universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

„Molecular Informatics of Next Generation Macrocycles“

verfasst von / submitted by

Christian Permann, MSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree

Doktor der Naturwissenschaften (Dr.rer.nat.)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 796 610 449
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet:	Pharmacy
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Thierry Langer

Acknowledgements

First of all, I would like to thank my supervisor Univ.-Prof. Mag. Dr. Thierry Langer for the supervision of this thesis. The open communication and the discussions we had were especially useful in the process of developing new ideas and methodologies. I look forward to future projects and collaborations, where I expect that we will create and discover a lot of novel and interesting concepts.

It is also important to me to express my gratitude for the support I received from my fiancée Rebecca. Our spare time spent together gave me the opportunity to wind down from work and keep motivated in the long run.

I also want to thank my parents Eva Maria and Otto, who have always supported me. Without them, I would never have achieved my goals, both on a personal and on a work-related level.

My friends and colleagues were also an important factor. Both the time working on projects, where lots of ideas were exchanged, and the time spent on past time activities are of immeasurable value to me.

Finally, I want give a special thanks to my friends and colleagues Thomas Seidel, Gökhan Ibis and Colin Manson for their time spent reviewing this thesis.

Contents

List of Figures

Chapter 1

Introduction

This chapter aims to present the motivation behind this work, as well as to briefly introduce the most important concepts and methods, which are relevant to the conducted research.

1.1 Computer aided drug design (CADD)

Drug discovery is both a costly and lengthy process. Estimates state that bringing a new medicine to market can incur costs between 944m and 2,826m USD when accounting for the cost of failures [43] and between 10 and 15 years of research and clinical studies are needed before a drug is administered to patients [2]. Additionally, only 20% of drugs tested in clinical trials will get approved by the authorities. [2] For these reasons, novel approaches are needed to improve upon the cost, time, and success rates of the drug discovery process.

Computer aided drug design (CADD) has been used successfully in pharmaceutical chemistry by providing methods and tools that enable to discover, develop and analyze novel drug candidates both faster and more efficiently through the use of computational resources. [49] CADD methods reduce the need for creating and/or testing compounds in laboratories, lowering research cost and time while improving the odds of a drug successfully passing the different development stages. This can for example achieved by identifying sets of promising compounds [45] through computational pre-selection methods, reducing the number compounds which must be experimentally tested out of a potentially very large library of candidates. Additionally, CADD enables the generation of additional insight into the chemical processes and the interactions of different compounds and targets through simulations and visualizations.

Generally, the in silico process of CADD [8] can be started from two different methodological design standpoints, namely ligand- or structure-based drug design. The following two sections will present the general idea behind these methodologies and give examples for methods and application domains.

1.1.1 Ligand-Based Drug Design

Ligand-based drug design approaches facilitate the process of drug discovery by analyzing the structure of known active compounds for a certain target and deriving information to find similar novel compounds. Thereby, it is assumed that molecules that are structurally or otherwise similar should exhibit similar effects on a given target [1]. The most common approaches for this are quantitative structure-activity relationship studies (QSAR) [15] and pharmacophore modeling. [4]

QSAR methods build a statistical model from a dataset by considering the correlation between the affinities of ligands to their binding sites and various descriptors, such as atom count, lipophilicity, polarizability (many others are possible), or any combination thereof. The resulting model can then be used to generate a prediction for the affinity of other compounds. Note that different more specific QSAR variants are available, such as 2D QSAR [39], 3D QSAR [46], and many others, depending on what kind of descriptors are used by the predictive model. In contrast to QSAR models, pharmacophore models are usually used as a template to align the chemical features of one ligand to another or to a presented model. Thereby, the number of matching features and quality of the fit can be evaluated, indicating how similar the two molecules or the molecule and the pharmacophore model are. Additional information regarding pharmacophores can be found in Section 1.2.

1.1.2 Structure-Based Drug Design

Structure-based drug design, in contrast to ligand-based, is not oblivious to the protein or possible interactions but includes this information to at least some extent [3]. This additional knowledge is thereby used to further refine candidate compounds or even design them differently from the start. Prominent methods from this domain include molecular docking [29, 9], structure-based pharmacophores [34] and molecular dynamics (MD) simulations [7].

Molecular docking aims to predict the predominant binding mode(s) of a ligand with its 3D-structure in a protein complex and compute a score for the predicted pose(s). [31] The most important factors for successful docking are the speed and accuracy of the exploration of the 3D-space as well as a correct ranking of candidate positions. Docking is often used for the virtual screening of compound libraries employing the docking score as primary ranking criterion. The concept of (structure-based) pharmacophores is quite similar in this regard but simplifies the alignment process by considering only a small set of representing non-bonding interactions with the binding site (see Section 1.2) for both the model that is aligned to as well as the ligand. A resulting alignment score, or even just the information, whether or not a molecule fits the model, can again be used for virtual screening. Molecular dynamics simulations, in contrast to the other methods, do not limit a ligand/protein into a rigid conformation but can also consider which internal motions and resulting conformational changes are at play. [21] They are thereby a valuable tool for analyzing the different present interactions in finer detail. This comes with the downside, though, that simulating such protein-ligand complexes, even only over short time periods, is very computationally expensive and as a consequence relatively slow.

1.2 Pharmacophores

Pharmacophores are 3D spatial models that abstract the chemical features of molecules into labeled points or spheres with distance thresholds (pharmacophore feature) [24, 23, 25]. These points/spheres thereby represent the components responsible for the biological activity of a compound. [20] Typical labels include Hydrophobic (H), H Bond Donor (HBD), H bond Acceptor (HBA), Halogen Bond Donor (XBD), Halogen Bond Acceptor (XBA), Positive Ionizable (PI), Negative Ionizable (NI) and Aromatic Ring (AR). This common set of labels can easily be extended to include additional interactions, for example with a metal binding label, if desired. The pharmacophore features are commonly represented as spheres with a distance threshold as radius. Other meta-information on the interaction can also be included, such as a direction vector, describing in which direction the interaction partner is expected, as

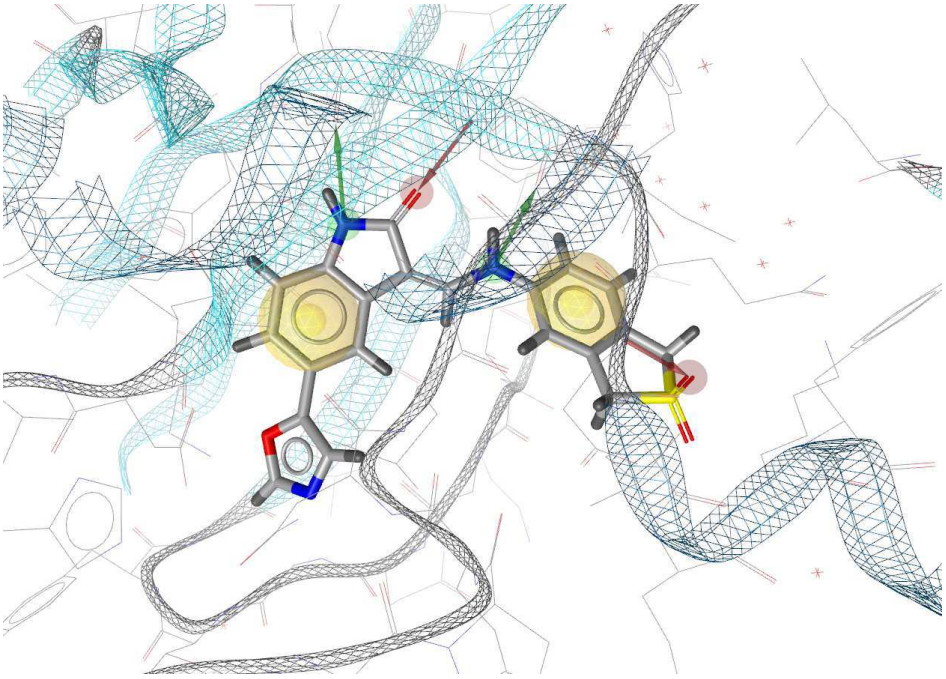


Figure 1.1: Example of a Pharmacophore in LigandScout [47]

well as a threshold angle for this direction. Additionally, exclusion volume features modeling the receptor structure may be defined, to indicate that no atoms of aligned molecules should overlap with these regions of 3D space since it is occupied by receptor atoms. A visual example of a pharmacophore aligned to a ligand within a protein complex can be seen in Figure 1.1, where some of the features have direction vectors assigned.

Abstracting molecules into a set of structural features can be especially useful in CADD, as often not the individual atoms of a compound are of interest, but its (possible) chemical interactions. [34] This allows the researcher to work on a higher level of abstraction, which can result in more explainable results. Also, as pharmacophores usually contain a lot fewer features than the abstracted compound contains atoms, the methods using pharmacophores can often be significantly faster than methods working with atoms directly. These aspects may also contribute to pharmacophore models being commonly used for the task of virtual screening, where the goal is to identify molecules that can trigger a desired biological effect. [36] For this, databases of molecules are aligned to the pharmacophore and filtered according to whether or not their features match the presented model, or even more granularly, via their fit score. Due to the simplicity of the structure of pharmacophores, even databases with millions of compounds can be screened quickly and efficiently. [41]

To create a pharmacophore model that most likely encodes the correct interaction patterns, the right 3D organizations of atoms (conformations) are needed. These conformations can either be identified experimentally from the bound state(s) of the ligand or generated in silico with conformer generation methods. Other than this, different options are available for the creation of a pharmacophore, depending on the availability of the target structure. If only the active/inactive ligands are known, ligand-based pharmacophores can be created while additional knowledge of the protein can refine the ligand-based model or enable the creation of a purely structure-based pharmacophore. If none of this information is available, there is no reasonable way to use pharmacophores, as there is no way of defining the relevant

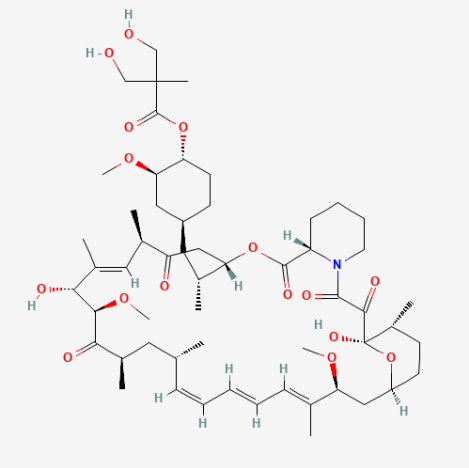


Figure 1.2: The Macrocycle Temsirolimus (PubChem [32])

chemical interactions. Generally, it is advisable to consider as much available information as possible when creating a model, while splitting the data into training and test sets to validate it. [36]

While pharmacophores are usually used as "positive" search filters in screening, it is important to note that they can also be used as "negative" ones, for example, if one wants to find compounds that do not interact with a target. [36] This can, for example, be useful when analyzing the potential toxicity of a molecule or when a compound that is inactive for a specific target is desired.

1.3 Macrocycles

Macrocycles are ring systems where at least one elemental ring consists of 12 or more atoms [27]. Natural products, which are often molecules with such a macrocyclic core have been found and successfully used as drugs [33]. This indicates that molecules with such scaffolds could have an evolutionary advantage, which may be exploitable in other medicinal drugs, as was, for example, found to be the case for the mTOR inhibitor temsirolimus (Torisel®)[22], a depiction of this compound can be seen in Figure 1.2. The efficacy of macrocyclization strategies when designing novel compounds with specific biological and physiochemical properties has also been shown in different publications. [28, 27, 50] These findings raise interest in macrocycles, but the synthesis of these structures proved to be difficult in the past and was, therefore, unfeasible in practice.

Recently, new environmentally friendly [37] and efficient synthesis methods have been discovered, allowing for the creation of molecules with innovative macrocycles. [42] Even more recent advances have demonstrated the ability to synthesize macrocycles in 1.5 hours at a gram scale. [44] These fast developments in the domain of molecular synthesis and the resulting increase in the feasibility of custom macrocyclic molecules further raise interest in such scaffolds in medicinal chemistry. For this reason, there is also added interest in reliable CADD methods for the research on such novel molecules with macrocycles. Still, due to the cyclic structure and the often larger size of the compound compared to classical drugs, many computational methods have difficulties processing molecules containing macrocyclic ring systems.

1.4 Scientific goal and motivation of the thesis

The main goal of this doctoral thesis is to support chemoinformatics research through the use and development of novel computer science methods. If the state of the art methods for a specific task fail for many or even only specific molecules currently, or they are too slow for practical application, this is a domain of interest. The research question can thereby be defined as: Where and how can the field of chemoinformatics be advanced with methods from computer science?

A problem observed with previous tools and algorithms is that they have difficulties dealing with macrocyclic or larger molecular structures, resulting in either inaccurate results or too long runtime for practical use. For this reason, novel methods were implemented that push forward the state of the art in chemoinformatics, with special emphasis on the usability with more complex structures, such as macrocycles.

Chapter 2

Virtual Screening

2.1 Introduction

Virtual molecule screening is the task of identifying molecules from a (usually large) collection or database fit a presented query model. [16] A variety of models can be used, but pharmacophores, describing a set of desirable chemical interaction capabilities, have been shown especially successful. [18] For this reason, the main focus of research of this work lies on pharmacophore-based alignment screening. Generally, two main questions must be answered for virtual screening, one after the other:

1. Does the molecule fit the model and its restrictions? (strict yes/no filtering)
2. How well does it fit the model? (ranking/scoring)

Firstly, any molecule that, per definition, fits the model should be found, i.e. no molecule, even if it only barely fits, should be lost by an optimal screening algorithm. The problem is that finding a rigid transformation that results in the best fit between the pharmacophore model and the molecule is a challenging problem. This task becomes even more difficult when considering additional constraints, such as exclusion volumes and varying distance thresholds. Allowing for omitted features, whereby only a subset of the features of the model pharmacophore needs to be matched, is also possible and commonly done in practice. Only if a query molecule fits a model and fulfills all of its constraints, a ranking/evaluation of the quality of the fit is sensible.

Secondly, given a molecule that fits a model, a meaningful way of quantifying the quality of the fit should be defined. The goal of such a measure is to be as discriminative as possible, i.e. to have a value that can be understood and based upon which educated choices can be made, as well as to, in the best case, correlate with activity (assuming a good model). Another important aspect of such a measure is that it should fulfill metric properties, as this would allow for broader use with other computational methods, such as many common clustering algorithms.

Contribution made

This publication focuses on the first part of this problem, finding whether or not a molecule fits a pharmacophore model, as the state of the art methods often fail at this task. The general goal is to find the transformation which results in the largest possible number of features of the query pharmacophore to match the features of the model pharmacophore.

Previous methods are not suitable for finding such a transformation, as they optimize for secondary goals such as Root Mean Squared Distance (RMSD) or volume overlap, disregarding the fundamental constraints of the model. This methodology does indeed produce good results regarding the targeted score but may miss out on possible solutions altogether because the correct transformation would have a worse score or because the best transformation barely creates a mismatch of some required features according to the thresholds of the model. In other words, a method may place a feature barely outside a threshold sphere to obtain a better RMSD, but thereby invalidate the solution altogether, according to the model's requirements.

This problem becomes worse the more omitted features are allowed and the more features are available in both the model pharmacophore and the queried compounds' pharmacophores, which also results in poor accuracy when working with large compounds such as ones with a macrocyclic core.

