Data-driven approaches for identifying hyperparameters in multi-step retrosynthesis

Annie M. Westerlund[a], Bente Barge[a,b], Lewis Mervin[c], Samuel Genheden[a*]

[a] Molecular AI, Discovery Sciences, R&D, AstraZeneca, Gothenburg, Sweden

[b]Hylleraas Centre for Quantum Molecular Sciences, Department of Chemistry, UiT The Arctic University of Norway, N9037 Tromsø, Norway

[c] Molecular AI, Discovery Sciences, R&D, AstraZeneca, Cambridge, UK

* Corresponding author e-mail: samuel.genheden@astrazeneca.com

Abstract

Multi-step retrosynthesis problem can be solved by a search algorithm, such as Monte Carlo tree search (MCTS). The performance of multistep retrosynthesis, as measured by a trade-off in search time and route solvability, therefore depends on the hyperparameters of the search algorithm. In this paper, we demonstrated the effect of three MCTS hyperparameters (number of iterations, tree depth, and tree width) on metrics such as Linear integrated speed-accuracy score (LISAS) and Inverse efficiency score which consider both route solvability and search time. This exploration was conducted by employing three data-driven approaches, namely a systematic grid search, Bayesian optimization over an ensemble of molecules to obtain static MCTS hyperparameters, and a machine learning approach to dynamically predict optimal MCTS hyperparameters given an input target molecule. With the obtained results on the internal dataset, we demonstrated that it is possible to identify a hyperparameter sets which outperform the current AiZynthFinder default setting and appeared optimal across a variety of target input molecules, both on proprietary and public datasets. The settings identified with the in-house dataset reached a solvability of 93% and median search time of 151s for the in-house dataset, and a 74% solvability and 114s for the ChEMBL dataset. These numbers can be compared to the current default settings which solved 85% and 73% during a median time of 110s and 84s, for in-house and ChEMBL, respectively.

Introduction

The problem of synthesizing novel chemical matter is a fundamental challenge in several areas of applied chemistry, including drug discovery and process chemistry[1]. The systematic approach presented by Corey, *retrosynthesis*, is now a standard part of textbooks on organic chemistry.[2] In multi-step retrosynthesis, the target molecule is iteratively broken down to smaller precursor molecules until the precursors are readily available from stock solutions. In recent times, software tools have been developed to aid chemists in this task. Such tools suggest how to break down a target molecule, and may also be used to predict the outcome of putative reactions (forward prediction) and under what conditions the reactions should be performed.[3,4] Although expert systems still have a significant impact on the field, many tools are now data-driven and exploit machine learning models.[5,6]

The multi-step retrosynthesis problem can be solved using a search algorithm which operates on a tree or graph. Although several search algorithms have been suggested, including Monte Carlo tree search (MCTS),[7,8] number-proof theory searches,[9,10,11] and A*-like algorithms,[12,13,14] it is still unclear if any algorithm is superior for the retrosynthesis task.[15,16] In short, the search algorithm proceeds by selecting a promising node in the search tree. This node is then expanded via a one-step retrosynthesis model (i.e. predicting precursors of the current molecule). An iteration is completed by updating a search tree statistic, which guides the selection of a promising node in the next iteration of the algorithm. In MCTS, for instance, the node for expansion is selected based on an upper confidence bound statistic, and an iteration is completed by backpropagating a reward obtained from a so-called roll-out process, which estimates the cost of the synthetic route.[17,18]

For the multi-step retrosynthesis field, an optimal algorithm would properly balance both speed and accuracy, resulting in a short search time and high accuracy. In a high-throughput context, for example, it is highly desirable to minimize the prediction time to enable queries of multiple compounds while decreasing energy usage. Both speed and accuracy are controlled by specified hyperparameters (or configuration settings). Therefore, it is desirable to find an optimal, or at least objectively satisfactory, set of hyperparameters for the multi-step retrosynthesis task. The speed-accuracy trade-off must be considered for any search algorithm and may be represented in an objective function using the response time from the search (T) versus the accuracy (in this case proportion of solved (PS) routes), respectively. The hyperparameters, which are tuned to maximize algorithmic performance, are often defined by the user, but can in general be automatically specified.[19,20] Examples of hyperparameters for multistep retrosynthesis algorithms include the width of the tree (the number of one-step retrosynthesis predictions to add to the search tree), and the maximum depth (a limit on the linear length of the synthetic route). Iterative algorithms, such as MCTS, often also require a priori specification of maximum number of iterations, and maximum elapsed compute time. Furthermore, specific search algorithms may have extra hyperparameters, e.g. a parameter in MCTS controlling the balance between exploitation and exploration. MCTS are used in a variety of fields and hyperparameters have been explored for a variety of MCTS implementations. The computational budget expressed as the number of iterations and the balance between exploitation and exploration have been most commonly been investigated,[17,18] but it is unlikely that the learnings from previous studies on other fields can be transferred to retrosynthesis, as the implementation of the MCTS algorithm is significantly different. Although hyperparameters are likely to influence the output of the multi-step retrosynthesis experiments, the process of selecting (or *optimizing)* them is largely unexplored in this field. For AiZynthFinder,[21] default values have been introduced based on limited manual explorations with a small number of compounds. As such, it is unclear if these values are optimal, or even performant, for the typical set of molecules that AiZynthFinder is routinely used to predict synthetic routes for. It should be noted that such an empirical approach to finding good hyperparameters is a common approach in MCTS research.[17] In principle, one could approach this in a naïve manner and employ the maximum number of iterations, maximum search tree depth, and maximum width affordable within a fixed computational budget. However, this tactic is approaching a systematic exploration of the search tree and it quickly becomes tedious for the search algorithm to effectively navigate the large search tree effectively.

In this paper, we first demonstrate the effect of three MCTS hyperparameters (number of iterations, tree depth, and tree width) on route solvability and search time, while exploring metrics that are tailored toward optimizing both aspects of a model. We then exploit three data-driven approaches for optimizing performance of multi-step retrosynthesis depending on these three hyperparameters. In the first approach, we obtain an optimal static hyperparameter configuration by exploiting Bayesian optimization over an ensemble of molecules in an AstraZeneca in-house (*AZ designs*) dataset. In the second approach, we perform a systematic grid search and visually pinpoint a static hyperparameter configuration from the experiments. In the third approach, we train machine learning models based on the initial experiments to dynamically predict optimal MCTS hyperparameters given an input target molecule. The obtained results and conclusions derived from the AZ design dataset are further validated on a ChEMBL target set.[22]

Methods

Compound datasets. We constructed two datasets of compounds to be used as targets for the retrosynthesis experiments. The first consisted of approximately 28,600 compounds designed by AstraZeneca chemists for in-house drug discovery projects, and this set will be referred to as *AZ design*. The second set, which will be referred to as *ChEMBL*, consisted of 50,000 compounds randomly sampled from the ChEMBL database[22] which have a molecular weight between 100 and 800 Da and a QED score[23] above 0.2. The tautomeric form of the compounds was determined by RDKit.[24] The AZ design dataset represents contemporary drug design and the type of molecules AiZynthFinder would typically be applied to. The ChEMBL dataset, on the other hand, is publicly available and offers an external test set for our study. By using these two datasets, we can investigate transferability of the MCTS hyperparameter searches between different sets, which may cover slightly different chemical spaces.

