 14)	369.7 ± 15.0	0.69 ± 0.08	2.06 ± 0.15	0.89 ± 0.08	1,000 (3.2h ± 26m)
$R_{Double\ MPO}$	2 objectives (Docking, AiZynth)					
Normal ZINC						
Saturn (10)	24 ± 17 (71 ± 64)	414.0 ± 20.2	0.52 ± 0.11	2.30 ± 0.25	0.90 ± 0.07	1,000 (1.9h ± 22m)
Saturn-GA (10)	30 ± 11 (64 ± 28)	408.1 ± 12.4	0.46 ± 0.05	2.19 ± 0.10	0.86 ± 0.07	1,000 (1.6h ± 16m)
Purged ZINC						
Saturn (10)	27 ± 19 (114 ± 107)	425.7 ± 58.5	0.50 ± 0.15	2.66 ± 0.56	0.83 ± 0.13	1,000 (2.4h ± 35m)
Saturn-GA (10)	34 ± 17 (78 ± 57)	410.8 ± 16.3	0.44 ± 0.08	2.29 ± 0.16	0.78 ± 0.08	1,000 (2.2h ± 34m)
Purged ZINC SA						
Saturn (10)	46 ± 14 (268 ± 88)	443.5 ± 31.4	0.39 ± 0.10	2.13 ± 0.12	0.87 ± 0.08	1,000 (3.9h ± 56m)
Saturn-GA (10)	49 ± 10 (187 ± 55)	419.5 ± 10.6	0.42 ± 0.04	2.11 ± 0.05	0.77 ± 0.06	1,000 (2.9h ± 29m)

require synthesizability-constrained design principles. Lastly, we explore the effect of increasing the oracle budget and implications of heuristics-driven synthesizability and post-hoc retrosynthesis model filtering in Appendix G.

5 Conclusion

In this work, we adapt Saturn[40] which is a sample-efficient autoregressive molecular generative model using the Mamba[41] architecture to directly optimize for synthesizability using retrosynthesis models[42]. Our approach contrasts existing works in the field that tackle synthesizability in one of three ways: goal-directed generation with synthesizability heuristic scores such as SA score[4], post-hoc filtering generated molecules with a retrosynthesis model[63], or by enforcing synthesizability design principles in the generative process itself (synthesizability-constrained generation)[26,28,29]. We show that with a sufficiently sample-efficient model, treating retrosynthesis models as an oracle is feasible, and generated molecules can satisfy multi-parameter optimization objectives while being synthesizable (as deemed by a retrosynthesis model). The main comparison results we show are on a molecular docking case study proposed by the recent Reaction-GFlowNet (RGFN)[29] work which is a synthesizability-constrained generative model. With 1/400th the oracle budget, Saturn can generate molecules with better property profiles than GraphGA[68], SyntheMol[27], Fragment GFlowNet (FGFN)[69], and RGFN. Moreover, we conduct an artificial experiment to intentionally purge a training dataset of all molecules that are solvable by the AiZynthFinder[11–13] retrosynthesis model and pre-train a new model with this dataset. Generated molecules from this model are mostly not AiZynthFinder solvable, as expected (Fig. 4b). Despite this, we show that within 1/400th the oracle budget, this model can *still* generate molecules with property profiles better than all comparing models and are synthesizable (as deemed by AiZynthFinder). The take-home message is that with a sufficiently sample-efficient model, it is straightforward to treat retrosynthesis models as an oracle in goal-directed generation. Generating molecules deemed synthesizable by such models does not *require* synthesizability-constrained generation, which is currently, often *sample-inefficient*.

References

1. Yuanqi Du, Arian R Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, and Tom L Blundell. Machine learning-aided generative molecular design. *Nature Machine Intelligence*, pages 1–16, 2024.

2. Wenhao Gao and Connor W Coley. The synthesizability of molecules proposed by generative models. *Journal of chemical information and modeling*, 60(12):5714–5723, 2020.

3. Megan Stanley and Marwin Segler. Fake it until you make it? generative de novo design and virtual screening of synthesizable molecules. *Current Opinion in Structural Biology*, 82:102658, 2023.

4. Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009.

5. Milan Voršilák, Michal Kolář, Ivan Čmelo, and Daniel Svozil. Syba: Bayesian estimation of synthetic accessibility of organic compounds. *Journal of cheminformatics*, 12:1–13, 2020.

6. 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.

7. Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS central science*, 3(10):1103–1113, 2017.

8. Marwin HS Segler and Mark P Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry–A European Journal*, 23(25):5966–5971, 2017.

9. Connor W Coley, Luke Rogers, William H Green, and Klavs F Jensen. Computer-assisted retrosynthesis based on molecular similarity. *ACS central science*, 3(12):1237–1245, 2017.

10. Marwin HS Segler, Mike Preuss, and Mark P Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, 2018.

11. Amol Thakkar, Thierry Kogej, Jean-Louis Reymond, Ola Engkvist, and Esben Jannik Bjerrum. Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain. *Chemical science*, 11(1):154–168, 2020.

12. Samuel Genheden, Amol Thakkar, Veronika Chadimová, Jean-Louis Reymond, Ola Engkvist, and Esben Bjerrum. Aizynthfinder: a fast, robust and flexible open-source software for retrosynthetic planning. *Journal of cheminformatics*, 12(1):70, 2020.

13. Lakshidaa Saigiridharan, Alan Kai Hassen, Helen Lai, Paula Torren-Peraire, Ola Engkvist, and Samuel Genheden. Aizynthfinder 4.0: developments based on learnings from 3 years of industrial application. *Journal of Cheminformatics*, 16(1):57, 2024.