To improve upon the aforementioned methods, a novel alignment algorithm is proposed, which focuses on finding a transformation that results in the largest possible number of matching features. The main idea behind it is to set this number as the main optimization goal and incrementally build a better transformation starting from one or more rudimentary first guesses. Different ways of generating these initial guesses are proposed and the runtime/accuracy trade-offs of using more/fewer guesses are explored. It is also shown, that the new algorithm is runtime-independent of the "omitted feature" parameter, which greatly benefits its performance relative to the competition.

2.2 Greedy 3-Point Search

molecules

MDPI

Article

Greedy 3-Point Search (G3PS)—A Novel Algorithm for Pharmacophore Alignment

Christian Permann [1,2], Thomas Seidel [1,*] and Thierry Langer [1,2]

[1] Department of Pharmaceutical Sciences, University of Vienna, Althanstrasse 14, 1090 Vienna, Austria; christian.permann@univie.ac.at (C.P.); thierry.langer@univie.ac.at (T.L.)
[2] Inte:Ligand GmbH, Clemens Maria Hofbauer-Gasse 6, 2344 Maria Enzersdorf, Austria
* Correspondence: thomas.seidel@univie.ac.at

Abstract: Chemical features of small molecules can be abstracted to 3D pharmacophore models, which are easy to generate, interpret, and adapt by medicinal chemists. Three-dimensional pharmacophores can be used to efficiently match and align molecules according to their chemical feature pattern, which facilitates the virtual screening of even large compound databases. Existing alignment methods, used in computational drug discovery and bio-activity prediction, are often not suitable for finding matches between pharmacophores accurately as they purely aim to minimize RMSD or maximize volume overlap, when the actual goal is to match as many features as possible within the positional tolerances of the pharmacophore features. As a consequence, the obtained alignment results are often suboptimal in terms of the number of geometrically matched feature pairs, which increases the false-negative rate, thus negatively affecting the outcome of virtual screening experiments. We addressed this issue by introducing a new alignment algorithm, Greedy 3-Point Search (G3PS), which aims at finding optimal alignments by using a matching-feature-pair maximizing search strategy while at the same time being faster than competing methods.

Keywords: pharmacophore alignment; pharmacophore modelling; virtual screening; greedy algorithm; drug design

Citation: Permann, C.; Seidel, T.; Langer, T. Greedy 3-Point Search (G3PS)—A Novel Algorithm for Pharmacophore Alignment. *Molecules* **2021**, *1*, 0. https://doi.org/

Academic Editor: Martin Vogt

Received: 30 September 2021
Accepted:
Published:

1. Introduction

3D Pharmacophores are spatial models that abstract the chemical features of molecules into labeled points [1–3]. These points can be supplemented by additional information like feature position tolerances or direction vectors. Such pharmacophore models have proven to be a useful tool for describing macromolecule–ligand interactions [4] and provide an intuitive abstraction of ligand features that are key for high binding-site affinity. Typically, pharmacophores modelling energetically favourable binding-site interactions are created by first finding two or more active molecules that bind to the target receptor of interest and then generating a consensus pharmacophore model. Alternatively, pharmacophores can also be derived directly from the 3D structure of one or more ligand–target complexes [2,5,6]. The latter approach generally leads to higher quality models due to the availability of bound-state ligand structures and binding-site environment information. In any case, the obtained pharmacophores usually need to be further refined by an expert user towards higher quality regarding their ability to discriminate between active and inactive molecules [2].

Since pharmacophore features model non-bonding interactions of functional groups and moieties without being tied to the underlying chemical structure, a single pharmacophore model may cover a wide variety of molecular structures with distinct scaffolds and functional groups that expose the same feature pattern. Pharmacophore-based techniques thus have an inherent scaffold hopping ability, which is of high value for lead optimization tasks or the tailored design of drug molecule [7].

One of the most prominent application of pharmacophores is their use as a filter for the virtual screening of large compound libraries (for which pharmacophores also have

been pre-generated), which aims at finding molecules providing the same chemical feature pattern as specified by the query pharmacophore [5,8]. Found molecules where all or most of the query features match are likely to be also active towards the target of interest and represent a good first set of candidates for further experimental activity analyses. In comparison to a random selection of molecules, a refined set of candidate molecules significantly reduces resources spent with in vitro testing in the hit-finding phase and as a consequence also helps to reduce the overall drug discovery and development costs [9].

For virtual screening, query pharmacophore features are usually associated with certain positional tolerances to accommodate slight pharmacophore differences resulting from distinct molecular scaffolds and associated conformational spaces. How well two particular features match depends on how many feature pairs can be overlapped at most, given the optimal rotation and translation. The process of finding such an optimal rigid body transformation is called alignment. Pharmacophore alignment can be considered as the "heart" of the pharmacophore-based screening process in which the ultimate decision is made whether two pharmacophores match or do not match. Therefore, high robustness, reliability, and computational efficiency of the underlying algorithm are key factors regarding the usability of the screening software in real-world scenarios. Additional consideration has to be put on the conformational flexibility of molecules. Various methods for aligning molecules in a "flexible" way have been proposed [8], each with its benefits and downsides. In the context of pharmacophores, the problem of flexibility is often abstracted by simply pre-generating multiple low-energy conformations for a molecule and then aligning the pharmacophore of each conformer individually [8]. As pharmacophore-based alignments are relatively fast, large numbers of those alignments can be performed, which usually allows for a sufficiently good treatment of conformational flexibility. In the following, we do not go further into the details of molecule conformer generation since this topic is out of the scope of this study, and a wide variety of well-validated software tools for this task is already available [10–14].

For tackling the pharmacophore alignment problem, several methods and algorithms were devised [15–17]. Many of them have been implemented as part of well-known software platforms for pharmacophore modelling and screening like LigandScout [5,18], Phase [17,19], Catalyst [16,20], or MOE [21].

The algorithm by Wolber et al. [15] (due to the lack of a name, hereinafter referred to as the RM method/algorithm or RMM), for example, is used for generic pharmacophore alignment tasks and for pharmacophore matching in LigandScout's virtual screening pipeline. In the first step of the algorithm, each pharmacophore feature gets abstracted through its feature type and a histogram of distances to the surrounding features. For each feature type present in the neighbourhood, one histogram is created, counting occurrences in the bins of the histogram. As distances may be close to falling into a neighbouring bin, smoothing filters are applied to the histogram. This results in the histograms of two features, where the features are in neighbouring bins, to be more similar. Next, a cost matrix for solving the pair assignment problem of the features of the two pharmacophores to align is created. This is done by assessing the similarity between the encoded features and setting a cost accordingly. This cost matrix is then used with the Hungarian algorithm [22] to find the best assignment that minimizes the overall cost. As a result, matching feature pairs are obtained, which are then used to compute a pharmacophore 3D alignment employing Kabsch's method [23,24]. If the found alignment is not plausible according to the specified feature position tolerances, the pairs violating this condition are removed, and the method is repeated from the point of finding ideal feature pairs with the Hungarian method. If the alignment does meet the feature-matching criteria, the alignment transformation matrix is kept and applied to the underlying molecules. In a further post-processing step, the alignments are then checked whether they also fulfil any additional constraints imposed by directed features or exclusion volume spheres (see the methods section). Should those not be met, the alignment will be discarded and the algorithm proceeds until a valid alignment could be found, or it terminates without a found alignment solution.

Other alignment algorithms are based on volume overlap and model atoms or pharmacophore features such as Gaussian volumes, for which they optimize towards maximum overlap [25].

ROCS [26] is a well-known example for this group of methods, which considers Gaussian volumes defined by the atoms and (optionally) additional chemical feature information to align molecules.

Pharao [27] is also a method where pharmacophore features are modelled as Gaussian spheres. It operates by generating combinations of "feasible" pairs, which are then aligned by superimposing their coordinate centres and computing a rotation with a constrained gradient ascent. To create feasible pairs, the distance between two points of one pharmacophore and then the distance between two points (with matching feature types) of the other pharmacophore is measured. The difference between those distances is set in relation to the sum of distance thresholds for the four features and compared to a cut-off value ϵ (usually 0.5). If the value is smaller than ϵ, the pair is deemed feasible. Next, feasible pairs are assembled combinatorially, whereby any pair added to a combination must be feasible and compatible with all other pairs of the combination. A tuning of the ϵ parameter allows to reduce the total number of feasible combinations. For every combination produced, alignment is performed by first aligning the centres of the pharmacophore feature subsets. In the next step, constrained gradient ascent optimization is applied to find the rotation needed for the second pharmacophore to achieve maximum overlap with the first one. This procedure is carried out for four different starting orientations and results in a rotational transformation and an alignment score quantifying the overlap. Combinations with a larger number of matching features are aligned first, allowing for early termination if smaller combinations have a far smaller overlap.

Existing alignment methods that optimize towards minimizing the root mean squared distance (RMSD) of corresponding feature pairs disregard that worse RMSDs may allow for more features to match. This disconnect between the optimization goal and the actual goal of alignment can be intuitively understood when thinking about the best case for an unambiguous alignment with RMSD. For any pair of pharmacophores, the best solution regarding this criterion would be to only align the three best matching features, resulting in a smaller RMSD than if one were to include even only one additional feature pair. For this reason, alignments may be discarded as the algorithm favoured the perfect fit of few features over the user defined model definitions. Methods optimizing for volume overlap of pharmacophore features modelled by Gaussian volumes have the same inherent problem since after optimization for maximum overlap, features may no longer fulfil positional and/or directional constraints and will be considered non-matching. This is an issue as the aim of the alignment, even for these algorithms, is to maximize the number of matching features. This problem affects the results of virtual screening drastically as the tested pharmacophore may be able to fit all of the features of a query pharmacophore but may still not be found due to the optimization goal. For illustration, Figure 1 shows two pharmacophore models that both consist only of features of the same type. Depending on the primary search goal—optimum RMSD/volume overlap vs. maximum matching feature count—different alignment results will be obtained (note that features are only considered matching if the central point of one feature is within the tolerance sphere of the other).

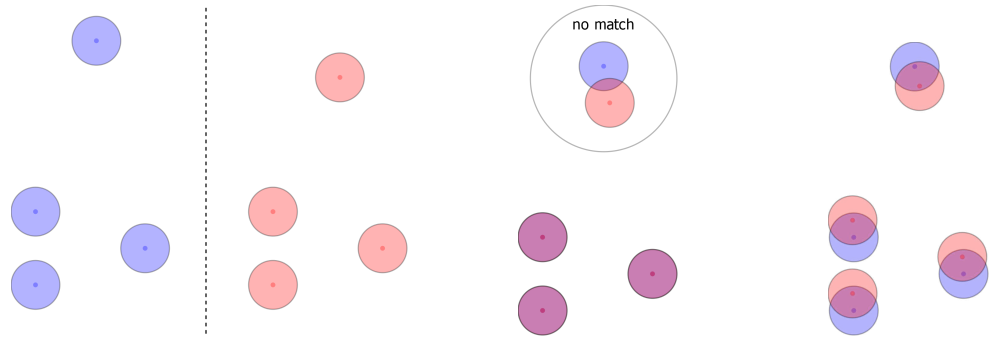


Figure 1. Illustration of different pharmacophore alignment strategies and the thus obtained results (**b**,**c**). For the sake of simplicity, pharmacophores (**a**) consist only of features of the same type. Different colours are just used to distinguish the pharmacophores. (**b**) Result from RMSD- or volume-based alignment. (**c**) Max. matching feature count-based alignment.

Another problem with the existing methods is how they deal with the allowed omission of user-defined less-critical "optional" features during pharmacophore alignment. The RM algorithm, for example, performs a combinatorial sampling and generates all possible combinations with #o missing features. If the models contain many features and #o is large, this will lead to an explosion of computational complexity and results in significantly longer runtimes. The generation of combinations based on omitted features and using them purely with Kabsch alignment additionally leads to another unintuitive issue: as the method may not identify a possible match without the omitted features due to the nature of purely optimizing towards RMSD, computing an alignment for less features may, by chance, find alignments where more of the features match. If aligning a larger number of features did not result in a valid alignment, less features are tried, and the centroid computed for the Kabsch alignment will be at a different place. With a different centroid, the algorithm may now be able to find an alignment for which the previously unmatched features match by coincidence. Thereby, allowing omitted features could lead to finding alignments that actually should have been found without them or even completely different alignments.

The treatment of exclusion volume spheres (see the methods section for details) is also problematic. Existing methods compute the alignments without considering these features and only discard the result post computation when they find exclusion volume constraints to be violated. This can lead to solutions, possibly even very good ones, to be discarded only because an exclusion sphere gets slightly touched. Visually, this would be an easy problem to solve, as just slightly moving a molecule may yield a valid alignment.

In summary, the core issue with all the discussed methods is that the optimization goal they follow is not in line with the intuition of pharmacophore models. As individual features have a positional tolerance threshold for matching, the best alignment solution may not be the one with minimal feature-pair RMSD or maximum volume overlap but the one providing the largest number of matched feature pairs under the given feature position and direction constraints. The latter criterion should be given preference especially when pharmacophore models are used as virtual screening filters in order to reliably identify all database molecules that provide a specific chemical feature pattern.

In the herein presented work, we address all of the above discussed main problems of the existing pharmacophore alignment methods by introducing a new alignment algorithm, Greedy 3-Point Search (G3PS), which aims at finding optimal alignments by using a matching feature pair maximizing search strategy while at the same time being faster than competing methods. The performance of the presented algorithm regarding its reliability in the identification of possible pharmacophore matches and its computational efficiency was assessed by screening several molecule datasets taken from the DUD-E database [28,29] with query pharmacophores derived from the corresponding ligand–target complexes.

To provide an impression of the advances that could be achieved with our new algorithm, the obtained results are presented in direct comparison to the ones obtained with the RM algorithm.

2. Methods

This section provides a detailed description of our new alignment algorithm G3PS. Before presenting any details of the new algorithm, we first want to provide a definition of the used pharmacophore representation, introduce important concepts regarding the goodness of fit of two pharmacophore models given an optimal transformation, and point out additional aspects that are relevant for pharmacophore alignment.

2.1. Definitions and Concepts

Pharmacophores. Pharmacophores are a collection of labelled points representing chemical features [1–3]. Commonly used features include aromatic rings, hydrogen-bond acceptors/donors, and positive/negative ionizable and hydrophobic interactions, although others exist and custom features may be designed. For a given pharmacophore A, we denote feature i as A_i for $i = [1, 2, \ldots, |A|]$. Features additionally contain a distance threshold $thresh(A_i)$ indicating how far the centres of matching features may be away and, optionally, other restricting factors such as feature direction vectors.

Exclusion Volume Spheres. A pharmacophore A may contain additional features A^x, which can be interpreted as "anti-features." These features restrict the space where atoms of the aligned molecule may occur, qualifying any alignment as invalid where atoms collide with such feature shapes. More generally, their goal is to abstract the presence of binding-site-environment atoms and model those locations, defining that there would not be enough space left for atoms of a binding molecule. For performance and simplicity reasons, these shapes are usually modelled as spheres, so we will use the terms "exclusion volume" and "exclusion sphere" synonymously. This greatly simplifies the evaluation of the validity of an alignment solution since just a check of the condition $dist(a, A_i^x) > thresh(A_i^x)$ for all atoms a and all exclusion spheres A_i^x needs to be performed. If the condition is true for all combinations, the alignment is valid and can be kept.