Retrosynthesis experiments. For each combination of MCTS hyperparameters, we ran retrosynthesis searches on each compound using the AiZynthFinder software.[21] The hyperparameters and their settings are shown in Table 1. Combining the different settings for hyperparameters in Table 1 gives rise to 27 different combinations and we performed retrosynthesis each of the 27 combinations for all target molecules in both compound sets. No time limit was specified. As an expansion policy, we use a one-step template-based retrosynthesis models previously explained in the literature, and we also apply a quick filter policy based on a binary classification model to remove the most unlikely predictions of the expansion policy.[25] For the *AZ designs,* we used filter and expansion policies trained on in-house data, and an in-house stock set.[26] For the *ChEMBL* compounds, we used policies trained on public reaction data,[27] and a stock that was a combination of ZINC[28] and eMolecules.[26]

***Table 1** – Hyperparameters varied in the retrosynthesis experiments*

Hyperparameter	**Settings**
Search tree width	10, 50, 100
Maximum search tree depth	3, 7, 12
Number of iterations	50, 100, 400

The performance of each set of MCTS hyperparameters was determined by the fraction of solved molecules and by the search time to find these *n* solutions, with a solution being defined as a synthetic route with all starting material existing in stock. This will be referred to as the *solvability*. The total search time was computed by multiplying the average iteration time by the number of iterations used for that experiment, whereas the time to a solution was defined as the average iteration time multiplied with the maximum number of iterations it took to find *n* solutions.

Retrosynthesis performance evaluation. We explored a few metrics previously reported in literature for scoring with speed–accuracy trade-offs[29]. Here we detail their definitions in terms of combining the search time, *T*, and proportion of solved targets, *PS*. The *inverse efficiency score*[30] (IES) was given by the median search time of solved targets (T_{solved}), divided by the proportion of solved targets (PS),

$$IES = \frac{med(T_{solved})}{PS}.$$

The *rate-correct score*[31] (RCS) was defined as the number of solved (N_{solved}) over the summed search times, T_i,

$$RCS = \frac{N_{solved}}{\sum_i T_i}.$$

Both *IES* and *RCS* perform a direct the speed-accuracy weighting. The *linear integrated speed–accuracy score*[32] (LISAS) instead accounts for the magnitude changes in accuracy and speed by scaling with their corresponding standard deviations,

$$LISAS = med(T_{solved}) + \frac{\sigma(T_{solved})}{\sigma(1-PS)}(1 - PS).$$

Datasets for training, validation and testing. In this section we discuss the data driven approaches employed, namely the Bayesian hyperparameter optimization approach using the publicly available optimization software framework, Optuna[33], and a dynamic hyperparameter prediction with machine learning models (see the following two paragraphs for details). The initial Optuna hyperparameter parameter search was derived from a smaller distribution of two random subsets of 1000 molecules from the *AZ designs* and ChEMBL datasets. We selected these smaller subsets because optimization over the full datasets would be intractable due to the relatively long retrosynthesis search time. For training machine learning models, we instead needed a larger subset from which the models could learn. The AZ design compound set was therefore divided into an 80% training set, 10% validation set and 10% test set, using a stratified split over compounds. The stratification was done based on which of the 27 combinations of the three hyperparameters resulted in at least one solved route in the shortest amount of time.

The dataset split used for the machine learning models differed from the one used for Optuna. Consequently, the machine learning test set and Optuna test sets differed as well. To allow for proper comparison of the different approaches, the results from both Optuna and machine learning models were evaluated on a shared test set of AZ design targets. This test set included the targets in the machine learning test set, excluding the few test set targets which were used in the Optuna optimization. Furthermore, the results were evaluated on the ChEMBL Optuna test set of 49,000 targets. The specific datasets, as well as their splits and sizes, are listed in Table S1 for clarity.

Optuna for Bayesian hyperparameter optimization. Optuna[34] (version 3.1.0) was used to perform two searches of the hyperparameter space using a random subset of 1000 molecules from the *AZ designs* and ChEMBL datasets, respectively (Table S1). Trials were optimized to minimize the LISAS for the subset dataset. The Tree-structured Parzen Estimator (TPE) was used to explore the three hyperparameters, with a minimum and maximum value of 5 to 150 for the width, 2 and 15 for the depth and 50 to 400 for the max iterations specified, respectively. The total number of trials was set to 30, with the number of start-up trials set to 10, to ensure that sufficient random sampling of the hyperparameter space was performed to avoid reaching a local minimum. The performance of the optimal hyperparameter configuration was subsequently evaluated on the full set of test target inputs, as well as the part of the test set shared with machine learning models (see the following paragraph).

Machine learning for dynamic hyperparameter prediction. We trained a nearest neighbor (*k*NN) model, a random forest model[35] and a ChemProp model[36] on each compound set. For all models, the three MCTS hyperparameters (width, depth and number of iterations) were treated as individual multiclass tasks, i.e. we framed the modeling as a multitask-multiclass problem. The input to *k*NN and random forest consisted of Morgan-like fingerprints[37] of radius 2 and length 2024, calculated with RDKit[24], while the input to ChemProp consisted of the graph representation of the compounds. Based on the retrosynthesis experiments, we assigned binary labels to each MCTS hyperparameter such that a positive label corresponded to the hyperparameter being in the most efficient configuration: If a configuration produced a solved route in the shortest time across experiment hyperparameters, the corresponding training labels were set to 1, otherwise 0. The *k*NN and random forest models were fitted using scikit-learn version 1.2.0[38]. The *k*NN model used five neighbors and the Jaccard distance between Morgan-like fingerprints while the random forest used 100 decision trees and "balanced subset" as class weight. For the random forest, we trained one model per MCTS hyperparameter. Furthermore, ChemProp version 1.5.2 was used with a cross-entropy loss. Since the choice of ChemProp network parameters are reported to influence performance, we conducted an additional hyperparameter optimization with the built-in standard settings of the ChemProp hyperparameter optimization procedure for 100 iterations. Each model was fitted on the training set (Table S1). The predefined training, validation and test sets (Table S1) were supplied to ChemProp as separate sets to ensure the same split across models. The fitted models were then evaluated on the test set using two metrics that were computed separately for each task (each MCTS hyperparameter). The first being the area under the curve of the receiving operator characteristic using one-versus-all and weighted average (to account for class-imbalanced data), and the second being the multiclass accuracy.

Results and discussion

Selection of hyperparameters greatly influences retrosynthesis solvability and search time

As a first step, we performed retrosynthesis experiments on all AZ design targets with AiZynthFinder using 27 different combinations of the three MCTS hyperparameters: tree width, depth, and number of iterations. There are other hyperparameters that could have a significant effect on the route predictions, but we limit ourselves to the three that we judged as having the most significant effect, to reduce the complexity of the optimization search problem. In the future, it could also be of interest to tune the balance between exploitation and exploration. Figure 1a shows the percentage of targets for which AiZynthFinder finds at least one route, while Figure 1b shows the search time. As evident, the choice of hyperparameters affects both the percentage of solved targets, and the median search time. The fewest targets were solved with the set {*d*=3, *w*=10, *i*=50} (53%) and the most targets were solved with the set {*d*=12, *w*=100, *i*=400} (87%). Moreover, 88% of the targets were solved with at least one setting, and the current default set {*d*=7, *w*=50, *i*=100} solved 75% of the targets, Figure 1a. We also noticed large deviations in the median search time depending on the hyperparameters. More iterations and wider or deeper search trees required longer searches to find a solution. In particular, the best resolving set {*d*=12, *w*=100, *i*=400} on average required a longer search time (719 seconds (s)) than the current default set (113s) and the worst performing set (19s). We notice similar wide-spread behavior on the ChEMBL target dataset, Figure S1. Specifically, the fewest ChEMBL targets were solved by {d=3, w=10, i=50} (62%) while most were solved with the most generous setting {d=12, w=100, i=400} (79%). The search time covered the range 10s-674s. These numbers can be compared to what is achieved with the current default (73% / 84s). This wide range of outcomes demonstrates the value of devising a good strategy for finding MCTS hyperparameters by considering a trade-off in solvability and search time.