14. 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. *Chemical science*, 12(9):3339–3349, 2021.

15. Connor W Coley, Dale A Thomas III, Justin AM Lummiss, Jonathan N Jaworski, Christopher P Breen, Victor Schultz, Travis Hart, Joshua S Fishman, Luke Rogers, Hanyu Gao, et al. A robotic platform for flow synthesis of organic compounds informed by ai planning. *Science*, 365(6453): eaax1566, 2019.

16. IBM. Rxn for chemistry.

17. Cheng-Hao Liu, Maksym Korablyov, Stanisław Jastrzebski, Paweł Włodarczyk-Pruszynski, Yoshua Bengio, and Marwin Segler. Retrognn: fast estimation of synthesizability for virtual screening and de novo design by learning from slow retrosynthesis software. *Journal of Chemical Information and Modeling*, 62(10):2293–2300, 2022.

18. Kevin Yu, Jihye Roh, Ziang Li, Wenhao Gao, Runzhong Wang, and Connor W Coley. Double-ended synthesis planning with goal-constrained bidirectional search. *arXiv preprint arXiv:2407.06334*, 2024.

19. H Maarten Vinkers, Marc R de Jonge, Frederik FD Daeyaert, Jan Heeres, Lucien MH Koymans, Joop H van Lenthe, Paul J Lewi, Henk Timmerman, Koen Van Aken, and Paul AJ Janssen. Synopsis: synthesize and optimize system in silico. *Journal of medicinal chemistry*, 46(13): 2765–2773, 2003.

20. Markus Hartenfeller, Heiko Zettl, Miriam Walter, Matthias Rupp, Felix Reisen, Ewgenij Proschak, Sascha Weggen, Holger Stark, and Gisbert Schneider. Dogs: reaction-driven de novo design of bioactive compounds. *PLoS computational biology*, 8(2):e1002380, 2012.

21. John Bradshaw, Brooks Paige, Matt J Kusner, Marwin Segler, and José Miguel Hernández-Lobato. A model to search for synthesizable molecules. *Advances in Neural Information Processing Systems*, 32, 2019.

22. John Bradshaw, Brooks Paige, Matt J Kusner, Marwin Segler, and José Miguel Hernández-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags. *Advances in neural information processing systems*, 33:6852–6866, 2020.

23. Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pages 3393–3403. PMLR, 2020.

24. 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.

25. Julien Horwood and Emmanuel Noutahi. Molecular design in synthetically accessible chemical space via deep reinforcement learning. *ACS omega*, 5(51):32984–32994, 2020.

26. Wenhao Gao, Rocío Mercado, and Connor W Coley. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *Proc. 10th International Conference on Learning Representations*, 2022.

27. Kyle Swanson, Gary Liu, Denise B Catacutan, Autumn Arnold, James Zou, and Jonathan M Stokes. Generative ai for designing and validating easily synthesizable and structurally novel antibiotics. *Nature Machine Intelligence*, 6(3):338–353, 2024.

28. Miruna Cretu, Charles Harris, Julien Roy, Emmanuel Bengio, and Pietro Liò. Synflownet: Towards molecule design with guaranteed synthesis pathways. *arXiv preprint arXiv:2405.01155*, 2024.

29. Michał Koziarski, Andrei Rekesh, Dmytro Shevchuk, Almer van der Sloot, Piotr Gaiński, Yoshua Bengio, Cheng-Hao Liu, Mike Tyers, and Robert A Batey. Rgfn: Synthesizable molecular generation using gflownets. *arXiv preprint arXiv:2406.08506*, 2024.

30. Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357, 2022.

31. Soojung Yang, Doyeong Hwang, Seul Lee, Seongok Ryu, and Sung Ju Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation. *Advances in Neural Information Processing Systems*, 34:7924–7936, 2021.

32. Tianfan Fu, Wenhao Gao, Connor Coley, and Jimeng Sun. Reinforced genetic algorithm for structure-based drug design. *Advances in Neural Information Processing Systems*, 35:12325–12338, 2022.

33. Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pages 18872–18892. PMLR, 2023.

34. Seul Lee, Seanie Lee, and Sung Ju Hwang. Drug discovery with dynamic goal-aware fragments. *arXiv preprint arXiv:2310.00841*, 2023.

35. Tony Shen, Mohit Pandey, and Martin Ester. Tacogfn: Target conditioned gflownet for drug design. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023.

36. Jeff Guo, Franziska Knuth, Christian Margreitter, Jon Paul Janet, Kostas Papadopoulos, Ola Engkvist, and Atanas Patronov. Link-invent: generative linker design with reinforcement learning. *Digital Discovery*, 2(2):392–408, 2023.

37. Michael Dodds, Jeff Guo, Thomas Löhr, Alessandro Tibo, Ola Engkvist, and Jon Paul Janet. Sample efficient reinforcement learning with active learning for molecular design. *Chemical Science*, 15(11):4146–4160, 2024.

38. Jeff Guo and Philippe Schwaller. Augmented memory: Sample-efficient generative molecular design with reinforcement learning. *JACS Au*, 2024.

39. Jeff Guo and Philippe Schwaller. Beam enumeration: Probabilistic explainability for sample efficient self-conditioned molecular design. In *Proc. 12th International Conference on Learning Representations*, 2024.