Fully Fitting Models. For two models to fully fit, there must exist a transformation from pharmacophore A to B (and B to A) that includes only translation and rotation (no scaling) for which the following holds:

- There exists a mapping for all features of A and B where:
  1. All individual features map uniquely to a feature of the other model, creating index pairs (i, j).
  2. The labels (feature types) of paired features are equal
     $$(i, j) \implies label(A_i) = label(B_j).$$
  3. The paired features distance is smaller than the larger distance threshold.
     $$(i, j) \implies dist(A_i, B_j) < max(thresh(A_i), thresh(B_j))$$

Maximum Fitting Models. Models with maximum fit may not meet the criterion of all features mapping to pairs, but all features of the query pharmacophore do. In other words, the pharmacophore that is used as a query will fully fit a subset of the other pharmacophore. This is the best achievable fit two models of different size can have. If the query pharmacophore is larger than the tested one, such a fit is not possible, and, at best, a partial fit can be achieved by allowing features to be omitted.

Partially Fitting Models. Models fitting partially are ones for which less than the number of features in the query pharmacophore match. To define the bound for a sufficient fit, a maximum number of omitted feature #o needs to be defined. If the best possible transformation, where the largest number of features match, is missing at most #o features of the query pharmacophore, the alignment is accepted, otherwise, the pharmacophores are considered non-matching.

Quality of Model Fit. The goal of aligning two pharmacophore models is to find a transformation that allows for the largest number of features to match. For this reason, the most important measure of quality is the number of matching pairs. A matching pair is defined as two features, one from each pharmacophore, for which all restricting properties are fulfilled, including having the same feature type and their distance being smaller than the larger distance threshold. If the number of matching feature pairs is equal for two found model alignment solutions, the quality of alignment can then be asserted by calculating the RMSD of the matched pairs. We want to put additional emphasis on the fact that the RMSD should only be considered regarding the alignment quality after determining the maximum number of matching features, as aiming for the smallest possible RMSD does not optimize towards finding the largest match. The best example for this is to only consider the best matching pair for computing the root mean squared distance. Any model (where at least one feature type matches) can achieve an RMSD of 0 if we only align one feature pair. By extension, a smaller number of matching features is more likely to have a small RMSD.

2.2. Greedy 3-Point Search (G3PS)

Our new alignment algorithm was implemented in a development version of Ligandscout [5,18] using the programming language Java and consists of two main parts followed by a post-processing step. First, the features of both pharmacophores are encoded and compared to identify which features of one pharmacophore are likely to match which features of the other pharmacophore. From this, first guesses are constructed using exactly three pairs of features, as this is the minimum required for computing an unambiguous 3D transformation. In the second step, iterative refinement is performed for each of the three-pair guesses. This refinement process collects additional pairs that do not break the alignment created from all previously collected pairs and applies transformations that optimize the alignment to better fit with the newly found pairs. Optionally, after this second step, exclusion volumes can be considered. Final additional checks can then be performed to filter out alignments for which more specific feature requirements are not met.

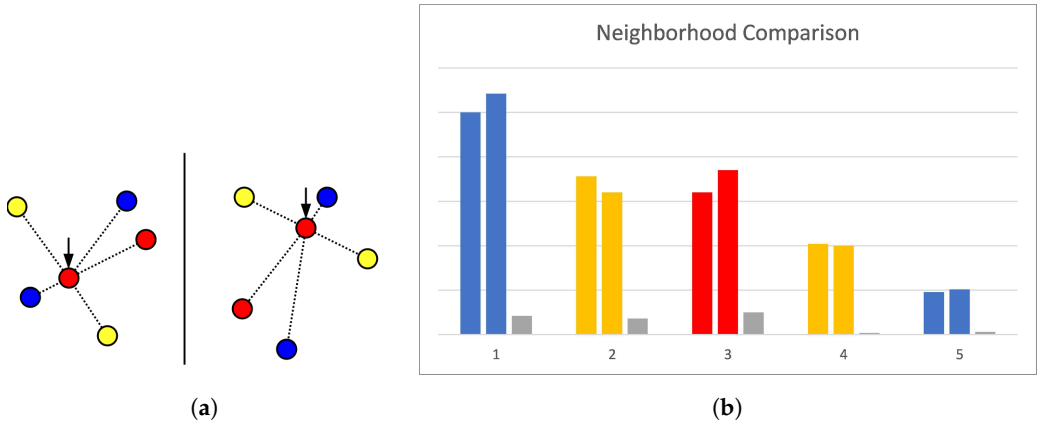
2.2.1. Finding Three Pairs

To find a starting point for iterative refinement, at least three points need to be fixed in each of the pharmacophores, which allows to create an initial alignment. The number of such three-pair guesses being generated is an important factor for the overall runtime of the algorithm, as it defines how often refinement needs to be performed. Still, the downside of only having one or few starting alignments is that the globally optimal alignment may be missed. Therefore, we allow the user to define the maximum number of sampled guesses, enabling them to trade computation time for accuracy.

Single Best Guess. To compute an initial guess that is likely to result in a globally optimal matching, we computed a matrix describing how dissimilar each feature of one pharmacophore is to the features of the other. The creation of this dissimilarity matrix is performed by encoding each feature, taking its neighbourhood into account. Observing a feature A_i, we captured which kind of feature is at what distance to the observed one, looking at all features A_j where $j \neq i$. This list of labelled (by feature type) distances and the type of the observed feature itself now represent the encoding used to compare features and measure their dissimilarity.

Comparing these encodings for all features of one pharmacophore (rows) with all features of the other (columns) will then result in the values of the matrix (see Figure 2 for an example). If two encodings are representing features of different types, then the pair is marked invalid, otherwise they are compared by solving a mathematical assignment problem for which the algorithm of Jonker and Volgenant [30] is used. There, the difference is computed for each pair of encoded distances, and entries with different labels are treated as being maximally different. As too-different distances indicate that the features do not match, and starting at some difference does not have discriminative value, this difference

has to be limited. For this reason, the distance difference is divided by the larger sum of positional tolerances of the two compared feature pairs. If the resulting number is larger than 1, we know that the pairs are incompatible, and the value is set to exactly 1. The total cost resulting from this assignment is then normalized by the number of features in the larger pharmacophore and represents how dissimilar the two encoded features are regarding their surroundings. This feature dissimilarity measure is used in the next step as a pseudo-cost for the assignment of one feature to the other, allowing us to fill the cost matrix.



(**a**) (**b**)

Figure 2. Visualization of neighbourhood comparison. Distances are measured for fixed features in each pharmacophore (**a**) and compared to compute the deviation between corresponding pairs (**b**). Colours = labels, coloured bars = encoded distances, gray bars = difference between distances

$$\begin{pmatrix} X & X & 0.81 & 0.12 & X & 0.04 \\ 0.01 & 0.62 & 0.12 & 0.69 & 0.32 & 0.71 \\ 0.62 & 0.12 & X & 0.04 & 0.95 & X \\ X & 0.02 & 0.89 & 0.08 & X & 0.23 \\ X & 0.72 & 0.45 & 0.23 & 0.01 & 0.13 \\ 0.21 & 0.92 & 0.03 & X & X & 0.17 \end{pmatrix} \rightarrow \begin{pmatrix} X & X & 0.81 & 0.12 & X & 0.04 \\ 0.01 & 0.62 & 0.12 & 0.69 & 0.32 & 0.71 \\ 0.62 & 0.12 & X & 0.04 & 0.95 & X \\ X & 0.02 & 0.89 & 0.08 & X & 0.23 \\ X & 0.72 & 0.45 & 0.23 & 0.01 & 0.13 \\ 0.21 & 0.92 & 0.03 & X & X & 0.17 \end{pmatrix}$$

$$\text{Match:}\{(2,1),(4,2),(5,5)\}$$

Figure 3. Example matrix used for the assignment of matching pairs. X indicates feature mismatches, marking a given pairing as impossible

From this matrix, we can obtain a guess by solving another mathematical assignment problem and picking the three matches with minimal dissimilarity values. An example of this can be seen in Figure 3.

This guess should in theory represent the best option for running an iterative refinement process. Still, as we abstracted the 3D information of where points are located relative to each other to simple distances, our guess may not always result in the optimal solution after refinement. This could, for example, be the case if the pharmacophores would only fit after a symmetry transformation. To combat these non-optimal guesses, we propose methods to obtain multiple starting combinations.

Multiple Guesses. To obtain an arbitrary number of guesses m, we can reuse the dissimilarity matrix mentioned above. Iterating over this matrix, we can efficiently check the feasibility of every combination of three pairs, keeping track of the m best ones. To compute how good a guess is we simply sum up the dissimilarity values for the three pairs, with a smaller sum indicating a better choice. Note that even though this process has an algorithmic complexity of $O(n^6)$ (where n is the number of pharmacophore features), this is in practice very fast, as permutations of the same three pairs can be ignored, and pairs

with incompatible feature types can be skipped. We also found that this guess-generating process is negligible in runtime compared to the iterative refinement of the found guesses.

Trying All Possibilities. It is also possible to just use all combinations of three pairs as starting points for iterative refinement. For this, we do not need to compute a dissimilarity matrix for features but just keep track that all pairs contain features of the same type. This should result in the most accurate alignment, as the largest possible number of starting combinations is sampled, but it has the downside of being more computationally expensive, due to the time needed for refinement. Still, this should be faster than choosing a different algorithm altogether as there are fewer three-pair combinations than if one computes all possible combinations (of all sizes) for the features, as other methods may do.

2.2.2. Collecting Additional Pairs

For each initial guess of three matching pairs, iterative refinement is performed to both align the collected pairs and to find additional pairs that fit. This idea is similar to the Iterative Closest Point (ICP) algorithm [31], which is usually used to register unordered point clouds from rigid objects by iteratively matching points, evaluating the quality of the transformation and improving on it. Our implementation was modified to better represent the goal of maximizing the number of features with overlapping threshold spheres, while having the same feature type. The concrete steps of the method are as follows:

- Given a list of starting matches P (three initial pairs),
- Align pairs P using the Kabsch Algorithm.
- Build a matrix F "forbidden" for all feature pairs.
  - Mark pairs with different types forbidden.
  - Mark columns and rows related to features in P forbidden.
- While F contains unmarked entries:
  - Iteratively improve P.
- Return matching pairs and alignment matrix.

The matrix F tracks which pairs could be feasible to expand the matched feature pairs and can indicate when the algorithm should be terminated. Each iteration, this improvement process checks a possible addition to the matching pairs and computes a new alignment while setting at least one entry of F as forbidden. Due to the strict increase in marked entries, one can be sure that the method terminates. Additionally, one can stop this process earlier if all features match or if we only care about finding any alignment, while omitted features are allowed and have a sufficiently large matching. Each iteration of the improvement performs the following steps:

- Find closest pair (by distance) p_{test} not marked in F.
- Create temporary pair combination $P_{test} = P \cup p_{test}$.
- Align pairs P_{test}.
- If P_{test} does not fulfill threshold,
  - Try applying translation.
- If P_{test} does still not fulfill threshold,
  - Delete P_{test}.
  - Mark pair p_{test} as forbidden in F.
  - Restore alignment of P.
  - Continue.
- Otherwise,
  - Set $P = P_{test}$.
  - Mark rows and columns related to p_{test} in F forbidden.
  - Continue.

In many cases, it is not necessary to compute the alignment for all of the guesses presented. After refining a guess, the contained tree-pair combinations are stored, and subsequent guesses, starting with three points that were already collected by a previous refinement, are skipped. This is done as in most cases, and refining from those pairs would converge toward the same solution as was previously found.

2.2.3. Post-Processing

In the post-processing step, additional requirements for a valid alignment are checked. One of these checks is the collision with exclusion volumes (see Section 2.1), for which adjustments can sometimes be made. Other than that, features may come with more specialized requirements, for which we can only decide if they are met or not and thereby keeping or discarding the solution. These restrictions on the results are especially useful in screening, as reducing the number of results by using domain knowledge is beneficial for further processing.

Dodging Exclusion Volume Spheres. If the resulting alignment includes atoms that collide with exclusion volumes, additional translations (currently up to three) can be applied to try to "dodge" those exclusion volumes to save the alignment. An example for this is provided in Figure 4 where an exclusion volume would lead to discarding an otherwise good alignment solution.

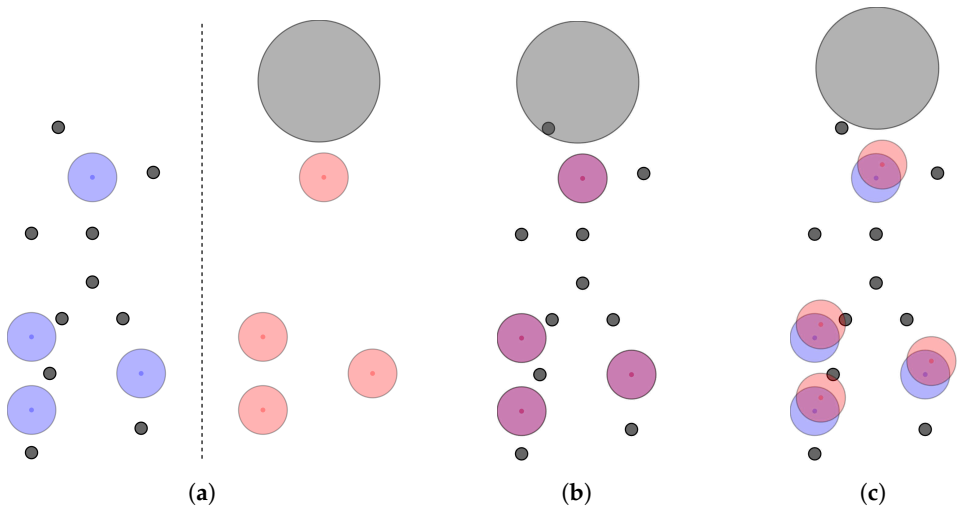


Figure 4. Illustration of exclusion volume clashes leading to an invalid alignment. For the sake of simplicity, pharmacophores (**a**) consist only of features of the same type and atoms of one pharmacophore, shown as black circles. After an alignment of the pharmacophores, atoms may clash with exclusion volumes (**b**), which can sometimes be fixed with a simple translation (**c**).

After those translations, if there are still collisions, the alignment has to be discarded, otherwise the pharmacophore feature thresholds for matched pairs need to be checked again. Based on whether or not the features still are within their allowed threshold distances, the alignment may now be valid or not. This may save a solution that only had a minor exclusion volume clash from being discarded even though it may have been a reasonable alignment.

2.2.4. Other Requirements

Other requirements imposed by pharmacophore features could include a variety of properties like direction vectors or plane normals. Here, as part of the model, one can assume the direction in which the feature should point, and any matching features should also roughly point in that direction after alignment. If this is not the case, the feature is considered unmatched,, and if this leads to less-matched features than required, the solution is discarded. This could also be an interesting topic for further investigation in a follow-up work, as it may be possible to consider these direction vectors in the alignment process itself.

3. Results and Discussion

3.1. Benchmarking Methodology

To benchmark the number of found matches and the required computation time, we implemented our alignment algorithm in a development version of LigandScout for use in virtual screening. Here, the goal was to find molecules, abstracted by pharmacophores, matching one or more query pharmacophores. Since also a significant amount of completely unfeasible pharmacophore combinations would be aligned in a typical virtual screening run, pre-processing filters are usually applied to reduce the overall computation time [8]. This was not done in our benchmarks as we only cared to measure the time spent for the actual alignment process.