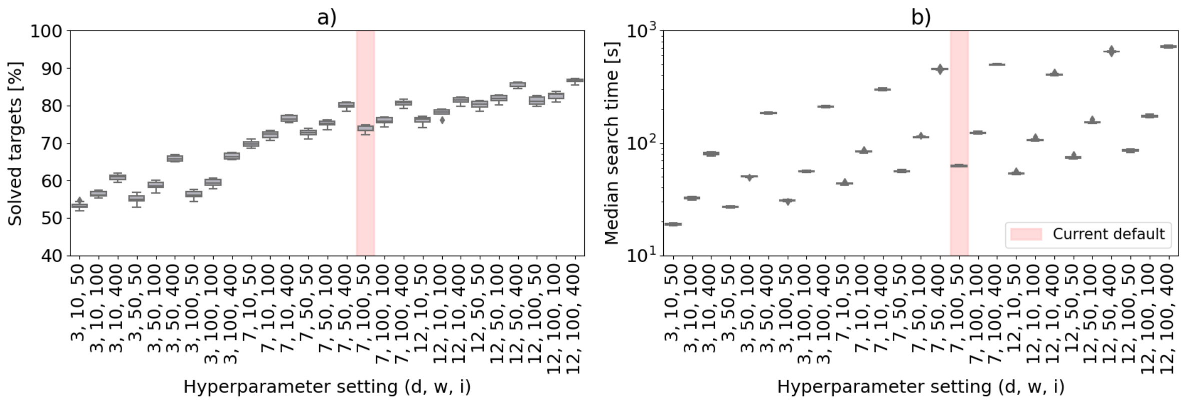

***Figure 1** – a) Percentage of solved targets and b) median search time in seconds for best solution for the 27 hyperparameter sets. The analysis was done on the full set of AZ design targets. The x-axis indicates the hyperparameters in the order of d (depth), w (width), and i*

(iterations). The boxes reflect the distribution after dividing the data into 10 batches. The pink shade marks the current default configuration in AiZynthFinder.

Optimizing static hyperparameters over the ensemble of molecules

To obtain optimal static MCTS hyperparameters, we exploited Bayesian optimization using an objective function which takes both search time and solvability (finding one solution) into account. We used the linear integrated speed-accuracy score (LISAS) for this purpose. Figure 2a-c shows the performance on the targets in the test set, as evaluated by four different metrics: percentage of solved targets vs. a) LISAS, b) IES, and c) RCS. Together with the optimized hyperparameter performance, we also evaluated the performance of the 27 combinations of hyperparameters in the experiment sets (see Figure 1 for configurations) using the various metrics on the test set, Figure 2 a-c).

Figure 2d shows the median search times against the percentage of solved targets. Although the hyperparameters found by Optuna using the LISAS metric resulted in a short median search time (22s), the percentage of solved targets is also low (53%). This indicates that the chosen metric puts too much weight on search time compared to solvability and may therefore not be suitable for optimizing MCTS hyperparameters. We observed similar issues with both the inverse efficiency score and the rate-correct score metrics (Figure 2b-c)). Furthermore, when observing the optimization landscapes of *median search time* and *percentage of solved targets* as a function of the MCTS hyperparameters (Figure S2), we note that finding a good weight between *median search time* and *percentage of solved targets* is nontrivial and subjective. However, using the results in Figure S3 (Optuna training set), we can perform a visual grid search to identify a set of MCTS hyperparameters among the experiment sets which qualitatively outperforms both the Optuna-found hyperparameters and the current default set for finding 1, 5, and 10 solutions, namely {*d*=12, *w*=50, *i*=100}, Figure S3 d-f). We call this set of hyperparameters the "experiment best" set. The findings of this set translate to both the test set (Figure 2 and S4) and the ChEMBL compounds (Figure S5). For the AZ design test set, this configuration resulted in a median search time of 151 seconds, and a solvability of 93% for finding 1 solution. Furthermore, this specific configuration was able to retain high solvability for 5 and 10 solutions, Figure 2e-f).

Regarding the optimization approach, one may argue that finding one solution for each target is insufficient in practice due to inaccuracies of the current retrosynthesis methods. Among other things, there is a lack of forward synthesis validation and reaction conditions. To mitigate such inaccuracies, we can instead request multiple solutions from AiZynthFinder. These solutions can then be evaluated by a chemist before initiating wet-lab synthesis. Clearly, if we select too restrictive MCTS hyperparameters, we cannot expect to find a diverse set of solutions. Hence, we wonder if the current analysis considering one solution is sufficient. Figure 2d-f display the percentage of solved targets with at least d) 1, e) 5 and f) 10 solutions as a function of median search time for different static hyperparameter sets. Although more extensive tree searches naturally produce more solutions, we note that the overall hyperparameter performance remains quite similar regardless of whether we consider finding one, five or ten solutions. It should be noted that we do not equate solvability with the quality of the produced routes. However, there is a current lack of an agreed metric of route quality to can be computed in a high-throughput fashion.[15] Therefore, we focus on the solvability as the most important metric of the success of the multistep retrosynthesis, which also is aligned with the fact that such tools are primarily used as idea generators. Given that the route prediction often produces more than one route, it is likely that some of the predictions are useful for further optimization by a domain expert.

Taken together, the MCTS hyperparameters optimized over the ensemble of molecules indeed decreased the overall search time, but also the solvability of most targets. Instead, we could qualitatively pinpoint an MCTS configuration which outperformed both the current default and the optimized configurations. Nonetheless, one could hypothesize that a dynamic approach for selecting the hyperparameters based on molecular features may further optimize both solvability ability and search time. We therefore decided to also explore such a dynamic approach using machine learning to predict hyperparameters.