40. Jeff Guo and Philippe Schwaller. Saturn: Sample-efficient generative molecular design using memory manipulation. *arXiv preprint arXiv:2405.17066*, 2024.

41. Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

42. Albin Ekborg. De novo molecular generation of molecules with consistent synthetic strategy. Master's thesis, Chalmers University of Technology, 2024.

43. Grzegorz Skoraczyński, Mateusz Kitlas, Błażej Miasojedow, and Anna Gambin. Critical assessment of synthetic accessibility scores in computer-assisted synthesis planning. *Journal of Cheminformatics*, 15(1):6, 2023.

44. Rebecca M Neeser, Bruno Correia, and Philippe Schwaller. Fsscore: A machine learning-based synthetic feasibility score leveraging human expertise. *arXiv preprint arXiv:2312.12737*, 2023.

45. Oh-Hyeon Choung, Riccardo Vianello, Marwin Segler, Nikolaus Stiefl, and José Jiménez-Luna. Extracting medicinal chemistry intuition via preference machine learning. *Nature Communications*, 14(1):6651, 2023.

46. Sara Szymkuć, Ewa P Gajewska, Tomasz Klucznik, Karol Molga, Piotr Dittwald, Michał Startek, Michał Bajczyk, and Bartosz A Grzybowski. Computer-assisted synthetic planning: the end of the beginning. *Angewandte Chemie International Edition*, 55(20):5904–5937, 2016.

47. Bartosz A Grzybowski, Sara Szymkuć, Ewa P Gajewska, Karol Molga, Piotr Dittwald, Agnieszka Wołos, and Tomasz Klucznik. Chematica: a story of computer code that started to think like a chemist. *Chem*, 4(3):390–398, 2018.

48. Ian A Watson, Jibo Wang, and Christos A Nicolaou. A retrosynthetic analysis algorithm implementation. *Journal of cheminformatics*, 11:1–12, 2019.

49. Molecule.one. The m1 platform.

50. Philippe Schwaller, Riccardo Petraglia, Valerio Zullo, Vishnu H Nair, Rico Andreas Haeuselmann, Riccardo Pisoni, Costas Bekas, Anna Iuliano, and Teodoro Laino. Predicting retrosynthetic pathways using transformer-based models and a hyper-graph exploration strategy. *Chemical science*, 11(12):3316–3325, 2020.

51. Amol Thakkar, Alain C Vaucher, Andrea Byekwaso, Philippe Schwaller, Alessandra Toniato, and Teodoro Laino. Unbiasing retrosynthesis language models with disconnection prompts. *ACS Central Science*, 9(7):1488–1498, 2023.

52. Gian Marco Ghiandoni, Michael J Bodkin, Beining Chen, Dimitar Hristozov, James EA Wallace, James Webster, and Valerie J Gillet. Renate: a pseudo-retrosynthetic tool for synthetically accessible de novo design. *Molecular Informatics*, 41(4):2100207, 2022.

53. Vendy Fialková, Jiaxi Zhao, Kostas Papadopoulos, Ola Engkvist, Esben Jannik Bjerrum, Thierry Kogej, and Atanas Patronov. Libinvent: reaction-based generative scaffold decoration for in silico library design. *Journal of Chemical Information and Modeling*, 62(9):2046–2063, 2021.

54. Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

55. Shitong Luo, Wenhao Gao, Zuofan Wu, Jian Peng, Connor W Coley, and Jianzhu Ma. Projecting molecules into synthesizable chemical spaces. *Proc. 41st International Conference on Machine Learning*, 2024.

56. Karol Molga, Ewa P Gajewska, Sara Szymkuć, and Bartosz A Grzybowski. The logic of translating chemical knowledge into machine-processable forms: a modern playground for physical-organic chemistry. *Reaction Chemistry & Engineering*, 4(9):1506–1521, 2019.

57. Barbara Mikulak-Klucznik, Patrycja Gołębiowska, Alison A Bayly, Oskar Popik, Tomasz Klucznik, Sara Szymkuć, Ewa P Gajewska, Piotr Dittwald, Olga Staszewska-Krajewska, Wiktor Beker, et al. Computational planning of the synthesis of complex natural products. *Nature*, 588 (7836):83–88, 2020.

58. Oleksandr O Grygorenko, Dmytro S Radchenko, Igor Dziuba, Alexander Chuprina, Kateryna E Gubina, and Yurii S Moroz. Generating multibillion chemical space of readily accessible screening compounds. *Iscience*, 23(11), 2020.

59. Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

60. Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with quickvina 2. *Bioinformatics*, 31(13):2214–2216, 2015.

61. Shidi Tang, Ji Ding, Xiangyu Zhu, Zheng Wang, Haitao Zhao, and Jiansheng Wu. Vina-gpu 2.1: towards further optimizing docking speed and precision of autodock vina and its derivatives. *bioRxiv*, pages 2023–11, 2023.

62. G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

63. Jason D Shields, Rachel Howells, Gillian Lamont, Yin Leilei, Andrew Madin, Christopher E Reimann, Hadi Rezaei, Tristan Reuillon, Bryony Smith, Clare Thomson, et al. Aizynth impact on medicinal chemistry practice at astrazeneca. *RSC Medicinal Chemistry*, 15(4):1085–1095, 2024.

64. Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

65. Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

66. Martin Buttenschoen, Garrett M Morris, and Charlotte M Deane. Posebusters: Ai-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 15(9):3130–3139, 2024.