To obtain more sensible results, the hits were again filtered after the alignment. Here, alignments were discarded if the features did not meet additional requirements like the directions of feature vectors not being sufficiently close. If there were still clashes with exclusion volumes (after trying to fix those), the alignment was also discarded. The remainder of the found solutions was then accepted and presented as the output hit-list. This filtering based on model restrictions was performed equally for both of the benchmarked algorithms.

Additionally, note that the pharmacophores benchmarked were not manually optimized for separating actives from inactives/decoys. Even though this is what one wants to achieve in practice, the type of retrieved molecules should only depend on the quality of the model. In constrast, the alignment method (which was the focus of our benchmarks) should be able to accurately retrieve all molecules that fit the query model. For this reason, finding additional hits, regardless of if they are actives or decoys, indicates that the method that did not find them was not able to fulfil the user query accurately and omitted results the user asked for. As a side effect of using a more-accurate alignment method and retrieving molecules that fit the query model more precisely, one may need to adjust the model as, in expectation, more hits will be found. As this is often not desirable, reducing the size of feature spheres or additionally filtering the hits with stronger restrictions after screening may be advisable.

All computations for the measurements below were run multi-threaded on an AMD Ryzen 9 5900X 12-Core (24 threads) 3.7 Ghz CPU in Windows 10, taking the average of five runs. Before the first measured run, one full non-measured benchmarking run was performed as a "warmup phase" to ensure the different methods and runs neither benefited nor got punished for start-up-related factors like the CPUs branch prediction and I/O caching.

3.2. Results

First, we assessed the impact of the chosen number of computed guesses for our G3PS algorithm. For this, we compared different datasets in regard to the computation time needed per guess and the number of hits found depending on the number of allowed omitted features.

As can be seen in Figure 5 (using the CDK2 dataset as an example), the runtime of our method did not depend on how many omitted features are allowed but purely the number of computed guesses. As the number of guesses increases, the iterative improvement will check combinations of features that it has seen before. Therefore, the increase in runtime is less than linear in the number of guesses. To now find a good default value, we analysed the number of hits with different parameter values.

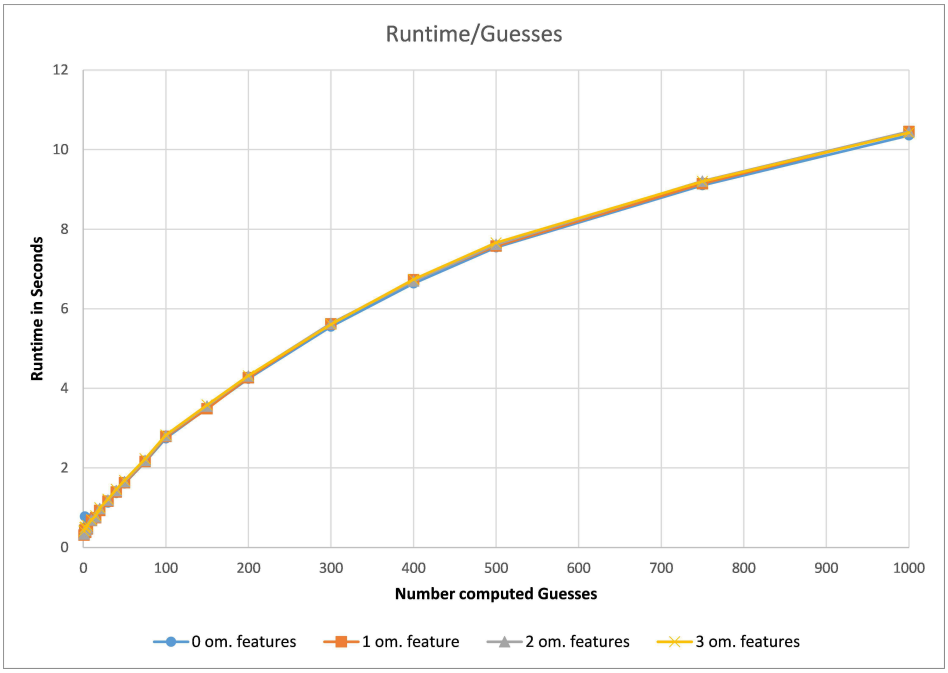


Figure 5. CDK2, runtime (full dataset) vs. computed guesses.

Figure 6 shows that, in practice, only a few guesses are needed to find the majority of hits. In our experiments, we observed that 20 guesses lead to fast and accurate results, while 300 guesses found the maximum amount of hits most of the time. In the following, we call the 20-guess variant "G3PS Fast" and the 300-guess variant "G3PS Accurate".

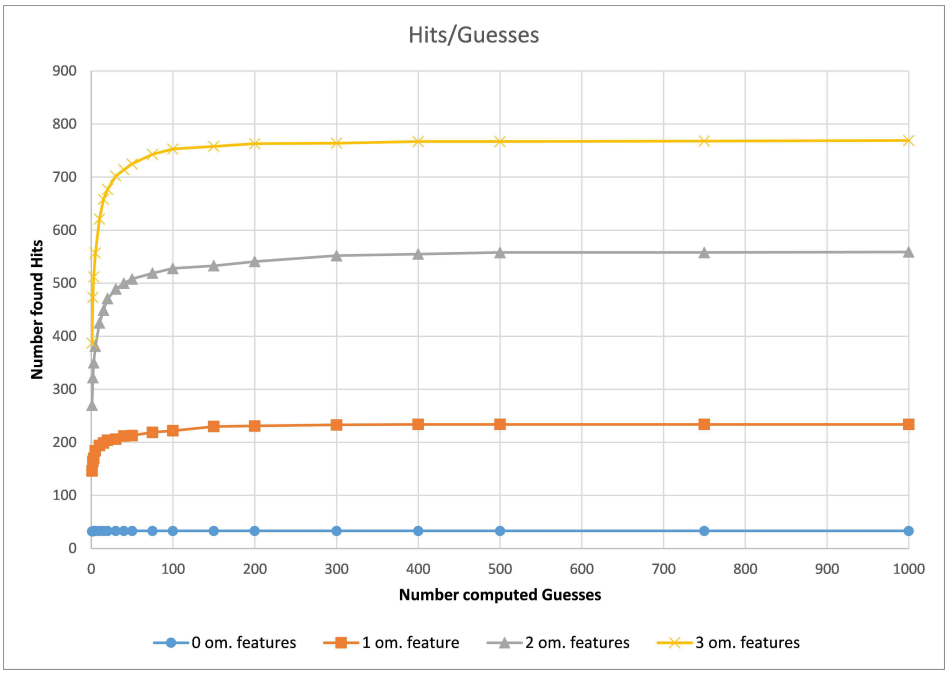


Figure 6. CDK2, number of hits vs. computed guesses.

During testing, we visually analysed differences between RM and G3PS. An example can be seen in Figures 7 and 8, where the same two molecules were aligned based on their pharmacophore features with both methods. The RM method found four matching feature pairs and aligned the molecules according to those while G3PS (accurate) found and aligned by seven matching feature pairs.

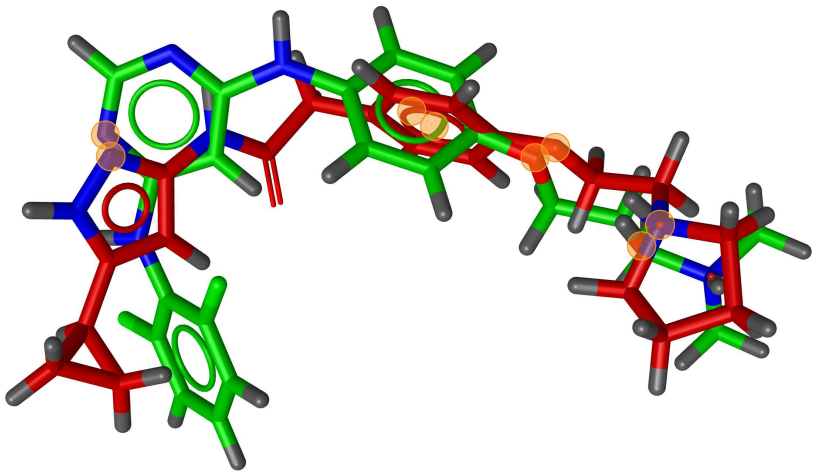


Figure 7. RM method example alignment of two molecules; four matching feature pairs were found. Orange circles = centre of matched features.

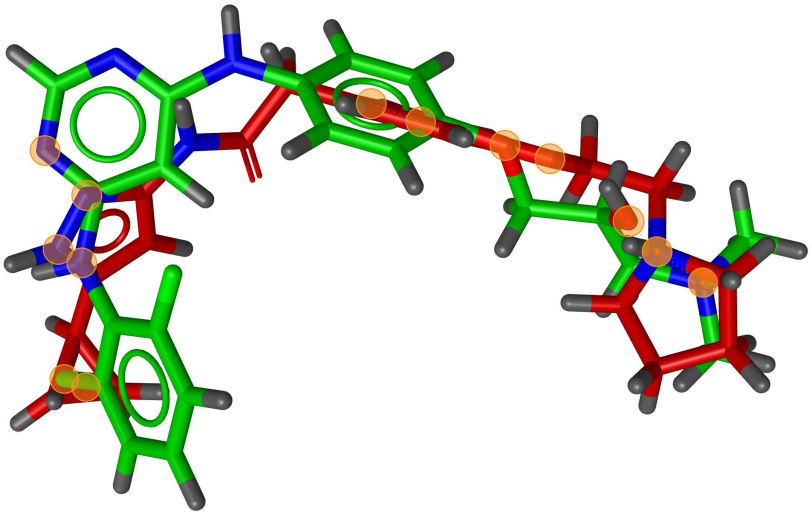


Figure 8. G3PS example alignment of two molecules; seven matching feature pairs were found. Orange circles = centre of matched features.

We also found that many combinations of molecules could not be aligned with the RM method as it could not identify three matching feature pairs. An example of such a combination can be seen in Figure 9, where seven matching feature pairs were identified by G3PS (accurate) and used for alignment.

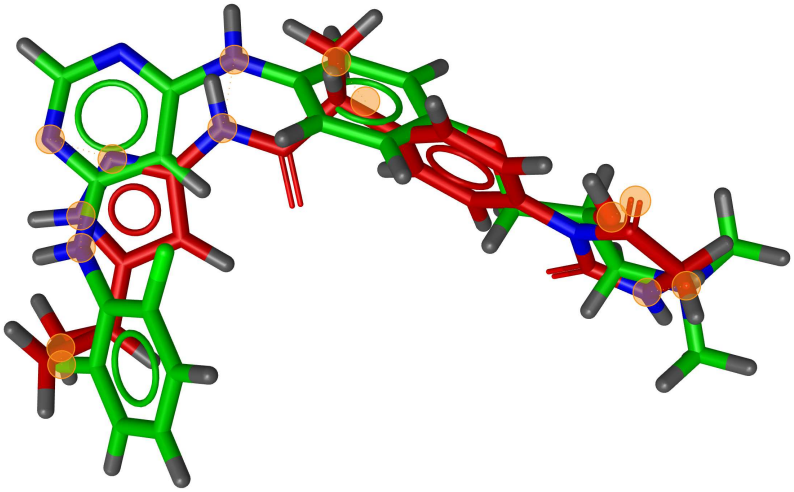


Figure 9. Example of G3PS alignment with seven matching feature pairs where RMM failed due to not finding three pairs. Orange circles = centre of matched features.

For the benchmarking datasets, we selected 10 different targets from the DUD-E database [28,29] and just used the known active ligand sets. For these molecules, up to 25 conformations were created using LigandScout's iCon conformer generator [11]. For use as query pharmacophores, structure-based models were created using DUD-E's provided receptor and co-crystallized ligand structure, removing excess features until hits could be found. The exclusion volumes that were created in the modelling process were kept. These pharmacophores were not further optimized for practical use and only served for benchmarking the accuracy and performance of the methods, disregarding the selectivity of actives vs. decoys (see Section 3.1). Images of the pharmacophores can be found in our supplementary materials. The used DUD-E datasets and the number of pharmacophore features (after removing excess ones) are shown in Table 1.

Table 1. Summary of benchmarking datasets.

Name	Pha. Features	Molecules	Conformations
ACES	6	664	15,391
CDK2	6	798	17,740
DRD3	6	877	20,539
EGFR	6	832	19,899
FA10	5	792	19,786
GCR	9	563	9610
PDE5A	7	706	17,015
PGH1	5	251	4041
THRB	7	861	21,474
VGFR2	6	620	15,041

Note that we included the GCR dataset and used a pharmacophore with nine features to also measure the performance and accuracy with comparatively larger models. Due to this large number of features, the model was more selective than the ones from the other datasets, and fewer found hits were expected regardless of the chosen alignment method.

In the following, all comparisons are made to the RM algorithm [15]. Both methods use the same file back-end and processing pipeline.

The runtime and resulting hits for our virtual screening experiments without omitted features can be seen in Figures 10 and 11. Our G3PS "Fast" variant was able to retrieve hits more accurately than the RM method while still being faster. For the measured datasets, this is a speed-up (time RMM/time G3PS Fast) of approximately 2.3× on average. Using the "Accurate" variant settings, one can sometimes find additional hits at the cost of

a longer runtime. This may be a valid option for users that do not want to miss hits and that are willing to accept longer computation times.

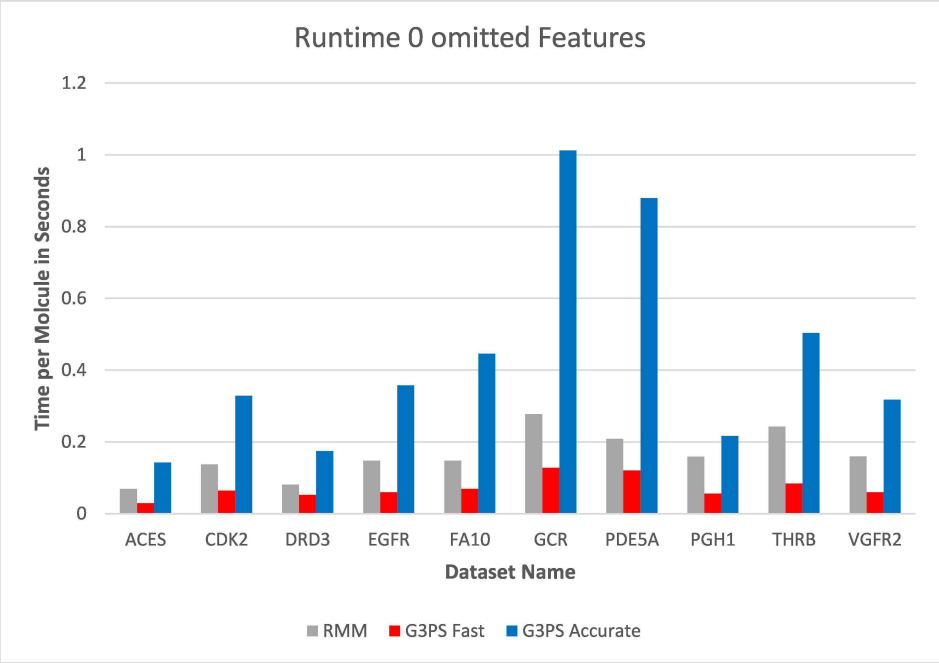


Figure 10. Screening benchmark runtimes without omitted features.