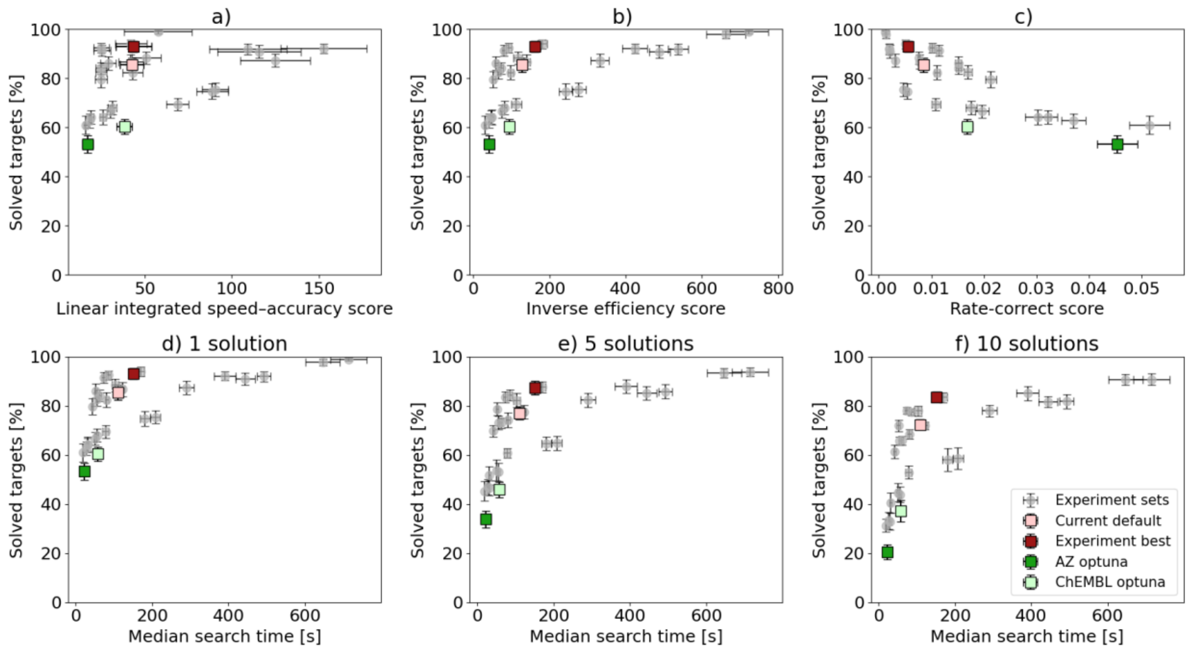

***Figure 2** – a-c) Percentage of solved targets (1 solution) vs. a) LISAS, b) IES, and c) RCS. d-f) Percentage of solved targets of the test set of AZ designs as a function of median search time. Percentage corresponds to finding at least d) 1 solution, e) 5 solutions and f) 10 solutions. The error bars show the standard deviation across 10 splits. AZ and ChEMBL Optuna refer to Optuna applied to the AZ design and ChEMBL datasets, respectively.*

Predicting dynamic hyperparameters on a molecule-specific basis

To generate dynamic (i.e. molecule-specific) MCTS hyperparameters, we used a machine learning approach where optimal hyperparameters were predicted given an input molecule. We implemented and evaluated three models for this purpose: 1) a simple k-nearest neighbors (*k*NN) model, 2) three random forest models each trained on a hyperparameter (task) while accounting for class imbalance, and 3) the deep-learning neural network ChemProp.[36] The models were evaluated by accuracy and (class-weighted) area under the receiver-operating characteristic curve (AUC-ROC) (see Table 2).

***Table 2** – Performance metrics for individual tasks*

Model	**Hyperparameter**	**AUC-ROC (unweighted)**	**AUC-ROC (weighted)**	**Accuracy**
*k*NN	*d*	0.79	0.69	0.73
*k*NN	*w*	0.80	0.62	0.74
*k*NN	*i*	0.91	0.59	0.88
Random forest	*d*	0.81	0.72	0.75
Random forest	*w*	0.81	0.64	0.75
Random forest	*i*	0.91	0.60	0.88
ChemProp	*d*	0.80	0.73	0.73
ChemProp	*w*	0.81	0.61	0.74
ChemProp	*i*	0.92	0.59	0.89

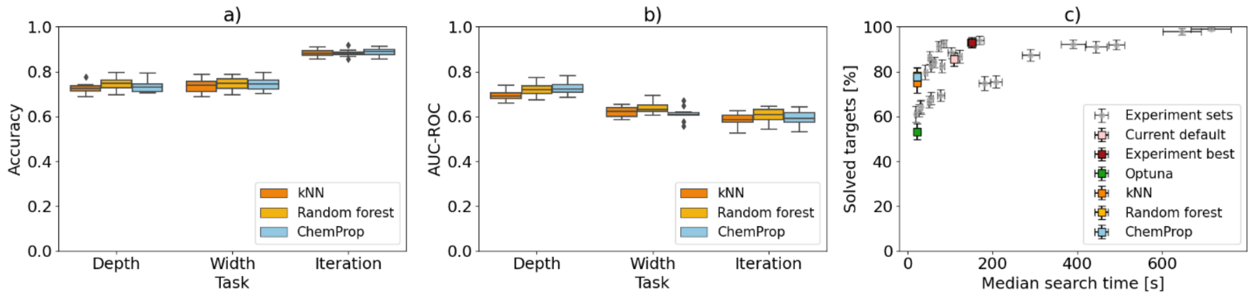

***Figure 3** - Performance of the dynamical MCTS hyperparameter predictors. a) Accuracy per task, b) (class-weighted) AUC-ROC, and c) percentage of solved targets vs. median search time comparing to the static hyperparameters. Error bars denote the standard deviation across 10 splits.*

Can we now use these multi-task models together to predict MCTS hyperparameters for retrosynthesis searches? If so, the models should not only predict each hyperparameter correctly, but also the combination. For the targets in the test set, we compare the models with the current default hyperparameters ({d=7, w=50, i=100}) and the best performing hyperparameter set ({d=12, w=100, i=400}). The three dynamic models (kNN, random forest and ChemProp), albeit providing hyperparameters yielding short search times, are outperformed when it comes to the number of targets that are solved, Figure 3c. While the current default finds solutions to 85% of the targets in the test set, the dynamically suggested configurations only find solutions to 75-78% of the targets. The drop in performance comes with a substantial decrease in search time, from an average of 109 seconds with the default setting to about 22 seconds for the machine learning models. Although not able to outperform the current default setting, the dynamic approach indeed appeared more successful than the static optimization approach, Table 3. Provided the same amount of time (22-23s), the machine learning models improve solvability compared to the configuration proposed by Optuna.

Table 3 *– Median time and solvability of different strategies for selecting MCTS hyperparameters. AZ Optuna refers to Optuna applied to the AZ design dataset.*

MCTS hyperparameter strategy	**Configuration (*d, w, i*)**	**Median time**	**Solvability**
Current default	(7, 50, 100)	110	85
Experiment best	(12, 50, 100)	151	93
AZ Optuna	(2, 108, 51)	22	53
*k*NN	*dynamic*	22	75
Random forest	*dynamic*	23	78
ChemProp	*dynamic*	23	78

Remarks on difficulties in predicting dynamic hyperparameters

The difficulty in using basic machine learning to dynamically predict optimal hyperparameters may be caused by two main contributors. First, the "true optimal" hyperparameter values were selected such that the search time for obtaining one solution was minimized. However, the results were quite noisy. Specifically, many settings have similar search times and yield similar frequency of solved targets. Figure 4a shows the minimum search time over all hyperparameter sets that solves the different compounds. For more than half of the targets, the minimum search time is less than five seconds, and for most targets it is less than a minute, Figure 4a. Furthermore, the difference in search time between the two fastest sets is less than ten seconds for most targets, Figure 4b. The ten fastest settings are typically *not* more than 100s slower than the fastest one. This implies that modeling the relationship between input structure and best hyperparameters can be difficult as there are potentially multiple suitable solutions. In addition to this, we did not observe any clustering of similar molecules in any of the "true optimal" hyperparameter settings, number of solutions, or search time in UMAP space[39] (Figure S6). One could argue that any time the model predicts sub-optimal hyperparameters, there is a chance that the predicted parameters are good-enough. However, we note that the "experiment best" configuration still outperform the dynamic parameters when evaluating over the ensemble of molecules, Figure 2 and 3. In other words, we still prefer using the identified static hyperparameters over the dynamic.