67. John J Irwin, Da Duan, Hayarpi Torosyan, Allison K Doak, Kristin T Ziebart, Teague Sterling, Gurgen Tumanian, and Brian K Shoichet. An aggregation advisor for ligand discovery. *Journal of medicinal chemistry*, 58(17):7076–7087, 2015.

68. Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

69. Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

70. John A Arnott and Sonia Lobo Planey. The influence of lipophilicity in drug discovery and design. *Expert opinion on drug discovery*, 7(10):863–875, 2012.

71. Jeff Guo, Jon Paul Janet, Matthias R Bauer, Eva Nittinger, Kathryn A Giblin, Kostas Papadopoulos, Alexey Voronov, Atanas Patronov, Ola Engkvist, and Christian Margreitter. Dockstream: a docking wrapper to enhance de novo molecular design. *Journal of cheminformatics*, 13:1–21, 2021.

72. Jeff Guo, Vendy Fialková, Juan Diego Arango, Christian Margreitter, Jon Paul Janet, Kostas Papadopoulos, Ola Engkvist, and Atanas Patronov. Improving de novo molecular design with curriculum learning. *Nature Machine Intelligence*, 4(6):555–563, 2022.

73. Josep Arús-Pous, Simon Viet Johansson, Oleksii Prykhodko, Esben Jannik Bjerrum, Christian Tyrchan, Jean-Louis Reymond, Hongming Chen, and Ola Engkvist. Randomized smiles strings improve the quality of molecular generative models. *Journal of cheminformatics*, 11:1–13, 2019.

74. Esben Jannik Bjerrum. Smiles enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076*, 2017.

75. Peter Eastman, Mark S Friedrichs, John D Chodera, Randall J Radmer, Christopher M Bruns, Joy P Ku, Kyle A Beauchamp, Thomas J Lane, Lee-Ping Wang, Diwakar Shukla, et al. Openmm 4: a reusable, extensible, hardware independent library for high performance molecular simulation. *Journal of chemical theory and computation*, 9(1):461–469, 2013.

76. Sereina Riniker and Gregory A Landrum. Better informed distance geometry: using what we know to improve conformation generation. *Journal of chemical information and modeling*, 55 (12):2562–2574, 2015.

77. Anthony K Rappé, Carla J Casewit, KS Colwell, William A Goddard III, and W Mason Skiff. Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American chemical society*, 114(25):10024–10035, 1992.

78. Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3: 1–14, 2011.

79. Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: learning retrosynthetic planning with neural guided a* search. In *International conference on machine learning*, pages 1608–1616. PMLR, 2020.

80. Krzysztof Maziarz, Austin Tripp, Guoqing Liu, Megan Stanley, Shufang Xie, Piotr Gaiński, Philipp Seidl, and Marwin Segler. Re-evaluating retrosynthesis algorithms with syntheseus. *arXiv preprint arXiv:2310.19796*, 2023.

81. Alan Kai Hassen, Martin Sicho, Yorick J van Aalst, Mirjam CW Huizenga, Darcy NR Reynolds, Sohvi Luukkonen, Andrius Bernatavicius, Djork-Arné Clevert, Antonius PA Janssen, Gerard JP van Westen, et al. Generate what you can make: Achieving in-house synthesizability with readily available resources in de novo drug design. *ChemRxiv*, 2024.

82. David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1): 31–36, 1988.

Appendix

The Appendix contains details on the procedure we took to reproduce RGFN's[29] oracle as the code is not released. In addition, we report the computational resources used, how Saturn was pre-trained, and AiZynthFinder execution details.

A Compute Resources

All experiments were run on a single workstation with an NVIDIA RTX A6000 GPU 48GB memory and AMD Ryzen 9 5900X 24-Core CPU. 48GB GPU memory is not required. QuickVina2-GPU-2.1[59–61] with 'thread' = 8,000 (following the RGFN[29] work) takes up to 12GB GPU memory. We further note that Saturn's wall times reported in Table 1 are longer than actually required as we always run 2-4 experiments in parallel, which share the workstation's resources, but makes the *total* wall time less.

B Saturn Pre-training Details

This section contains the exact protocol used for Saturn pre-training on ChEMBL 33[64] and ZINC 250k[65]. The details and pre-trained models are taken from the original Saturn[40] paper and included here.

B.1 ChEMBL 33

Each step is followed by the SMILES remaining after the filtering step.

1. Download raw ChEMBL 33 - 2,372,674
2. Standardization (charge and isotope handling) based on `https://github.com/MolecularAI/ReinventCommunity/blob/master/notebooks/Data_Preparation.ipynb`. All SMILES that could not be parsed by RDKit were removed - 2,312,459
3. Kept only the unique SMILES - 2,203,884
4. Tokenize all SMILES based on REINVENT's tokenizer: `https://github.com/MolecularAI/reinvent-models/blob/main/reinvent_models/reinvent_core/models/vocabulary.py`
5. Keep SMILES $\leq$ 80 tokens - 2,065,099
6. 150 $\leq$ molecular weight $\leq$ 600 - 2,016,970
7. Number of heavy atoms $\leq$ 40 - 1,975,282
8. Number of rings $\leq$ 8 - 1,974,522
9. Size of largest ring $\leq$ 8 - 1,961,690
10. Longest aliphatic carbon chain $\leq$ 5 - 1,950,213
11. Removed SMILES containing the following tokens (due to undesired chemistry and low token frequency): [S+], [C-], [s+], [O], [S@+], [S@@+], [S-], [o+], [NH+], [n-], [N@], [N@@], [N@+], [N@@+], [S@@], [C+], [S@], [c+], [NH2+], [SH], [NH-], [cH-], [O+], [c-], [CH], [SH+], [CH2-], [OH+], [nH+], [SH2] - **1,942,081**

The final vocabulary contained 37 tokens (2 extra tokens were added, indicating <START> and <END>).