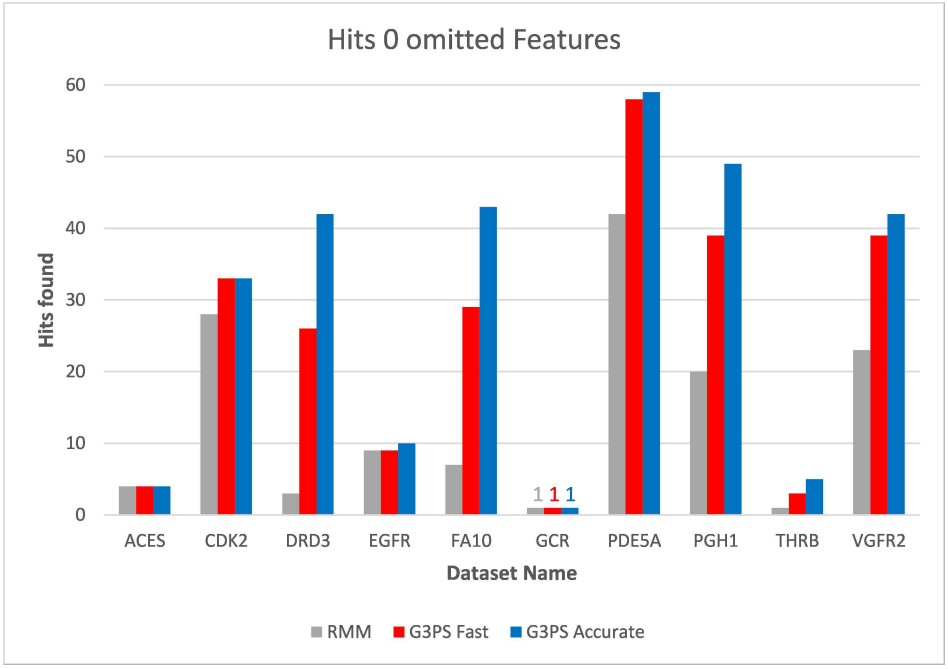


Figure 11. Screening benchmark hits without omitted features.

Starting at one omitted feature (Figures 12 and 13), the runtime of screening with the RM method increased with the number of omitted features, while staying constant with G3PS. Our "Fast" method still provided comparatively more hits while extending its runtime advantage (on average, an 8.5× speed-up). Even our "Accurate" variant was competitive in runtime, only being slightly slower for the PDE5A dataset. It still computed an even larger amount of hits for all datasets, while now being faster than the reference method for all other datasets.

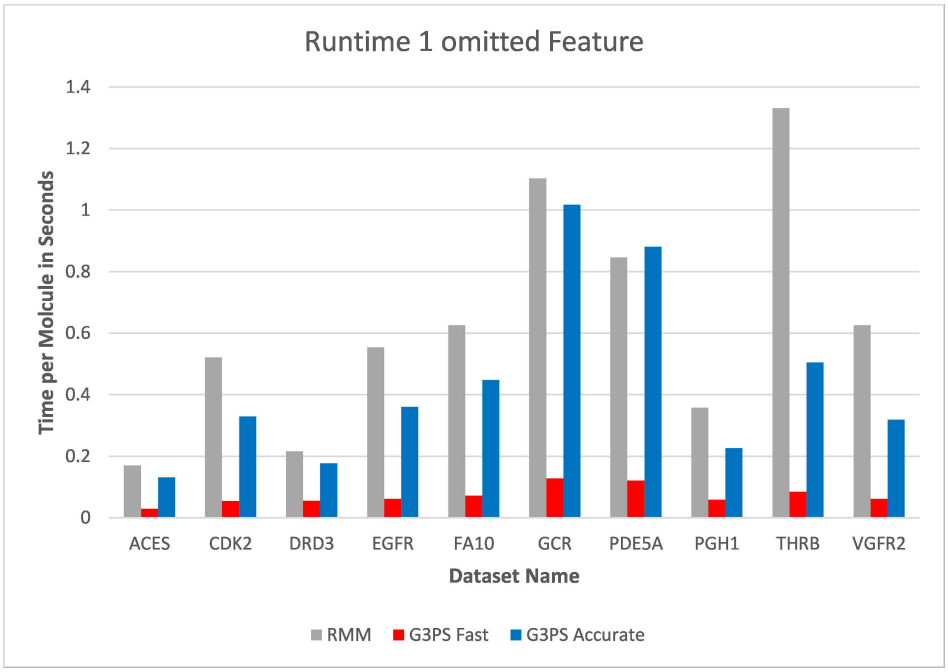


Figure 12. Screening benchmark runtimes with one omitted feature.

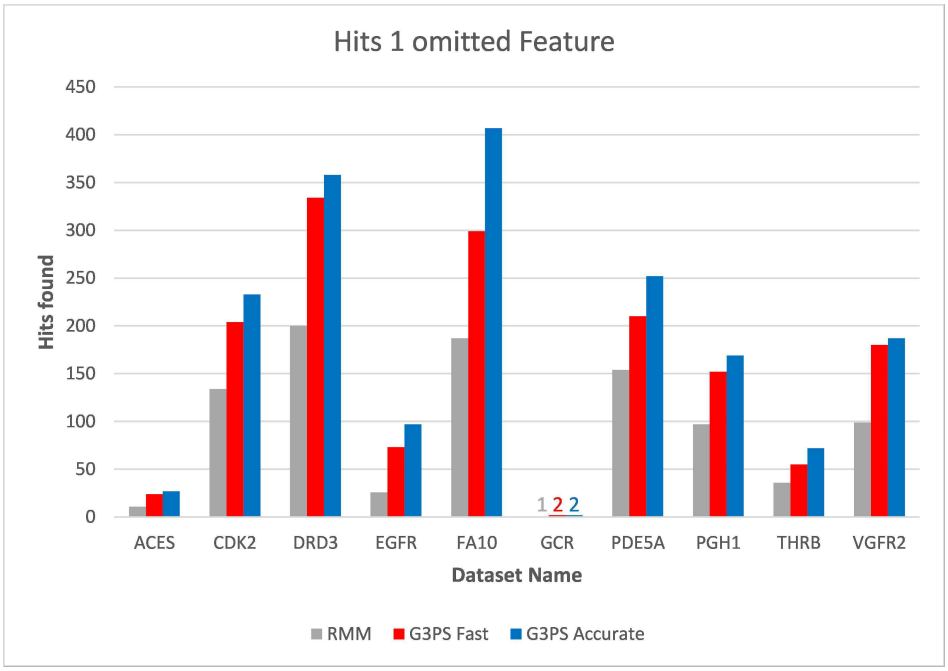


Figure 13. Screening benchmark hits with one omitted feature.

Increasing the number of omitted features to two (Figures 14 and 15) again increased the runtime of the RM algorithm. At this point, both presented presets for G3PS performed better in regard to runtime and the number of found hits. G3PS "Accurate" was now on average 4.6 times faster, and the "Fast" variant was 25.8 times faster.

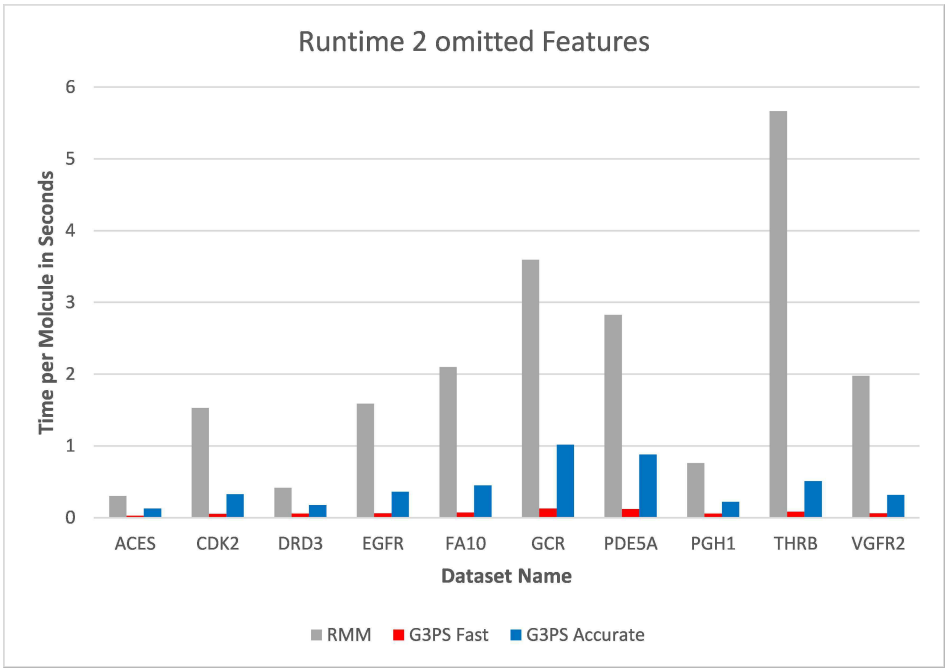


Figure 14. Screening benchmark runtimes with two omitted features.

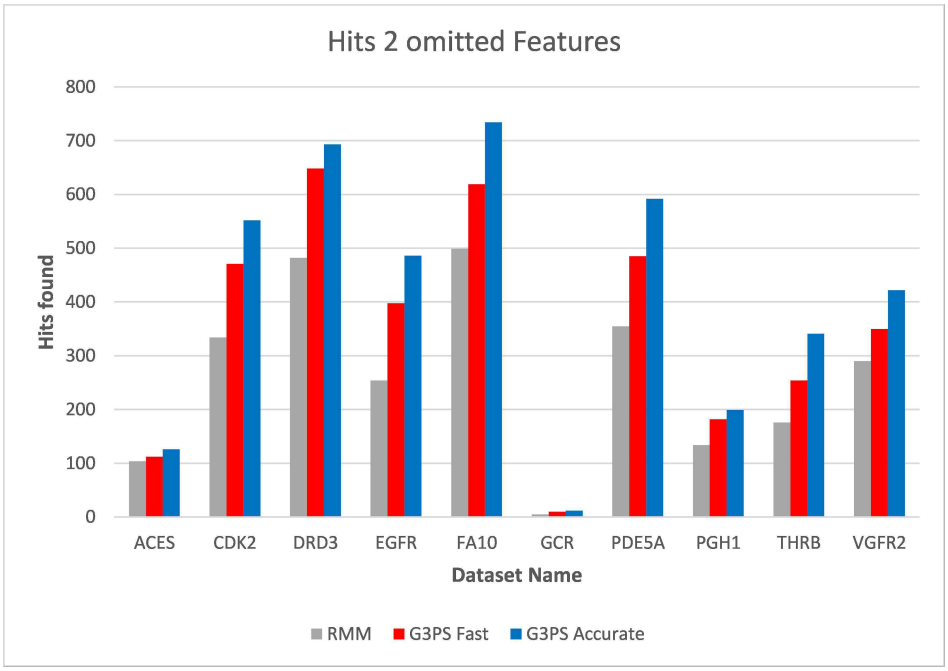


Figure 15. Screening benchmark hits with two omitted features.

As already seen with two omitted features, three omitted features (Figures 16 and 17) continued to show the independence of screening with G3PS to this number. At this point, the RM method starts becoming unfeasible for practical use, like for fragment screening where a larger number of omitted features is necessary. Even though the runtime increases this drastically, it is again not able to retrieve hits as accurately as our "Fast" variant. Our speed-up further increased to 70.7× and 12.3× for our "Fast" and "Accurate" variants, respectively. Two datasets could not be benchmarked with three omitted features, as the pharmacophores only contained five features, and removing three of those would result in a pharmacophore for which a defined, unambiguous 3D alignment is not possible.

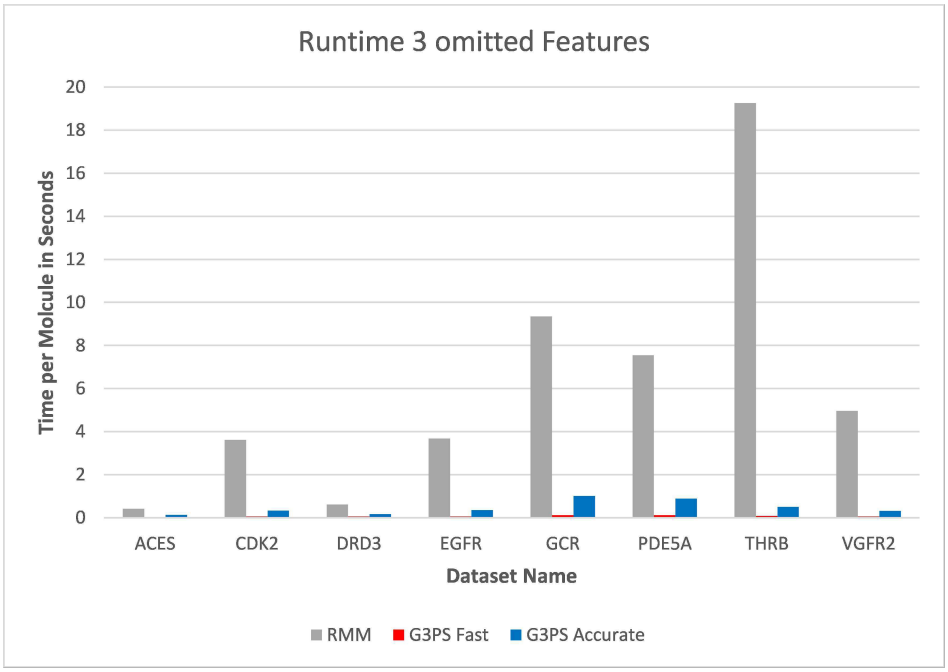


Figure 16. Screening benchmark runtimes with three omitted features.

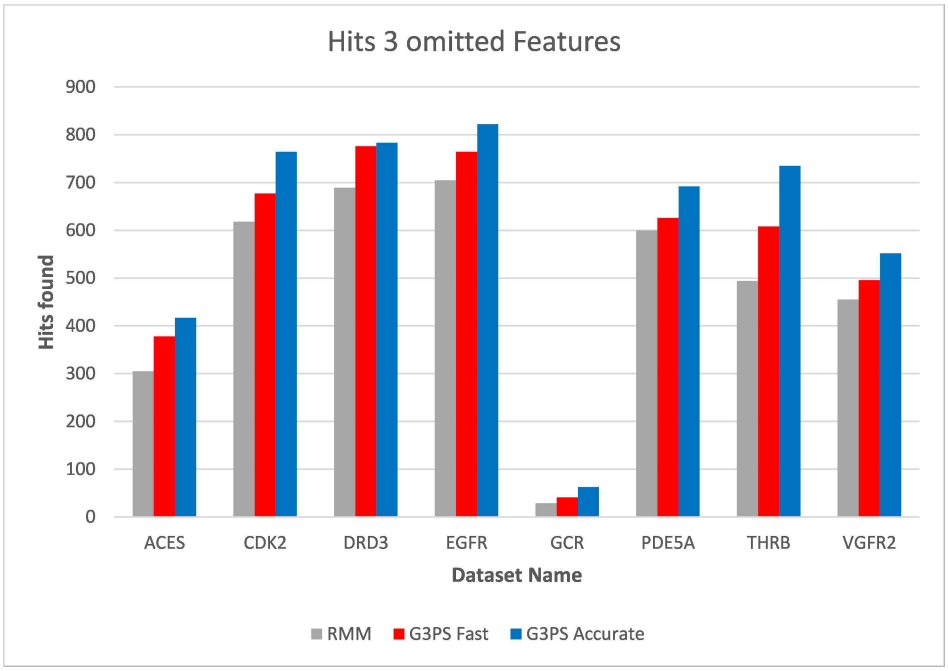


Figure 17. Screening benchmark hits with three omitted features.