The second reason for the difficulty in predicting dynamic parameters may reside in the imbalanced training data, Figure 5. However, it should be noted that our balanced random forest model was not able to accurately predict molecule-specific hyperparameters either. This hints that the absence of correlation between molecular similarity and MCTS hyperparameter configuration may be the key issue.

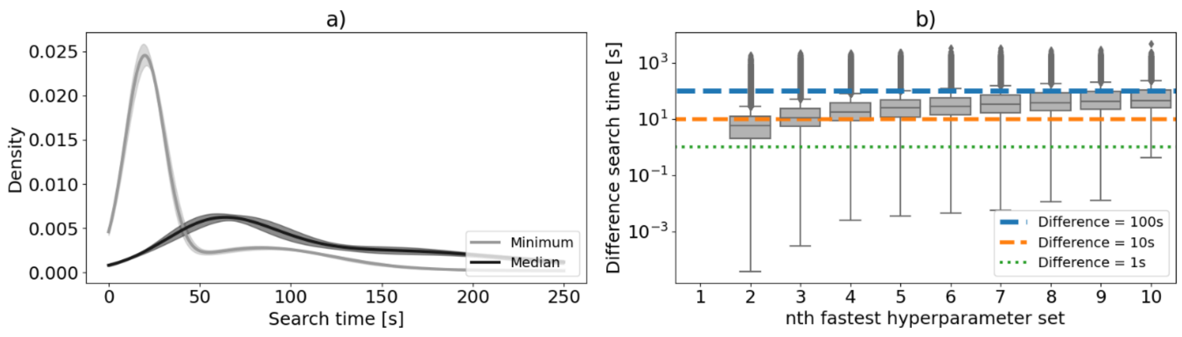

***Figure 4** – a) The distribution of minimum and median search time for the AZ design targets over all hyperparameter sets. The shaded region denotes the standard deviation error across 10 splits of the data. b) The distribution of the difference between the fastest hyperparameter sets and the nth fastest hyperparameter sets for the AZ design targets.*

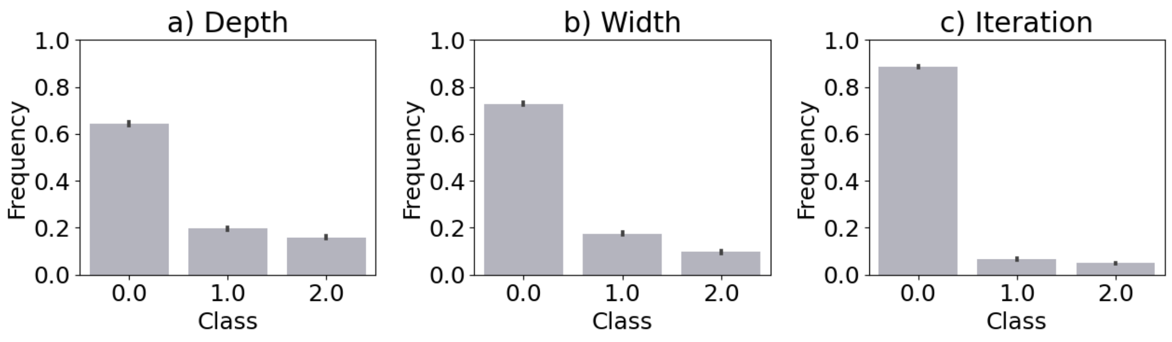

***Figure 5** – Percentage of AZ design targets which are solved fastest using each hyperparameter set. The x-axis indicates the hyperparameters in the order of d (depth), w (width), and i (iterations).*

Conclusions

The success of multi-step retrosynthesis is highly dependent on the selection of MCTS hyperparameters (i.e. the configuration). The current default configuration in AiZynthFinder was determined by manually running the retrosynthesis prediction on a small number of test targets and examining the results.

In this work, we investigated data-driven approaches and strategies for improving the default MCTS hyperparameter configuration (tree depth, width, and number of iterations). We specifically evaluated three approaches for optimizing on search time and solvability simultaneously: 1) a static approach where configurations were selected by optimizing over the ensemble of molecules, 2) a (static) systematic grid search over configurations, and 3) a dynamic approach where the hyperparameters were predicted for each specific molecule. By comparing between these three strategies and benchmarking against the current default configuration, we found a new set of MCTS hyperparameters which outperformed the current default configuration.

Surprisingly, the systematic grid search proved most successful among the three suggested strategies. As such, a static approach for selecting MCTS hyperparameters may be more reliable than the suggested dynamic approach. We attributed the difficulties of the dynamic strategy to three main issues: 1) there may exist many well-suited configurations for each molecule, 2) the training set was class-imbalanced, and 3) we observed a seemingly missing correlation between molecular similarity and MCTS configuration. Hence, future studies are needed to investigate a different treatment of the dynamic approach, as opposed to the current classification model requiring the discretization of the data into "optimal" hyperparameter classes.

References

[1] Oliveira JCA, Frey J, Zhang S-Q, Xu L-C, Li X, Li S-W, Hong X, Ackermann L (2022) When machine learning meets molecular synthesis. Trends in Chem, 4:10, 863-885, https://doi.org/10.1016/j.trechm.2022.07.005

[2] Corey EJ, Todd Wipke W (1969) Computer-assisted design of complex organic syntheses. Science (80-) 166:178–192. https://doi.org/10.1126/science.166.3902.178

[3] Johansson S, Thakkar A, Kogej T, et al (2019) AI-assisted synthesis prediction. Drug Discov. Today Technol. 32–33:65–72

[4] Schwaller P, Vaucher AC, Laplaza R, et al (2022) Machine intelligence for chemical reaction space. Wiley Interdiscip Rev Comput Mol Sci e1604. https://doi.org/10.1002/WCMS.1604

[5] Aspuru-Guzik A, Lindh R, Reiher M (2018) The Matter Simulation (R)evolution. ACS Cent Sci 4:144–152. https://doi.org/10.1021/acscentsci.7b00550

[6] Mervin L, Genheden S, Engkvist O (2022) AI for drug design: From explicit rules to deep learning. Artif Intell Life Sci 2:100041. https://doi.org/10.1016/J.AILSCI.2022.100041

[7] Segler MHS, Preuss M, Waller MP (2018) Planning chemical syntheses with deep neural networks and symbolic AI. Nature 555:604–610. https://doi.org/10.1038/nature25978

[8] Thakkar A, Kogej T, Reymond JL, et al (2020) Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain. Chem Sci 11:154–168. https://doi.org/10.1039/c9sc04944d

[9] Heifets A, Jurisica I (2012) Construction of new medicines via game proof search. In: Proceedings of the National Conference on Artificial Intelligence. pp 1564–1570

[10] Kishimoto A, Buesser B, Chen B, Botea Eaton A (2019) Depth-First Proof-Number Search with Heuristic Edge Cost and Application to Chemical Synthesis Planning