The Mamba model has 5,265,920 parameters. The hyperparameters are the default parameters in the code base.

The pre-training parameters were:

1. Max training steps = 20 (each training step entails a full pass through the dataset)
2. Seed = 0
3. Batch size = 512

4. Learning rate = 0.0001
5. Randomize[74] every batch of SMILES

The following checkpoint was used: Epoch 18, NLL = 32.21, Validity (10k) = 95.60%.

B.2 ZINC 250k

ZINC 250k[65] was downloaded and used as is.

The pre-training parameters were:

1. Training steps = 50 (each training step entails a full pass through the dataset)
2. Seed = 0
3. Batch size = 512
4. Learning rate = 0.0001
5. Train with SMILES randomization[74] (all SMILES in each batch was randomized)

The final vocabulary contained 66 tokens (2 extra tokens were added, indicating <START> and <END>).

The Mamba model has 5,272,832 parameters (slightly larger than ChEMBL 33 model because the vocabulary size here is larger). The following checkpoint was used: Epoch 50, NLL = 28.10, Validity (10k) = 95.20%.

C Reproducing RGFN's Oracle

This section contains the steps we took to reproduce RGFN's[29] ATP-dependent Clp protease proteolytic subunit (ClpP) docking case study as faithfully as we could.

Target Preparation. Following Appendix C.1 of the RGFN paper, we downloaded the 7UVU ClpP crystal structure here: `https://www.rcsb.org/structure/7UVU`. All molecules (complexed inhibitors, solvents, etc.) were removed, keeping only two monomeric units. Two structures were saved: The apo protein (no other molecules present) and the reference ligand. **The following step differs from RGFN**: the apo protein was processed with PDBFixer[75] to fix missing atoms and residues. We performed this step because errors were thrown during docking when using the raw apo protein structure.

Docking Details. We implement QuickVina2-GPU-2.1[59–61] following the instructions in the GitHub repository here: `https://github.com/DeltaGroupNJUPT/Vina-GPU-2.1`. The reference ligand structure that was saved out in the previous step is used here to define the docking box. Specifically, the average coordinates of the ligand denote the docking centroid. **The following *may* differ from RGFN**: We define the docking box as 20 Å x 20 Å x 20 Å as it was unclear how it should be defined based on RGFN's protocol. This box size has worked on many other protein targets[71] when docking with AutoDock Vina[59] which is the predecessor of QuickVina2-GPU-2.1.

Docking Workflow. Following RGFN's protocol, QuickVina2-GPU-2.1 used the following parameters: 'thread' = 8,000 with 'search depth' = "heuristic" which is the default. Next, all ligands were docked following RGFN's workflow:

1. Start with batch of generated SMILES from Saturn
2. Canonicalize the SMILES
3. Convert to RDKit Mol objects
4. Protonate the Mols
5. Generate 1 (lowest energy) conformer using 'ETKDG'[76]
6. Minimize energy with the Universal Force Field (UFF)[77]
7. Write out the conformers as 'PDB' files
8. Using Open Babel[78], convert the 'PDB' to 'PDBQT' format

9. Execute QuickVina2-GPU-2.1 docking

Protocol Validation. We make further efforts to ensure the oracle is as faithful as possible to RGFN's implementation. When executing QuickVina2-GPU-2.1, if a seed is not specified, a random seed is used. It is unclear if a seed was set in the RGFN[29] work. In our experiments, the seed is 0. We re-dock the reference ligand and find that the pose is similar to Figure 15 in the RGFN work. However, the docking score we obtain is -9.2 whereas the RGFN work reports -10.31. Subsequently, we execute docking 100 times (letting QuickVina2-GPU 2.1 select the random seed) and observed that seed = 448029751 gives a similar pose to RGFN's pose and yields a docking score of -10.1. We additionally found that seed = 1920393356 yields a docking score of -10.3 but the pose is reflected. Finally seed = 673697018 yields a docking score of -8.2 and is a completely different pose. It is intractable to try every seed.

Therefore, we end this section by stating that it is hard to say if we *exactly* re-implement RGFN's[29] docking oracle. However, we believe it still enables us to convey the primary message of our work: retrosynthesis models can be directly treated as an oracle and be explicitly optimized for during generation.

D AiZynthFinder

AiZynthFinder[11–13] was used as is, without modification. The source code was cloned from the GitHub repository here: `https://github.com/MolecularAI/aizynthfinder`. The environment and package were installed following the README. Following the documentation here: `https://molecularai.github.io/aizynthfinder/`, we downloaded the public data and used AiZynthFinder as is. Every batch of molecules generated by Saturn (16 at max) is chunked into 4 sub-sets for multi-thread execution. Finally, we consider a molecule AiZynthFinder "solvable" if the "is_solved" flag is True. This flag denotes whether the top scored (accounting for tree depth and fraction of building blocks in stock)[12,13] is solved.

E AiZynthFinder purged ZINC 250k Pre-training Details.

In Experiment 3, we pre-train Saturn on a sub-set of ZINC 250k[65] that *is not* AiZynthFinder[11–13] solvable. The goal is to show that Saturn can *still* optimize for generating molecules that are AiZynthFinder solvable despite being trained on no molecules that can be.