3.3. Summary

The Greedy 3-Point Search method for alignment brings a variety of benefits over existing methods. With translations applied in the iterative refinement and the general method of collecting additional pairs based on three starting pairs, our algorithm was able to optimize for maximum matching feature counts. In this way, alignments are not missed that should actually be possible for the two given pharmacophore models. As the algorithm looks for the largest possible match iteratively instead of computing alignments for combinatorially generated feature subsets, the number of omitted features does not impact the runtime, making it orders of magnitude faster than competing methods. With this iterative refinement process, the problem that more omitted features could lead to more found hits than with fewer omitted features due to feature centroid changes is also

avoided. The independence from the omitted feature count makes our method especially suitable for fragment screening where a larger model needs to be aligned to multiple very small fragment pharmacophores, resulting in a high number of required omitted features. The algorithm can also be tuned by the user for accuracy or performance by adapting the number of sampled refinement starting points, while a small number of starting points still finds the best alignment most of the time. Another commonly found weakness of existing methods is that alignments may be discarded because exclusion volumes are slightly touched, even though very minor changes would allow for a valid, possibly even near-optimal solution. G3PS can avoid this problem in the post-processing step by applying minor translations, thereby not dropping such good alignments.

3.4. Future Directions

A logical extension to the iterative refinement process could be to include additional information like feature-direction vectors already in this stage. This would allow for an earlier detection of feature pairings where not all constraints can be met and thus could potentially achieve an improvement in runtime. Alternatively, one could try to consider constraints like feature directions during the alignment itself. This could improve the positioning in space such that these restrictions are more likely to be fulfilled without additional intervention and could thereby reduce the number of guesses needed to find a good alignment.

Further research could also be targeted at extracting parts of the method for GPU computation. A large part of our algorithm consists of fundamental mathematical operations, and additional performance can possibly be gained by using the computational resources provided by graphics accelerators.

4. Conclusions

We presented a new algorithm, Greedy 3-Point Search, for aligning two pharmacophores in a way that the number of matching features is maximized. This avoids the problem of missing valid alignments due to an unfit optimization target, resulting in a higher number of solutions in agreement with the specified positional tolerances of the pharmacophoric features. The new method iteratively refines starting guesses, finding the largest match without having to consider the allowed number of omitted features. For this reason, the runtime of the algorithm does not depend on omitted features but only the number of starting guesses used. This allows a user to find a trade-off between high accuracy of results and low execution time, while still obtaining good results when the fast variant is chosen. For ease of use, we proposed two default parameter settings for the number of starting guesses, which should cover the needs of most users. Additionally, we touched on exclusion volume restrictions and how to recover from alignments that were deemed invalid due to only barely violating exclusion volume constraints.

As our method is able to find hits more accurately, and thereby presents the user more screening hits overall, it will be important to look into how this affects the way pharmacophores are usually refined by expert users. It may now be of benefit to reduce positional tolerances or to narrow down allowed angle deviations for directed features to improve a pharmacophore model's ability to discriminate between active and inactive molecules. Aside from its impact on virtual screening results, the higher probability to find valid alignments will also have an influence on the results obtained by other pharmacophore-based techniques. For lead optimization tasks, the robustness of our algorithm will significantly extend the possibilities when it comes to the structural modification of molecules. Pharmacophore-guided replacements of chemical functionalities by bioisosteric groups or even scaffold hops have a higher chance to succeed due to the largely reduced risk of failures when attempting a realignment of the template pharmacophore. For ligand-based pharmacophore modeling procedures, following a feature-count maximizing alignment strategy will, on average, result in the perception of a higher amount of common training-set ligand features than one would obtain using a pure RMSD-minimizing strategy. Unfor-

tunately, the average impact of our new alignment algorithm on the outcome of these pharmacophore-based techniques cannot be quantified easily since several other methods and case-specific factors are also of influence and need to be accounted for. However, we are confident that taking advantage of the robustness and the feature-count maximizing search strategy of our new algorithm would be beneficial for any pharmacophore-based technique relying on pharmacophore alignment and thus would contribute to its further development.

Supplementary Materials: The following are available online at https://www.mdpi.com/1420-3049/1/1/0/s1, images of pharmacophores used for benchmarking: ACES, CDK2, DRD3, EGFR, FA10, GCR, PDE5A, PGH1, THRB, VGFR2

Author Contributions: Conceptualization, C.P.; methodology, C.P.; software, C.P.; validation, C.P.; formal analysis, C.P.; investigation, C.P.; resources, C.P.; data curation, C.P.; writing—original draft preparation, C.P.; writing—review and editing, C.P., T.S., and T.L.; visualization, C.P.; supervision, T.S. and T.L.; project administration, T.L.; funding acquisition, T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the University of Vienna Research Platform NeGeMac (Next Generation Macrocycles to Address Challenging Protein Interfaces).Open Access Funding by the University of Vienna. (Please confirm this text for Institutional Open Access Program (IOAP): University of Vienna IOAP Discount

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Acknowledgments: We want to express our deep appreciation to Gökhan Ibis and Thomas Kainrad for the discussions we had, ultimately leading to a better method.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Langer, T.; Wolber, G. Pharmacophore definition and 3D searches. *Drug Discov. Today Technol.* **2004**, *1*, 203–207.
2. Langer, T.; Hoffmann, R.D. *Pharmacophores and Pharmacophore Searches*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
3. Leach, A.R.; Gillet, V.J.; Lewis, R.A.; Taylor, R. Three-dimensional pharmacophore methods in drug discovery. *J. Med. Chem.* **2010**, *53*, 539–558.
4. Voet, A.; Qing, X.; Lee, X.Y.; De Raeymaecker, J.; Tame, J.; Zhang, K.; De Maeyer, M. Pharmacophore modeling: Advances, limitations, and current utility in drug discovery. *Journal of Receptor, Ligand and Channel Research* **2014**, *7*, 81-92.
5. Wolber, G.; Langer, T. LigandScout: 3-D pharmacophores derived from protein-bound ligands and their use as virtual screening filters. *J. Chem. Inf. Model.* **2005**, *45*, 160–169.
6. Seidel, T.; Schuetz, D.A.; Garon, A.; Langer, T. The Pharmacophore Concept and Its Applications in Computer-Aided Drug Design. *Prog. Chem. Org. Nat. Prod.* **2019**, *110*, 99–141.
7. Burnett, J.C.; Wang, C.; Nuss, J.E.; Nguyen, T.L.; Hermone, A.R.; Schmidt, J.J.; Gussio, R.; Wipf, P.; Bavari, S. Pharmacophore-guided lead optimization: The rational design of a non-zinc coordinating, sub-micromolar inhibitor of the botulinum neurotoxin serotype a metalloprotease. *Bioorg. Med. Chem. Lett.* **2009**, *19*, 5811–5813.
8. Seidel, T.; Ibis, G.; Bendix, F.; Wolber, G. Strategies for 3D pharmacophore-based virtual screening. *Drug Discov. Today Technol.* **2010**, *7*, 221–228.
9. Shoichet, B.K. Virtual screening of chemical libraries. *Nature* **2004**, *432*, 862–865.
10. Hawkins, P.C.D.; Skillman, A.G.; Warren, G.L.; Ellingson, B.A.; Stahl, M.T. Conformer generation with OMEGA: Algorithm and validation using high quality structures from the Protein Databank and Cambridge Structural Database. *J. Chem. Inf. Model.* **2010**, *50*, 572–584.
11. Poli, G.; Seidel, T.; Langer, T. Conformational Sampling of Small Molecules With iCon: Performance Assessment in Comparison With OMEGA. *Front. Chem.* **2018**, *6*, 229.
12. Friedrich, N.O.; Flachsenberg, F.; Meyder, A.; Sommer, K.; Kirchmair, J.; Rarey, M. Conformator: A Novel Method for the Generation of Conformer Ensembles. *J. Chem. Inf. Model.* **2019**, *59*, 731–742.
13. Vainio, M.J.; Johnson, M.S. Generating conformer ensembles using a multiobjective genetic algorithm. *J. Chem. Inf. Model.* **2007**, *47*, 2462–2474.
14. Friedrich, N.O.; de Bruyn Kops, C.; Flachsenberg, F.; Sommer, K.; Rarey, M.; Kirchmair, J. Benchmarking commercial conformer ensemble generators. *J. Chem. Inf. Model.* **2017**, *57*, 2719–2728.
15. Wolber, G.; Dornhofer, A.A.; Langer, T. Efficient overlay of small organic molecules using 3D pharmacophores. *J. Comput. Aided Mol. Des.* **2006**, *20*, 773–788.

16. Greene, J.; Kahn, S.; Savoj, H.; Sprague, P.; Teig, S. Chemical Function Queries for 3D Database Search. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1297–1308.
17. Dixon, S.L.; Smondyrev, A.M.; Knoll, E.H.; Rao, S.N.; Shaw, D.E.; Friesner, R.A. PHASE: A new engine for pharmacophore perception, 3D QSAR model development, and 3D database screening: 1. Methodology and preliminary results. *J. Comput. Aided Mol. Des.* **2006**, *20*, 647–671.
18. LigandScout. Inte:Ligand GmbH, Clemens-Maria-Hofbaurer-G. 6, 2344, Maria Enzersdorf, Austria. Available online: https://www.inteligand.com/ligandscout/ (accessed on 26 November 2021).
19. Phase. Available online: https://www.schrodinger.com/products/phase (accessed on 28 September 2021).
20. BIOVIA Discovery Studio—BIOVIA—Dassault Systèmes®. Available online: https://www.3ds.com/products-services/biovia/products/molecular-modeling-simulation/biovia-discovery-studio/ (accessed on 28 September 2021).
21. Molecular Operating Environment (MOE). Available online: https://www.chemcomp.com/Products.htm (accessed on 28 September 2021).
22. Kuhn, H.W. Variants of the hungarian method for assignment problems. *Nav. Res. Logist. Q.* **1956**, *3*, 253–258.
23. Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. A* **1976**, *32*, 922–923.
24. Kabsch, W. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. A* **1978**, *34*, 827–828.
25. Grant, J.A.; Pickup, B.T. A Gaussian Description of Molecular Shape. *J. Phys. Chem.* **1995**, *99*, 3503–3510.
26. Hawkins, P.C.D.; Skillman, A.G.; Nicholls, A. Comparison of shape-matching and docking as virtual screening tools. *J. Med. Chem.* **2007**, *50*, 74–82.
27. Taminau, J.; Thijs, G.; De Winter, H. Pharao: Pharmacophore alignment and optimization. *J. Mol. Graph. Model.* **2008**, *27*, 161–169.
28. DUD-E: A database of useful (docking) decoys—Enhanced. Available online: http://dude.docking.org/ (accessed on 29 September 2021).
29. Mysinger, M.M.; Carchia, M.; Irwin, J.J.; Shoichet, B.K. Directory of useful decoys, enhanced (DUD-E): Better ligands and decoys for better benchmarking. *J. Med. Chem.* **2012**, *55*, 6582–6594.
30. Jonker, R.; Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* **1987**, *38*, 325–340.
31. Zhang, Z. Iterative Closest Point (ICP), In *Computer Vision: A Reference Guide*; Ikeuchi, K., Ed.; Springer US: Boston, MA, **2014**; pp. 433–434.

2.3 Conclusion

In this chapter, the novel Greedy 3-Point Search (G3PS) algorithm was presented. The benchmarks show that this algorithm is able to retrieve compounds that fit a pharmacophore model more accurately than root mean squared distance optimizing methods. While G3PS, just like its competitors, is not guaranteed to retrieve all valid compounds, the fundamental change of optimizing mainly for the largest possible number of matched features instead of a secondary goal (such as small RMSD or large volume overlap) avoids many of the pitfalls of the previously available methods.

Due to the competitors' overemphasis on these secondary goals, their found alignment transformations often no longer suit the initial pharmacophore model, as the model's threshold spheres are not (or not sufficiently) taken into consideration. A common case where this occurs is when most, but not all, features can be aligned very precisely. Those methods will then obviously generate a transformation that exactly aligns them, due to the great score this would achieve. If there then are a few additional features, that don't align as nicely, these methods accept that their transformation is worse for those matching pairs, as the resulting overall score is still better than if the alignment adapted strongly for these features. The resulting alignment would be scored higher initially, but when the thresholds are accounted for in a follow-up step, some of the features may not fit the threshold requirements and the alignment must be discarded altogether. Note that this is more likely to occur the larger a compound is and the more chemical features are present, such as for compounds with macrocycles. In contrast, G3PS does not suffer from this problem, as it will adjust the alignment in each step of the iterative refinement process to account for the threshold spheres, such that an underemphasis of "worse" matching features is avoided.

Also, additional pharmacophore model restrictions, such as exclusion volumes specifically, are mostly only considered after the computation of the alignment and used in a post-filtering step by previous methods. Applying those as a filter after finding a transformation often results in discarding a solution, even though only a small translation may be necessary to obtain a valid, albeit slightly worse, result. For this reason, such information is included during the refinement process, once again leading to increased accuracy in finding compounds that fit a given pharmacophore model.

Interestingly, the improved accuracy of G3PS does not come at the cost of runtime. The performed benchmarks show that even without omitted features the proposed method outperforms the current state of the art regarding time for execution. When the omission of features is allowed, the runtime lead grows even larger, which can easily be explained by the method's ability to incrementally improve the transformation in contrast to trying increasing numbers of combinations of matches based on the number of omitted features. This leads to the method's runtime being independent of the number of omitted features, while the competitor needs exponentially more computations based on them.

Chapter 3

Conformer Generation for Small Molecules

3.1 Introduction

Conformer generation for small molecules is the task of finding possible 3D poses, so-called conformations, for the atoms of a molecule, such that they best represent how the given molecule could be observed in reality. Thereby, the most interesting conformations are the ones with the smallest conformational energy, as those are more likely to occur and be found when observing the molecule bound to a protein. The most important criteria for successful conformer generation are the size of the generated ensembles (generally smaller ensembles are preferred), computation time, and accuracy (fitness for purpose/quality of best conformer). [17] A good balance between these three factors is crucial. Too large ensembles are problematic as it is difficult to decide which conformations are actually relevant, long runtimes may make the method infeasible in practice, and generating inaccurate conformations may render the results useless. The computation of accurate conformations of molecules is especially useful when a researcher wants to assess possible 3D interactions without the need of an experimentally determined 3D structure of that molecule.

Conformers are most commonly generated by using either molecular dynamics (MD) simulations or dedicated conformer generation methods. As MD simulations are very computationally demanding [48], specialized conformer generation methods, for high-throughput use, are an important tool in computer aided drug discovery. For this reason, non MD-based conformer generation tools are the focus of this work.