[11] Shibukawa R, Ishida S, Yoshizoe K, et al (2020) CompRet: A comprehensive recommendation framework for chemical synthesis planning with algorithmic enumeration. J Cheminform 12:52. https://doi.org/10.1186/s13321-020-00452-5

[12] Chen B, Li C, Dai H, Song L (2020) Retro*: Learning retrosynthetic planning with neural guided A* search. In: 37th International Conference on Machine Learning, ICML 2020. pp 1586–1594

[13] Klucznik T, Mikulak-Klucznik B, McCormack MP, et al (2018) Efficient Syntheses of Diverse, Medicinally Relevant Targets Planned by Computer and Executed in the Laboratory. Chem 4:522–532. https://doi.org/10.1016/j.chempr.2018.02.002

[14] Jeong J, Lee N, Shin Y, Shin D (2021) Intelligent generation of optimal synthetic pathways based on knowledge graph inference and retrosynthetic predictions using reaction big data. J Taiwan Inst Chem Eng 000:. https://doi.org/10.1016/j.jtice.2021.07.015

[15] Genheden S, Bjerrum E (2022) PaRoutes: towards a framework for benchmarking retrosynthesis route predictions. Digit Discov 1:527–539. https://doi.org/10.1039/D2DD00015F

[16] Tripp A, Maziarz K, Lewis S, et al (2022) Re-Evaluating Chemical Synthesis Planning Algorithms.

[17] Świechowski M, Godlewski K, Sawicki B, Mańdziuk J (2023) Monte Carlo Tree Search: a review of recent modifications and applications. Artif Intell Rev 56:2497–2562. https://doi.org/10.1007/S10462-022-10228-Y/FIGURES/11

[18] Kemmerling M, Lütticke D, Schmitt RH (2023) Beyond Games: A Systematic Review of Neural Monte Carlo Tree Search Applications ArXiv http://arxiv.org/abs/2303.08060

[19] Volkamer A, Riniker S, Nittinger E, Lanini J, Grisoni F, Evertsson E, Rodriquez-Perez R, Schneider N (2023) Machine learning for small molecule drug discovery in academia and industry. Artif Int Life Sci. 3, 100056, https://doi.org/10.1016/j.ailsci.2022.100056

[20] Westerlund AM, Hawe JS, Heinig M, Schunkert H (2021) Risk Prediction of Cardiovascular Events by Exploration of Molecular Data with Explainable Artificial Intelligence. Int J Mol Sci. 22(19). 10291.

[21] Genheden S, Thakkar A, Chadimová V, et al (2020) AiZynthFinder: a fast, robust and flexible open-source software for retrosynthetic planning. J Cheminform 12:70

[22] Gaulton A, Bellis LJ, Bento AP, et al (2012) ChEMBL: A large-scale bioactivity database for drug discovery. Nucleic Acids Res 40:D1100. https://doi.org/10.1093/nar/gkr777

[23] Bickerton GR, Paolini G V., Besnard J, et al (2012) Quantifying the chemical beauty of drugs. Nat Chem 4:90–98. https://doi.org/10.1038/nchem.1243

[24] RDKit: Open-source cheminformatics, http://www.rdkit.org.

[25] Genheden S, Engkvist O, Bjerrum, E J. (2020) A Quick Policy to Filter Reactions Based on Feasibility in AI-Guided Retrosynthetic Planning. *ChemRxiv* https://doi.org/10.26434/CHEMRXIV.13280495.V1

[26] Genheden S, Norrby PO, Engkvist O (2022) AiZynthTrain: Robust, Reproducible, and Extensible Pipelines for Training Synthesis Prediction Models. J Chem Inf Model. https://doi.org/10.1021/ACS.JCIM.2C01486/SUPPL_FILE/CI2C01486_SI_001.ZIP

[27] Lowe D, Chemical reactions from US patents, 1976–Sep 2016, https://figshare.com/articles/Chemical_reactions_from_US_patents_1976-Sep2016_/ 5104873

[28] Sterling T, Irwin JJ (2015) ZINC 15 - Ligand Discovery for Everyone. J Chem Inf Model 55:2324–2337. https://doi.org/10.1021/acs.jcim.5b00559

[29] Liesefeld HR, Janczyk M (2018) Combining speed and accuracy to control for speed-accuracy trade-offs, Behavior Res. Methods, 51:40-60, https://doi.org/10.3758/s13428-018-1076-x

[30] Townsend JT, Ashby FG (1983) Stochastic modelling of elementary psychological processes. New York, NY: Cambridge University Press.

[31] Woltz DJ, Was CA (2006) Availability of related long-term memory during and after attention focus in working memory, Memory & Cognition, 34, 668-684, doi:https://doi.org/10.3758/

[32] Vandierendonck A (2017) A comparison of methods to combine speed and accuracy measures of performance. Behav Res Methods, 49, 653-673, doi:https://doi.org/10.3758/s13428-016-0721-5

[33] Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: A Next-generation Hyperparameter Optimization Framework. Proc 25th ACM SIGKDD Int Conf Knowledge Disc Data Mining, 2623-2631. https://doi.org/10.1145/3292500.3330701

[34] Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: A Next-generation Hyperparameter Optimization Framework. Proc 25th ACM SIGKDD Int Conf Knowledge Disc Data Mining, 2623-2631. https://doi.org/10.1145/3292500.3330701

[35] Breiman L (2001) Random forests. Mach Learn 45:5–32. https://doi.org/10.1023/A:1010933404324/METRICS

[36] Yang K, Swanson K, Jin W, et al (2019) Analyzing Learned Molecular Representations for Property Prediction. J Chem Inf Model 59:3370–3388. https://doi.org/10.1021/ACS.JCIM.9B00237/ASSET/IMAGES/LARGE/CI9B00237_0020.JPEG

[37] Rogers D, Hahn M (2010) Extended-connectivity fingerprints. J Chem Inf Model 50:742–754. https://doi.org/10.1021/CI100050T/ASSET/IMAGES/LARGE/CI-2010-00050T_0017.JPEG

[38] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res, 12, 2825-2830

[39] McInnes et al., (2018). UMAP: Uniform Manifold Approximation and Projection. J Open Source Softw, 3(29), 861. https://doi.org/10.21105/joss.00861

Supplementary Information

***Table S1** – Datasets and splits used in study*

Dataset	Total size	Optuna train. set	Optuna test set	ML train. set	ML val. set	ML test set	Shared test set
AZ design	28586	1000	27586	19913	2489	2504	2416
ChEMBL	50000	1000	49000	-	-	50000	49000