Purged Dataset. We first run AiZynthFinder on the entirety of ZINC 250k on a single workstation with an NVIDIA RTX A6000 GPU 48GB memory and AMD Ryzen 9 5900X 24-Core CPU. The process was run using multi-threading across 12 workers and took 62 hours. We save the unique SMILES (98,110) of all the molecules that *are not* AiZynthFinder solvable. This is the dataset used for pre-training.

Pre-training. Following the same pre-training parameters used in the original Saturn[40] work:

1. Training steps = 100 (each training step entails a full pass through the dataset)
2. Seed = 0
3. Batch size = 512
4. Learning rate = 0.0001
5. Train with SMILES randomization[74] (all SMILES in each batch was randomized)

The final vocabulary contained 57 tokens (2 extra tokens were added, indicating <START> and <END>). This is less than the normal ZINC 250k model (66 tokens) because some tokens are not present in the purged dataset.

The Mamba model has 5,271,040 parameters (less than the normal 250k model because the vocabulary size is smaller). The following checkpoint was used: Epoch 100, NLL = 27.78, Validity (10k) = 92.27% and the training time was 4.7 hours.

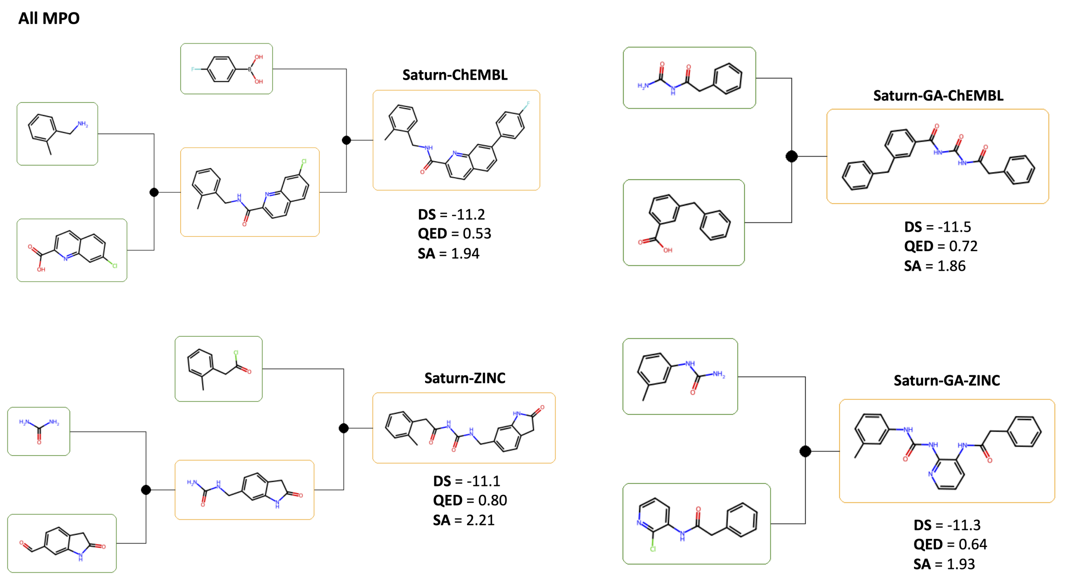


Figure F5: AiZynthFinder solved routes (top-scoring) for All MPO example molecules.

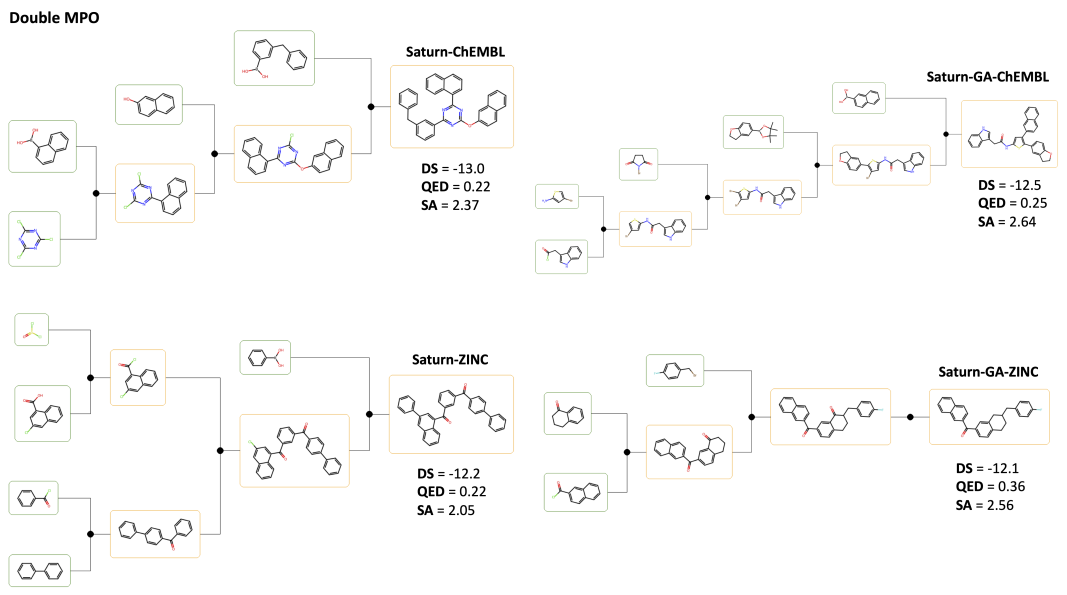


Figure F6: AiZynthFinder solved routes (top-scoring) for Double MPO example molecules.