There are vast amounts of commercial [10] and non-commercial [11] tools available, that can compute such conformations. Generally, commercial conformer generators have been shown to be both faster and more accurate than non-commercial ones, which indicates headroom for openly available methods. [10] Still, for a newly created method, the goal should be to not only surpass the openly available state of the art methods but also outperform the commercial-only ones through novel ideas and a highly optimized implementation. As conformer generation is one of the first steps in many drug design workflows, the runtime, accuracy, and ensemble size improvements will have a large positive impact on the scientific community.

Contribution made

For the creation of a faster and more accurate conformer generator, the iCon [35] conformer generator was used as the starting point and improved upon. The choice of the best conformer space sampling method and special consideration of the molecule's structure, thereby further refining the sampling, were the main aspects to improve. This adaptive strategy, which considers the molecule's structure, also has significant benefits when working with and generating conformations for macrocyclic molecules. Additionally, various optimizing techniques, such as caching, were applied to drastically speed up the proposed method. To simplify its use, as well as the use of many other important chemoinformatics algorithms, the implementations are publicly available as part of the CDPKit. [40]

3.2 CONFORGE

High-Quality Conformer Generation with CONFORGE: Algorithm and Performance Assessment

Thomas Seidel[‡*], Christian Permann[§], Oliver Wieder[†], Stefan M. Kohlbacher[‡] and Thierry Langer[†‡§]

[‡]Department of Pharmaceutical Sciences, Division of Pharmaceutical Chemistry, University of Vienna, Josef-Holaubek-Platz 2, 1090 Vienna, Austria
[†]Christian Doppler Laboratory for Molecular Informatics in the Biosciences, Department of Pharmaceutical Sciences, Division of Pharmaceutical Chemistry, University of Vienna, Josef-Holaubek-Platz 2, 1090 Vienna, Austria
[§]NeGeMac Research Platform, Department of Pharmaceutical Sciences, Division of Pharmaceutical Chemistry, University of Vienna, Josef-Holaubek-Platz 2, 1090 Vienna, Austria

* Corresponding author
e-mail: thomas.seidel@univie.ac.at

Abstract

The majority of compounds in the drug-like chemical space show flexibility regarding their three-dimensional structure which, in solution, results in an equilibrium of multiple interconvertible conformational states. Knowledge of the putative bound-state conformation of a molecule is an essential prerequisite for the successful application of many computer-aided drug design methods that aim to assess or predict its capability to bind to a particular target receptor of interest. An established approach to predict bioactive conformers in the absence of receptor structure information is to sample the low-energy conformational space of the investigated molecules and derive representative conformer ensembles which can then be expected to

comprise members that closely resemble possible bound-state ligand conformations. The high relevance of such conformer generation functionality led to the development of a wide panel of dedicated commercial and open-source software tools throughout the last decades. Several published benchmarking studies have shown that open-source tools lag behind their commercial competitors in many key aspects like accuracy in reproducing bioactive conformations, speed of processing, output ensemble size, range of applicability, stability and user-friendliness.
In this work, we introduce the novel open-source conformer ensemble generator CONFORGE, which builds upon proven concepts and algorithms and aims at delivering state-of-the-art performance for all types of organic molecules in the drug-like chemical space. The ability of CONFORGE, and several well-known commercial and open-source conformer ensemble generators, to reproduce experimental 3D structures as well as their computational efficiency and robustness has been assessed thoroughly both for typical drug-like molecules and macrocyclic structures. For small molecules, CONFORGE outperforms all other tested open-source conformer generators and performs comparably well or better than the evaluated commercial generators, both in terms of processing speed and accuracy in the reproduction of bioactive conformations. In the case of macrocyclic structures, CONFORGE achieved the best average accuracy among all benchmarked generators with high statistical significance. To our knowledge, CONFORGE is the first open-source conformer ensemble generator that is able to truly keep up with commercial software in this field and thus represents a valuable addition to the open-source software toolbox for computer-aided drug design.

Keywords

Conformer Generation, Virtual Screening, Computer-aided Drug Design, Free Software, Cheminformatics, Pharmacoinformatics

Introduction

The vast majority of compounds in the drug-like chemical space comprise one or more rotatable bonds and thus offer some degree of variability regarding their three-dimensional (3D) structure. In solution at room temperature, this structural flexibility manifests in an equilibrated mixture of multiple interconvertible conformational states that correspond to local energetic minima on the

potential energy surface. The bioactive conformation of a drug molecule adopted upon binding to the target receptor is usually quite close to one of its low-energy conformers in solution but can also equal a higher energy conformational state if increased structural strain is outweighed by a significant gain in energetically favorable binding site interactions.[1]

Computer-aided drug design (CADD) methods that perform an assessment or prediction of the receptor binding capability of molecules rely on the knowledge of their bound-state conformation. Unfortunately, experimental data on the bioactive conformations of small molecules is only available for a small fraction of the chemical space. For the majority of the molecules - especially those that exist only 'virtually' - bioactive conformations have to be predicated as accurately as possible by specialized *in silico* methods. An established approach for computing potential bioactive ligand conformations without requiring prior knowledge about the receptor structure is to sample the low-energy conformational space of the ligand and derive a representative conformer ensemble. In most cases, some of the low-energy conformers in this ensemble can then be expected to closely resemble bound-state ligand conformations. Since the generation of conformer ensembles is a prerequisite for the application of many CADD techniques like structure-based virtual screening (VS),[2] 3D ligand-based VS,[3,4] ligand-based pharmacophore modeling, pharmacophore-based VS,[5,6] and 3D quantitative structure-activity relationship (QSAR) studies, a plethora of dedicated commercial and open-source software tools emerged during the last decades. Well-known commercial tools for the purpose of conformer ensemble generation include iCon,[7] Omega,[8] ConfGen,[9] CAESAR,[10] Conformator,[11] COSMOS,[12] Molecular Operating Environment (MOE) LowModeMD,[13] MOE Stochastic and MOE Conformation Import,[14] and ForceGen.[15,16] Corresponding open-source tools are, e.g., FROG2,[17,18] Confab,[19] BCL::Conf,[20] RDKit,[21] Ballon DG and GA,[22] and Multiconf-DOCK.[23]

Based on the employed search strategy, applied methods for conformational sampling can be divided into two major categories: (i) systematic and (ii) stochastic approaches.

With the first approach, conformers are generated by the systematic alteration of torsion angles at the rotatable bonds of the molecule. A benefit of this approach over stochastic sampling is that the number of conformers that have to be generated for an exhaustive sampling of the conformational space is exactly defined. Furthermore, the computation of conformers is significantly faster for typical drug-like molecules since multiple conformers can be generated from a single starting 3D structure by just having to perform sequences of simple rigid body

rotations. However, an exhaustive systematic sampling of conformers is usually only feasible for smaller numbers of rotatable bonds due to the combinatorial explosion in the amount of conformers to generate. Nevertheless, the number of rotor bonds which can be handled by this method is still large enough to process most drug-like molecules without any issues.
Stochastic sampling, in contrast, explores the conformational space in a random manner which results in changes of torsion angles at all rotatable bonds at once. Therefore, this sampling strategy is especially suitable for molecules with high numbers of rotatable bonds or flexible macrocyclic ring systems since representative conformer ensembles can be obtained with relatively few iterations. Software tools implementing a stochastic sampling approach[21,22] often employ distance geometry (DG)[24,25] to generate random conformers of the molecule. DG is based on the principle that all possible conformations of a molecule can be described by pairwise atom distance and volume constraints. To this end, lower and upper distance bounds for all pairs of atoms in a molecule are specified in a distance bounds matrix. In addition, a set of tetrahedral volume ranges of four-membered atom groups may be specified in order to enforce planarity or particular configurations of stereogenic centers. The specified distance and volume constraints then serve as input for an embedding procedure that generates a set of atom 3D coordinates matching the given constraints. A specification of reasonable atom pair distance bounds is crucial to obtain proper 3D output structures. Therefore, empirical information like ideal bond lengths, bond angles, and torsion angles is often used to construct the distance bounds matrix. Nevertheless, the 3D structures generated by the embedding procedure are still quite raw and may contain a considerable amount of geometric errors. In order to obtain structurally sound low-energy output conformers, the raw conformers have to undergo an additional structure refinement procedure which can be computationally quite expensive for larger molecules (e.g. iterative force field energy minimization). A further issue with stochastic sampling arises from the fact that it is hard to judge whether the conformational space has been sampled thoroughly enough. Depending on the flexibility of the processed molecule, pursuing the naive approach of sampling a fixed number of unique conformers may then bear the danger of under- or oversampling.
Implementations of both systematic and stochastic sampling often make use of empirical rules and knowledge as well as structural information derived from experimental data like X-ray structures to increase the speed of conformer generation and/or to improve the quality of the

output ensembles. Frequently, empirical data are stored in the form of fragment 3D structure/conformer databases, ring conformer templates or torsion angle libraries which are then used for a fast fragment-based construction of molecule 3D structures or a directed exploration of conformational space employing only torsion angles that e.g. occur predominantly in crystallographic structures. Examples of conformer ensemble generators pursuing a rule-/knowledge-based approach are iCon, Omega, CAESAR, Confab, ConfGen, FROG2, Conformator, COSMOS, BCL::Conf and RDKit.[26]

Irrespective of the conformer sampling approach employed, conformer ensemble generators always have to face the challenge of achieving a balance between the conflicting objectives accuracy (commonly measured as the minimum heavy atom root-mean-square deviation (RMSD) between the experimentally determined bioactive conformation and any of the conformers in the generated ensemble), ensemble size, and processing time. Of course, the ultimate goal is to generate small ensembles that accurately reproduce receptor-bound ligand conformations with diminishing low computational costs. Since these demands usually cannot be met for all objectives at once, varying emphasis may be put on each parameter depending on the targeted application scenario. If accuracy is of utmost importance, the choice of a more thorough but also more computationally expensive sampling algorithm may be adequate. If large numbers of molecules have to be processed (e.g. preparation of databases for virtual screening), smaller ensembles may be preferred in order to reduce the storage consumption of the generated conformers and to speedup subsequent processing steps (e.g. repeated virtual screening runs). If the speed of conformer ensemble generation is essential and possible losses in quality are tolerable, less accurate but computationally more efficient approaches may be given preference. Recently, several studies have been published[11,27,28] which assessed the performance of well-know commercial and open source conformer ensemble generators regarding the objectives - accuracy in the reproduction of bioactive conformations, ensemble size and processing time - using a dataset of 2,859 (2,912 in ref. [27]) high-quality protein-bound ligand conformations extracted from the Protein Data Bank (PDB).[29] The benchmarking results have shown that commercial and non-free conformer generators are clearly ahead of open-source tools in all evaluated performance aspects and substantial improvements on the free software side are required to catch up with leading commercial tools.

Aside from fulfilling the basic quality criteria outlined above, state-of-the-art conformer ensemble generators nowadays have to face an additional challenge imposed by the growing attention flexible macrocyclic systems receive as a new class of promising drug molecules.[30–32] Macrocyclic molecules do not obey Lipinsk's rule of five[33] but exhibit interesting and useful properties which set them apart from the mass of typical drug-like small molecules. These are, e.g., improved metabolic stability[34], the ability to disrupt protein-protein interactions,[35] or a higher cellular permeability due to conformational restriction.[36,37] Furthermore, they are able to bind to proteins which are considered as non-druggable targets due to their lack of hydrophobic cavities which can serve as anchor points for functional groups.[38,39] Drugs based on macrocyclic scaffolds found widespread clinical application as antibiotics (e.g. rifampicin, vancomycin, macrolides), in cancer therapy,[40–43] in immunology and dermatology,[44] to name a few. When it comes to the generation of representative conformer ensembles of macrocyclic structures, algorithms face the challenge of uniformly searching a huge conformational space in an acceptable amount of time whose size increases dramatically with the number of (partially) rotatable bonds in the macrocycle.[45] Recently, several studies were published that benchmarked well-known conformer generators regarding their ability to sample macrocycle conformations with enough accuracy and diversity for common CADD applications.[46–48] The methods employed by these programs likewise can be divided into systematic and stochastic approaches. For example, the conformer generators Omega macrocycle,[49] MOE LowModeMD,[13] MacroModel,[50] Balloon[22] and RDKit ETKDG[26] all pursue a stochastic search strategy. Implementations of systematic sampling approaches are less common - with Conformator,[11] Prime[51] and ForceGen[15,16] being notable examples in this category. A major algorithmic challenge these programs have to face is imposed by the constrained flexibility of rings which prohibits an independent rotation of individual ring bonds. Commonly, this problem is handled by cutting one or more bonds of the macrocycle in order to obtain an open-ring equivalent[11] or multiple non-connected acyclic parts of the ring[51] which can then be sampled in a straightforward systematic way. Afterwards, the generated open-ring conformers are evaluated whether their geometry is suitable for carrying out ring closures. If so, previously cut bonds are re-introduced and a short 3D structure refinement step is carried out to obtain structurally sound conformers of the original ring system.

In this paper, we introduce the novel conformer ensemble generator CONFORGE (Conformer Generator). CONFORGE is fully open-source (GNU LGPL) and available as part of the Chemical Data Processing Toolkit (CDPKit)[52] in the form of a versatile command-line tool as well as a set of classes and functions provided by CDPKit's C++/Python-API. CONFORGE builds upon proven concepts and algorithms for conformer ensemble generation and aims at delivering state-of-the-art performance for all types of organic molecules in the drug-like chemical space. For the computationally efficient and accurate sampling of small molecule conformers, CONFORGE employs a knowledge-based systematic approach which makes extensive use of pre-generated fragment and torsion angle libraries that were derived from experimental 3D structures. For sampling macrocycle conformers, CONFORGE implements a purely stochastic approach based on DG and force field-driven structure refinement. The best suited conformer sampling approach for a processed molecule is either chosen automatically based on the detected presence of a macrocyclic ring system (default behavior) or can be specified by the user in advance for all molecules to process. Furthermore, CONFORGE is able to correctly handle multi-component molecules like salts and mixtures by the automatic separate generation and later combination of individual component conformer ensembles. CONFORGE's ability to reproduce experimental 3D structures as well as its computational efficiency and robustness has been assessed thoroughly both for typical drug-like molecules (Platinum Diverse Dataset[27]) and macrocycles (208 macrocyclic structures compiled by Sindhikara et al.[51]). The calculation of various performance metrics and the visual presentation of the obtained results largely follow the established protocol developed by Friedrich et al.[27] with some meaningful extensions for presenting the macrocycle sampling benchmarking outcome. For reference, several well-known commercial (iCon;[7] two modes), non-open-source (Conformator;[11] two modes) and open-source conformer generators (Balloon,[22] RDKit;[26] two modes/parameterizations) were assessed in addition to CONFORGE employing the same benchmarking protocol. The small molecule benchmarking results have shown (see section Results and Discussion) that CONFORGE outperforms all other tested open-source conformer ensemble generators and performs comparably well or better than the commercial generators, both in terms of speed of processing and accuracy in the reproduction of bioactive conformations. When it comes to the conformer sampling of macrocyclic structures, CONFORGE is able to reproduce experimental 3D structures with a higher mean accuracy than

all other tested generators and at the same time produces output ensembles that, on average, are only half of the allowed maximum size. To our knowledge, CONFORGE is the first open-source conformer ensemble generator that is able to truly keep up with commercial software with regard to accuracy, speed of processing, applicability, robustness and ease of use. Given the relevance of conformer ensemble generation as a mandatory preprocessing step for the application of many modern CADD methods, CONFORGE represents a valuable addition to the open-source CADD toolbox that will be highly welcome by the scientific community in the light of the previously present 'performance gap'[28] between open-source and commercial software in this field.