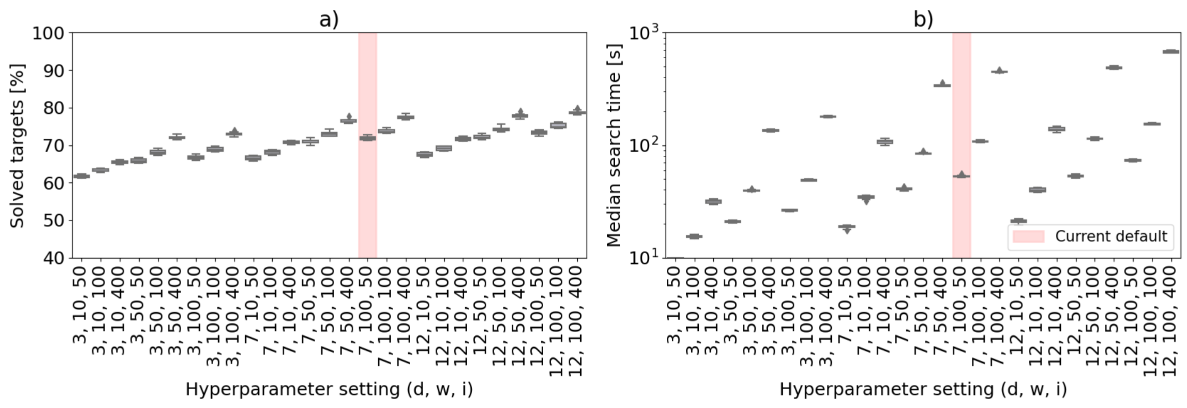

***Figure S1 -** ChEMBL targets. a) Percentage of solved targets and b) median search time in seconds for best solution for the 27 hyperparameter sets. The x-axis indicates the hyperparameters in the order of d (depth), w (width), and i (iterations). The boxes reflect the distribution after dividing the data into 10 batches. The pink shade marks the current default configuration in AiZynthFinder.*

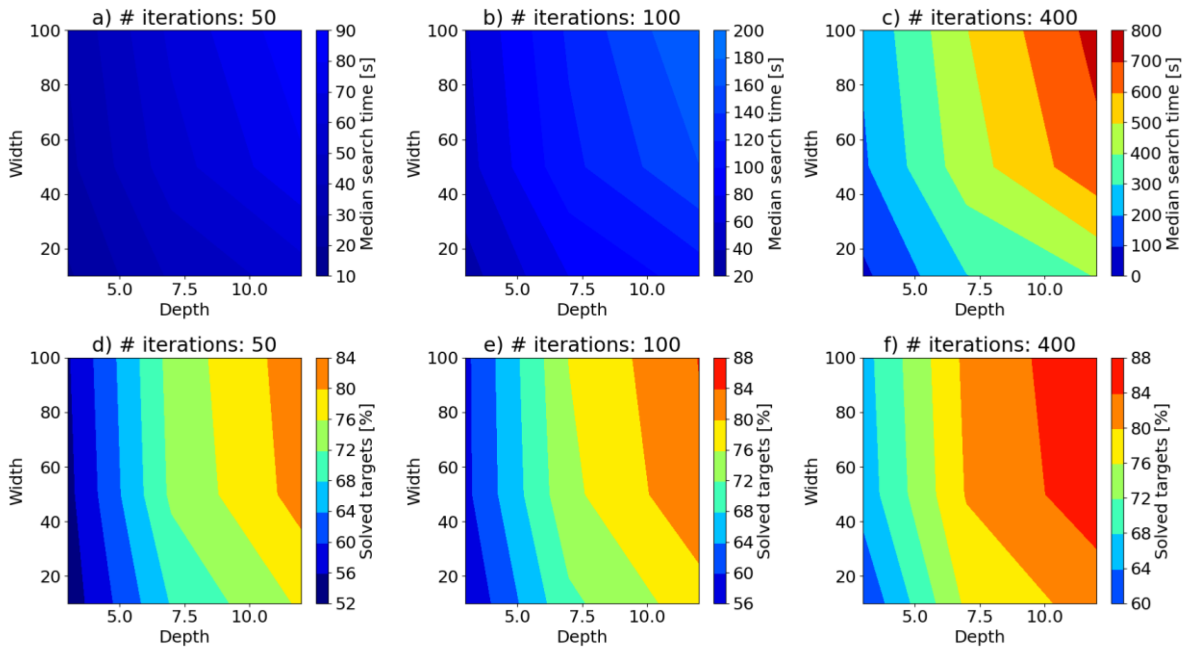

Figure S2 - AZ Optuna training set. *a-c) Optimization landscape over median search time as a function of the three hyperparameters. d-f) Optimization landscape over percentage of solved targets as a function of the three hyperparameters. AZ Optuna refers to Optuna applied to the AZ design dataset.*

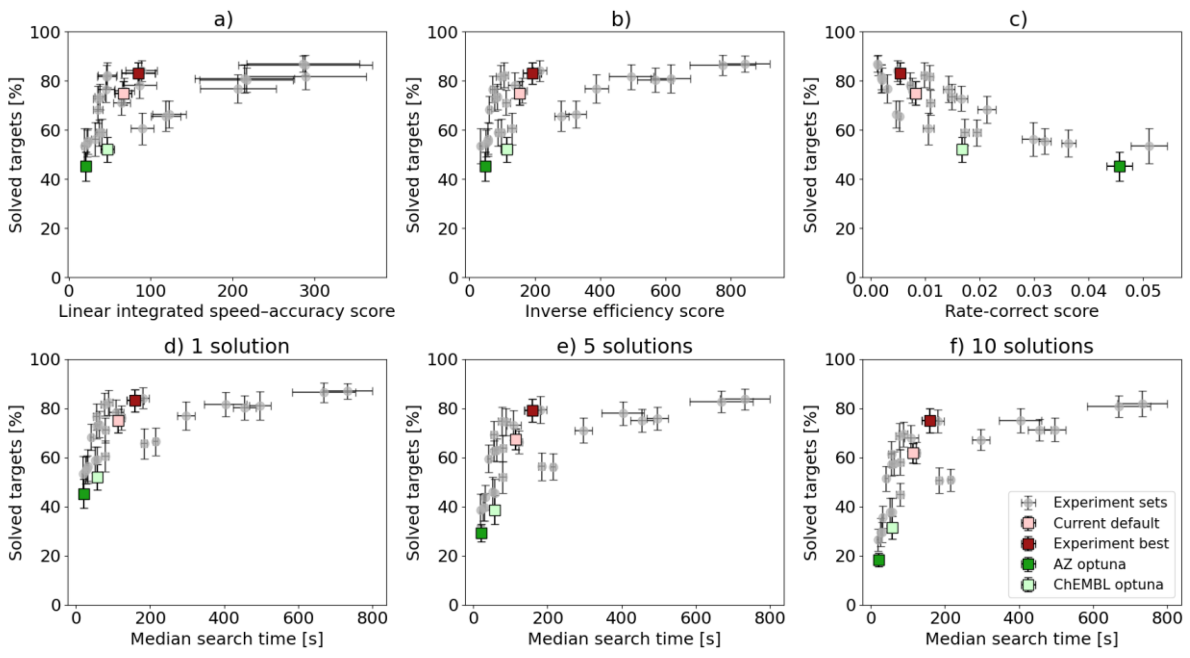

Figure S3 - AZ Optuna training set. *a-c) Percentage of solved targets (1 solution) vs. a) LISAS, b) IES, and c) RCS. d-f) Percentage of solved targets as a function of median search time. Percentage corresponds to finding at least d) 1 solution, e) 5 solutions and f) 10 solutions. The error bars show the standard deviation across 10 splits. AZ Optuna refers to Optuna applied to the AZ design dataset.*