F AiZynthFinder Routes

The AiZynthFinder solved routes for the 8 example molecules shown in Fig. 3 are shown here. The All MPO routes (Fig. F5) are generally shorter than the Double MPO routes (Fig. F6). This suggests that enforcing QED and SA score also implicitly makes the predicted forward syntheses shorter. We note that it is possible to design an objective function that also aims to generate short paths by rewarding short paths. We do not explore this here and leave it for future work.

G Supplementary Results

In this section, supplementary results are reported which aim to address/provide evidence for three points:

1. Effect of increasing the oracle budget when optimizing AiZynthFinder
2. *Jointly* optimizing QED with docking score is *considerably* more difficult than just optimizing docking score
3. Optimizing SA score *can* be a better allocation of computational resources

Table 3: Synthesizability metrics across various Saturn experiments. Metrics are reported for however many Modes are found. For these supplemental results, only one replicate is performed with seed = 0.

Method	Modes (Yield)	Mol. weight (↓)	QED (↑)	SA score (↓)	AiZynth (↑)	Oracle calls (Wall time)
$R_{All\ MPO}$			4 objectives (Docking, QED, SA, AiZynth)			
Saturn-GA-ChEMBL	108 (222)	370.9	0.84	2.44	0.70	5,000 (23.6h)
Saturn-GA-ZINC	74 (230)	371.1	0.81	2.45	0.69	5,000 (21.2h)
$R_{Double\ MPO}$			2 objectives (Docking, AiZynth)			
Saturn-ChEMBL	302 (3804)	486.2	0.28	2.40	0.82	5,000 (21.4h)
Saturn-GA-ChEMBL	323 (3053)	464.1	0.34	2.51	0.68	5,000 (11.3h)
Saturn-ZINC	266 (2783)	521.2	0.25	2.40	0.76	5,000 (13.5h)
Saturn-GA-ZINC	327 (2741)	455.6	0.34	2.48	0.72	5,000 (11.3h)
$R_{All\ MPO}$ (but without AiZynth)			3 objectives (Docking, QED, SA)			
Saturn-ChEMBL	332 (1219)	376.7	0.80	2.66	0.39	10,000 (2.4h)
Saturn-ZINC	332 (1108)	382.1	0.76	2.43	0.55	10,000 (2.3h)
R_{RGFN} - Results from Fig. 2			1 objective (Docking)			
Saturn-ChEMBL	469 (8389)	511.9	0.26	3.09	0.14	10,000 (2.2h)

Increasing the oracle budget leads to notably increased wall times. In the main text results, $R_{All\ MPO}$ does not find that many Modes. We investigate the effect of increasing the oracle budget (Table 3) with the GA activated (which recover diversity so as to satisfy the Modes criterion that Modes must have < 0.5 Tanimoto similarity with other Modes). More Modes are found but the wall time is *drastically* higher. With 5x the oracle budget (5,000 compared to 1,000 in the main text), one may expect 5x the wall time (12-15 hours) but the wall time is almost 24 hours. The reason is due to Saturn's sampling behaviour which locally explores chemical space[40]. The parameters of Saturn could be changed to loosen this local exploration behaviour but we do not explore this. We demonstrate the application of Saturn out-of-the-box. As a consequence of this, many repeat molecules are generated, which do not impose an oracle call as the reward is retrieved from an oracle cache, but makes the sampled batch (new molecules) smaller. Multi-threading was used to run AiZynthFinder faster (see Appendix D for more details). Consider batches of 1 molecule and 4 molecules. This can take a similar wall time as molecules can be chunked, thus benefiting from multi-threading. This could be mitigated, for example, by using a faster retrosynthesis model which can come with advantages and disadvantages[79,80] and/or CPU parallelization. Finally, we highlight that deactivating the GA will likely lead to higher Yield and AiZynthFinder solve rate, as shown in Tables 1 and 2. We reiterate that activating the GA was to satisfy the Mode metric.

***Jointly* optimizing QED with docking score is *considerably* more difficult than just optimizing docking score.** RGFN[29] reports their mean and standard deviation of QED values as 0.23 ± 0.04

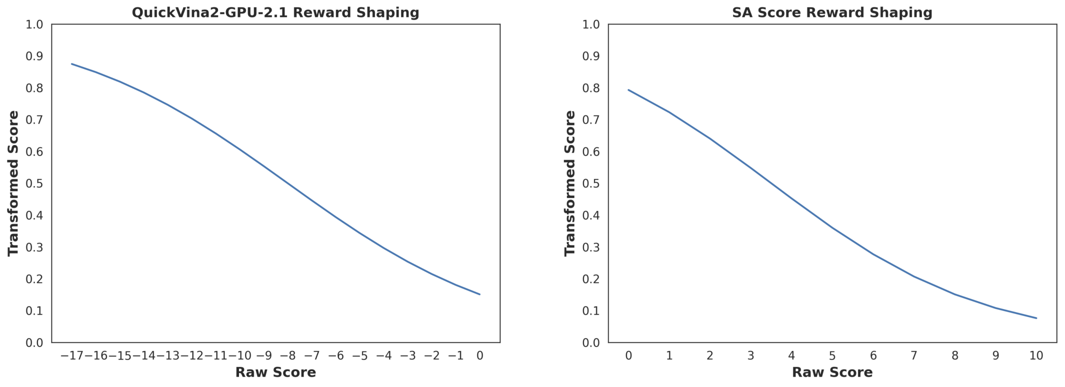


Figure H7: Saturn reward shaping functions for QuickVina2-GPU-2.1 and SA score.

(unclear how many replicates this was over). This is low and suggests the model is exploiting the docking algorithm as shown in Fig. 2. To show that *jointly* optimizing QED and docking is a *considerably* more difficult task, we first cross-reference the results for $R_{Double\ MPO}$ (Table 3) where the Modes and Yield are notably higher than $R_{All\ MPO}$. Next, we cross-reference the results when QED is *not* being optimized (Table 3 last row). The Yield is *much* higher (molecules with docking score < 10,000) but the QED values are similar to RGFN, which again, suggests the docking algorithm is being exploited.

Optimizing SA score *can* be a better allocation of computational resources. SA score[4] is correlated with AiZynthFinder solve rate[43]. In the main text Fig 4, we empirically demonstrate this, as 56 seconds of fine-tuning a pre-trained model that has *never* seen an AiZynthFinder solved molecule, results in a model that generates molecules almost all solvable. The natural next question is, would simply optimizing SA score be a better allocation of computational resources (as is commonly done)? Under the same wall time, many more queries to SA score can be made because it is computationally cheap. Correspondingly, we use the $R_{All\ MPO}$ objective function but omit AiZynthFinder (only docking, QED, and SA score) and run the ChEMBL and ZINC pre-trained models for 10,000 oracle calls (Table 3). Firstly, the wall time is similar to running 1,000 oracle calls of AiZynthFinder (cross-reference Table 1). Next, while a smaller fraction of the Modes are AiZynthFinder solvable, the raw number is higher than directly optimizing AiZynthFinder. This reinforces that post-hoc retrosynthesis model filtering is valid and is often what is done in practice[63]. Crucially, the actual percentage of AiZynthFinder solve rate may not *actually* matter. What matters is that a user can reasonably expect a generative model to generate molecules satisfying the objective function within the allotted oracle budget and/or wall time. In this specific example, it does not matter that Saturn-ZINC "only" has 55% solve rate when optimizing docking, QED, and SA score (Table 3). Running the 332 Modes through AiZynthFinder only took about 20 minutes (about 183/332 can be solved). A user would only care that in under 3 hours, 183 Modes were found that have low docking score, high QED, low SA score, and are AiZynthFinder solvable.

Finally, we wish to be prudent with making definitive statements about whether just optimizing SA score is strictly *better* than including a retrosynthesis model in the objective function. In this section alone, we have highlighted that different retrosynthesis models can have a large impact on wall time[79,80], where faster wall times would narrow the gap between SA score's wall time. Moreover, molecules deemed difficult to synthesize by SA score may actually be straightforward to synthesize. Retrosynthesis models have much more flexibility as the building block stock and reactions can be changed, whereas SA score was designed based on the fixed PubChem corpus[4]. One could even constrain the retrosynthesis model to only include building blocks and reactions that are available in-house, similar to what was done in a collaborative work involving Pfizer[81]. Thus, we leave a more thorough investigation regarding SA score optimization compared to various retrosynthesis models and search algorithms for future work. In this work, only the AiZynthFinder[11–13] retrosynthesis model was used, which leverages Monte Carlo Tree Search and ZINC building blocks.

H Saturn Reward Shaping

This section contains details on the reward shaping functions used such that the objective functions: $R_{RGFN}, R_{All\ MPO}, R_{Double\ MPO} \in [0, 1]$. Fig. H7 shows the functions for QuickVina2-GPU-2.1[59–61] and SA score[4]. QED[62] values were taken as is, and not subjected to reward shaping. AiZynthFinder[11–13] returns 0 for not solved and 1 for solved. Given a molecule, all oracle evaluations are aggregated via a weighted product and a single scalar value is returned as the reward:

$$R(x) = \left[\prod_i p_i(x)^{w_i}\right]^{\frac{1}{\sum_i w_i}} \tag{4}$$

x is a SMILES[82], i is the index of an oracle given many oracles (MPO objective), p_i is an oracle, and w_i is the weight assigned to the oracle (1 for all oracles in this work).

I GraphGA-augmented Experience Replay

Saturn[40] uses experience replay to enhance sample efficiency. GraphGA[68] can be applied on the replay buffer (stores the highest rewarding molecules generated so far) by treating the replay buffer as the parent population. Crossover and mutation operations then generate new molecules. For all the results in this work, activating the GA decreases the AiZynthFinder solve rate relative to no GA. This is because the generated molecules are not being sampled from the model itself (which is *learning* to generate AiZynthFinder solvable molecules). What is gained in return is diversity recovering (as found in the original Saturn[40] work). This can be advantageous since the RGFN[29] work defines **Discovered Modes** as the number of Modes (<-10 docking score) which also have < 0.5 Tanimoto similarity to every other mode. By activating the GA, more Modes are generally found, relative to no GA.

J Saturn Batch Generation

Saturn[40] generates SMILES[82] in batches of, at maximum, 16. Internally, there is an oracle caching mechanism such that repeat generated SMILES are not sent for oracle evaluation, and instead, the reward is retrieved from the cache. Saturn's sample efficiency comes from the local exploration of chemical space, such that, at adjacent epochs, identical SMILES can be generated. The effect is that at each generation epoch, sometimes only a few *new* (not generated before) SMILES are generated. In Fig. 4c, some batches have 0% solve rate by AiZynthFinder. These are batches that only have a few new SMILES that happen not to be solvable. If one new SMILES is generated, it being unsolvable equates to 0% solve rate.