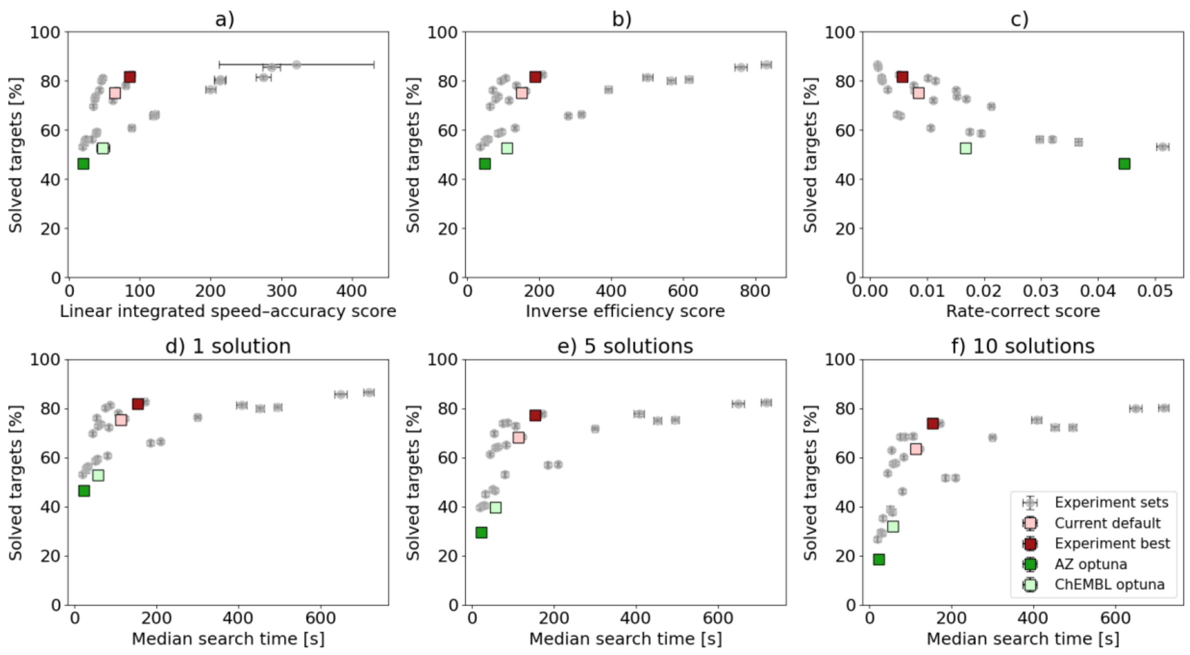

Figure S4 Full AZ Optuna test set. *a-c) Percentage of solved targets (1 solution) vs. a) LISAS, b) IES, and c) RCS. d-f) Percentage of solved targets as a function of median search time. Percentage corresponds to finding at least d) 1 solution, e) 5 solutions and f) 10 solutions. The error bars show the standard deviation across 10 splits. AZ Optuna refers to Optuna applied to the AZ design dataset.*

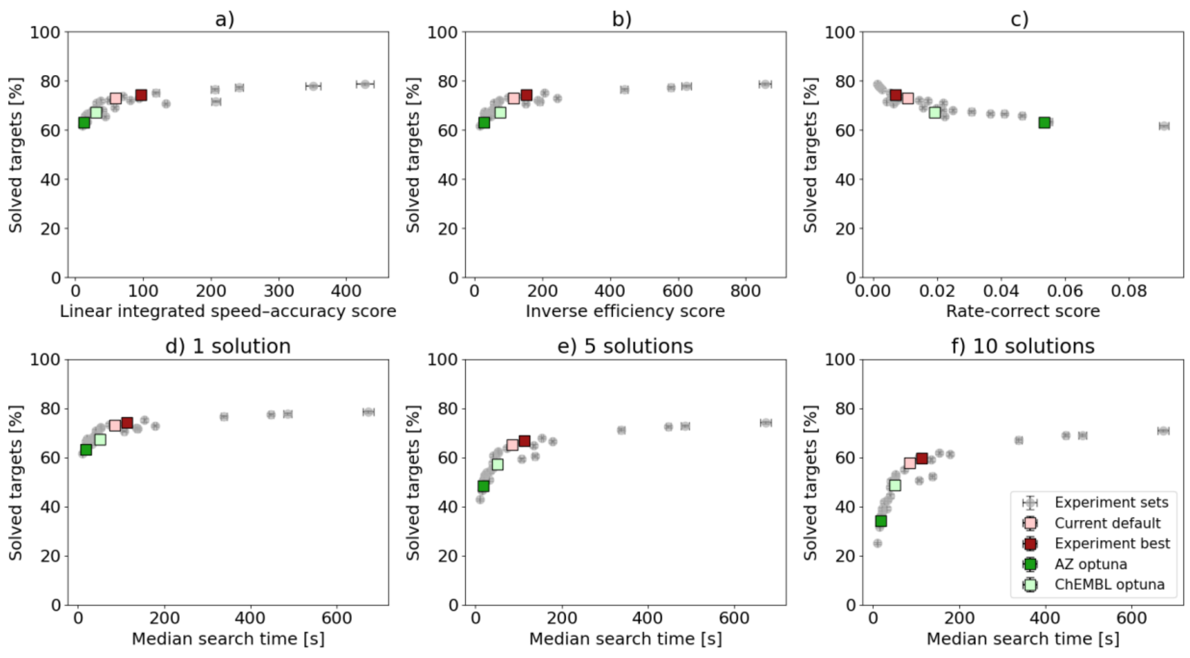

Figure S5 ChEMBL test set targets. *a-c) Percentage of solved targets (1 solution) vs. a) LISAS, b) IES, and c) RCS. d-f) Percentage of solved targets as a function of median search time. Percentage corresponds to finding at least d) 1 solution, e) 5 solutions and f) 10 solutions. The error bars show the standard deviation across 10 splits.*

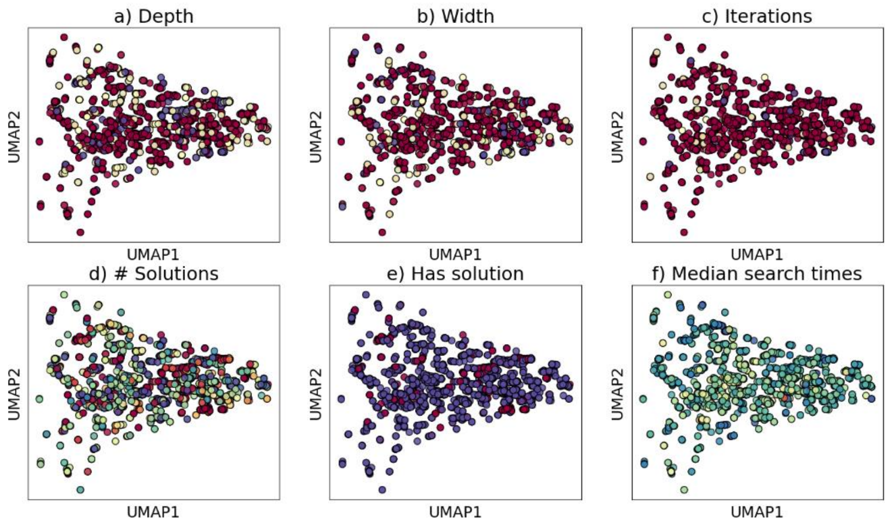

Figure S6 *UMAP projections of the molecular fingerprints (with Jaccard similarity metric). Each point is colored by a) optimal depth, b) optimal width, c) optimal number of iterations, d) number of found solutions, e) whether a solution was found (1) or not (0), f) median search times. The colormap maps values from red (low) to yellow (mid) to blue (high).*