Methods

The following sections provide insight into the algorithms and methods employed in the implementation of CONFORGE and discuss the design decisions taken to achieve a high performance in the reproduction of bioactive conformations and speed of processing for a wide variety of drug-like organic molecules. The sequence of individual processing steps that have to be performed during conformer generation is presented visually by a set of hierarchically linked flowcharts and will be described in detail throughout the text. In general, conformer ensemble generation is a complex endeavor and can fail at nearly all processing stages (e.g. for molecules containing atom types not supported by the forcefield) or take exceedingly long (e.g. for molecules with many rotatable bonds). Internally, CONFORGE thus has to perform a significant amount of error and timeout checks which may cause early termination or require intermediate error correction steps. For the sake of simplicity, error and timeout handling have not been incorporated into the flowcharts and the shown program flow is based upon successful execution of every processing step. Furthermore, in the flowcharts and text, a distinction is made between *compounds* and *molecules* according to the chemical definition of the two terms: A molecule represents a set of covalently bonded atoms where each atom is reachable from any other atom via one or more bonds. Compounds may consist of just one (the common case) or multiple molecules, as is the case for salts or arbitrary mixtures.

CONFORGE has been implemented in C++ and builds on the cheminformatics infrastructure provided by the CDPKit C++ API. For end-users, CONFORGE is provided in the form of a command-line tool called 'confgen' (Table 1 in Additional File 1 provides an overview of the supported options) that can be found in the application directory of CDPKit. For developers,

CONFORGE's functionality is also accessible as a set of classes and functions that are part of CDPKit's C++/Python-API and allow for a seamless integration of CONFORGE into own CDPKit-based applications.

Top-level Conformer Generation Workflow. Figure 1 shows the top-level conformer generation workflow executed for an uninitialized input compound (e.g. read from an input file) to obtain a structurally diverse, low-energy output conformer ensemble. In the first step, a preprocessing of the input compound takes place where the addition of missing hydrogens is performed and required data for the subsequent processing steps are calculated (see section Compound Preprocessing for details). For compounds comprising only a single molecule, the conformer ensemble generated in the next step already represents the output conformers of the compound. For multi-molecule compounds, CONFORGE generates a separate conformer ensemble for every molecule and then arranges combinations of selected molecule conformers in 3D space to obtain the final compound output conformers (see section Multi-Molecule Output Conformer Ensemble Generation).

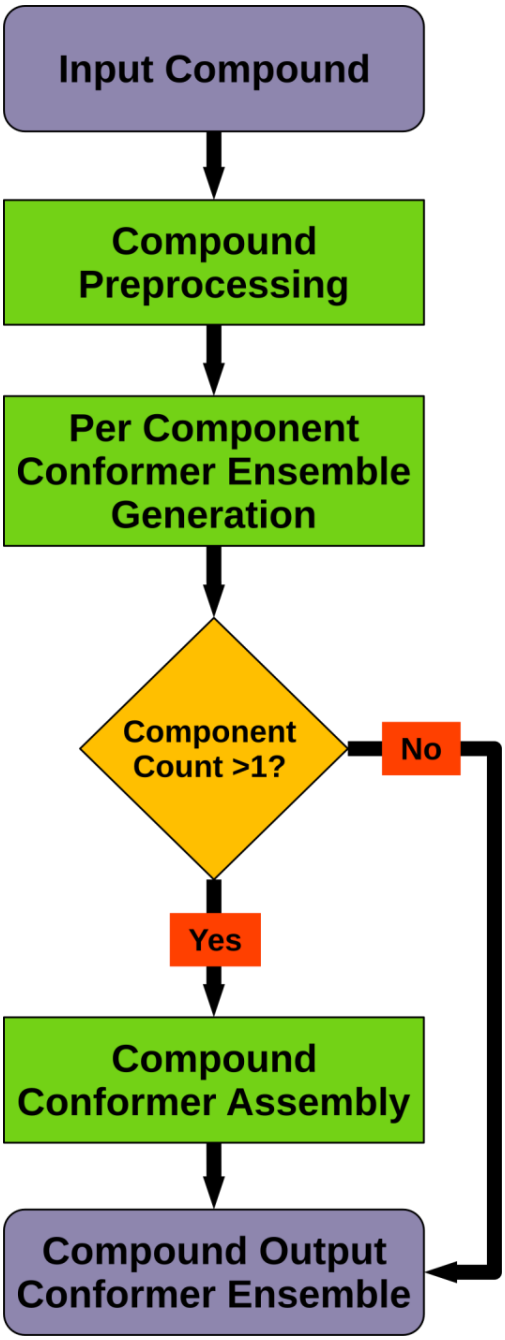


Figure 1. Top-level conformer ensemble generation workflow for a given input compound. Further details on individual processing steps can be found in Figure 2 (Compound Preprocessing), 3 (Per Component Conformer Ensemble Generation) and 13 (Compound Conformer Assembly).

Compound Preprocessing. Compound preprocessing (Figure 2) comprises several sub-steps that prepare the raw input compound and calculate various molecular properties required for the generation of proper 3D structures and conformer ensembles. In the first step, the hybridization state of every atom is determined from the atom type, formal charge and the number and order of incident bonds. In the next step, the smallest set of smallest rings (SSSR) is perceived, which, together with the previously determined atomic hybridization states, allows to identify atoms and bonds that are part of planar aromatics ring systems.

In general, stereo configurations specified for the atoms and bonds of the input compound are retained and will be considered accordingly in the 3D structure generation process. For chiral atoms and asymmetric double bonds with undefined stereochemistry, an attempt is made to calculate missing stereo descriptors from supplied 3D atom coordinates. If 3D atom coordinates are not available, the configuration of chiral atoms is left unspecified, leading to the selection of a configuration that turns out to be energetically favorable in the conformer generation process. For asymmetric double bonds with no predefined stereochemistry, a *trans* configuration of the sterically most bulky substituents on either side of the double bond is selected. To efficiently estimate the steric bulk of substituents, we use a modified version of the Morgan algorithm[53] that stops after the iterative calculation of atom connectivity values (CV). The higher the CV of an atom, the more complex its neighborhood is, and the more space-consuming the substituent it represents will presumably be.

The next processing step adds hydrogens onto heavy atoms which are unsaturated according to the associated chemical element's formal charge and typical valances.

CONFORGE employs a DG-based approach[24,25] whenever 3D structures need to be generated from scratch. Here, randomly distributed starting atom positions are iteratively refined until a valid local energetic minimum structure has been obtained. Which of the energetic minimum conformers is eventually obtained heavily depends on the order of atoms during the initial assignment of random atom positions. To guarantee an input atom order invariant generation of output conformer ensembles, the connection tables of input compounds need to be canonicalized before any conformers are generated. This canonicalization step is optional and has to be explicitly enabled by the user if possible slight output differences for structurally identical input compounds with varying atom orders are not acceptable. For the connection table

canonicalization task, CONFORGE employs an implementation of the algorithm devised by McKay.[54]

After (optional) compound canonicalization, a perception of structurally separated compound components (= molecules) is carried out. Components are identified by performing a depth-first search for all atoms reachable via a bond path from a given start atom. Found reachable atoms and the start atom belong to the same component and get marked accordingly. The search procedure is then repeated for the next not yet visited atom in the atom list until no more unvisited atoms are left.

In the last compound preprocessing step, a topological distance matrix is generated for every component of the input compound. Topological atom distances are needed to generate DG constraints and for the parameterization of Merck Molecular Force Field 94 (MMFF94) Van der Waals interactions.[55] To determine topological distances, we employ a breadth-first search approach where the length of the bond path from a given start atom to a reachable atom is noted in the corresponding cells of the distance matrix. Alternatively, we also implemented the Floyd-Warshall algorithm[56] to determine the topological distances. However, the Floyd-Warshall implementation was significantly slower than the breadth-first search approach and eventually was given up in favor of the latter.

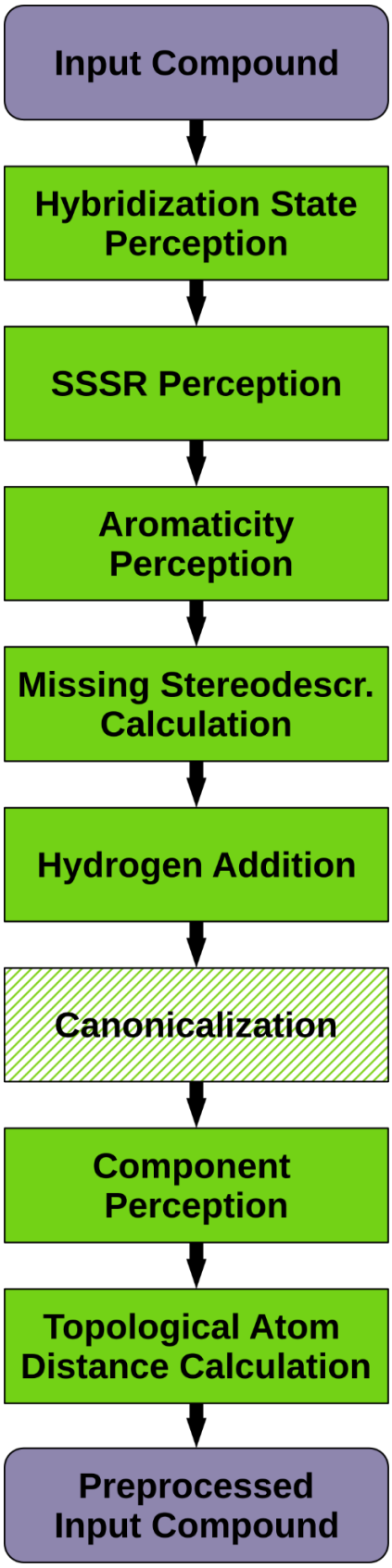

Input Compound
Hybridization State Perception
SSSR Perception
Aromaticity Perception
Missing Stereodescr. Calculation
Hydrogen Addition
Canonicalization
Component Perception
Topological Atom Distance Calculation
Preprocessed Input Compound

Figure 2. Sequence of processing steps carried out to prepare an input compound for conformer ensemble generation.

Molecule Conformer Ensemble Generation. As already noted, a separate molecule conformer ensemble will be generated for every component of the input compound. The molecule conformer generation workflow shown in Figure 3 may thus be executed more than once depending on the number of components that could be identified in the compound preprocessing stage.

The conformer generation process for a given input molecule starts with the perception and parameterization of MMFF94 interactions. Determined force field interactions and parameters will be required for 3D structure refinement and conformer energy estimation in later processing stages. CONFORGE utilizes CDPKit's full-featured MMFF94 implementation whose correctness has been thoroughly validated in all aspects using the datasets provided by the MMFF94 Validation Suite.[57] In the next step, the ultimate decision is made whether to use a systematic or a stochastic sampling approach for generating the molecule conformers. The systematic approach (see section Systematic Conformer Sampling) performs best for typical drug-like molecules composed of chains and relatively rigid ring systems. Stochastic sampling (see next section) has proven to be advantageous for structures comprising flexible macrocyclic rings where conformers cannot be sampled by means of simple systematic torsion driving due to the present rotational restrictions. By default, CONFORGE selects a stochastic sampling approach whenever the SSSR of the molecule contains a ring that incorporates more than ten non-aromatic single bonds. The automatic selection of a suitable sampling method can be overridden by specifying the method to use in advance in the CONFORGE settings. Once a pool of output conformer candidates has been generated by the chosen method, a final output conformer ensemble is compiled in the last processing step of the workflow. Which, and how many, of the candidate conformers end up in the output conformer ensemble depends on various user-specified settings which control allowed energy range, desired structural diversity and maximum ensemble size. Details regarding the output conformer selection process can be found in section Molecule Output Conformer Ensemble Compilation.

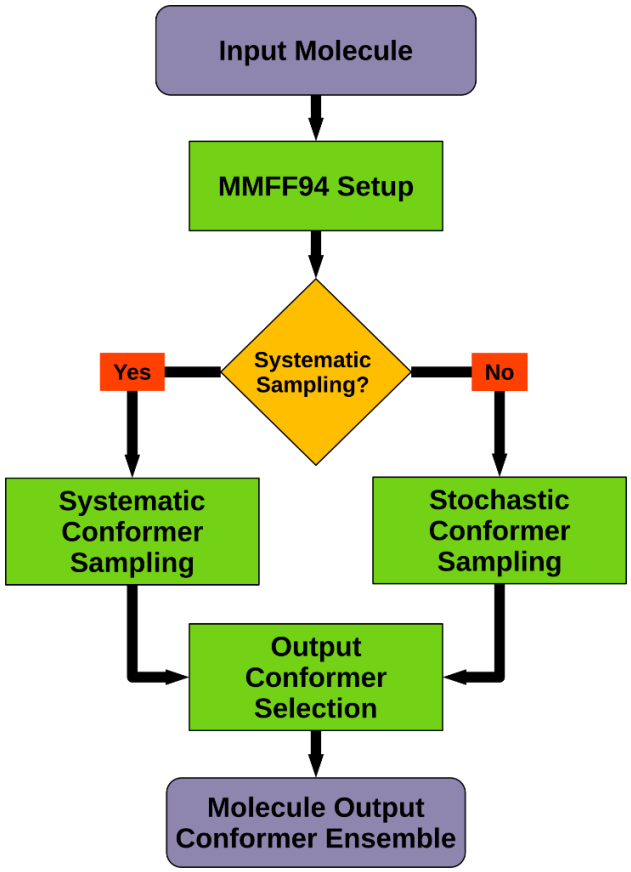


Figure 3. Molecule conformer ensemble generation workflow. Further details on individual processing steps can be found in Figures 4 (Stochastic Conformer Sampling), 6 (Systematic Conformer Sampling) and 12 (Output Conformer Selection).

Stochastic Conformer Sampling. Stochastic conformer sampling is a relatively simple but time consuming process which is based on the assumption that randomly generated structurally diverse low-energy conformers will be uniformly distributed over the whole conformational space of a molecule and that a sufficiently large set of such samples will represent a large fraction of the energetically most favorable torsion angle combinations. Figure 4 shows the principal steps of the stochastic conformer sampling workflow as implemented in CONFORGE. The first step is concerned with initializing the random conformer generation unit. Random low-energy conformers are generated by a DG-based approach, where, in accordance with predefined atom distance and volume constraints, initially randomly distributed atom positions are

successively refined until a valid energy-minimized 3D structure of the molecule is obtained (for details see next section). During the sampling process, random conformers are generated iteratively until a specified maximum number of conformers was sampled (default: 2,000 conformers), the granted sampling time has been exceeded, or the generation of new unique conformers ceased (convergence reached). Within the sampling loop, each newly generated conformer is first tested whether its energy is lower or equal to the current energy threshold. The energy threshold equals the lowest conformer energy encountered thus far plus the specified energy window size. If the generated conformer passes the energy check, it is added to the working ensemble and gets discarded, otherwise. Next, a sampling convergence check is carried out which decides whether the conformational space has been sampled densely enough and, therefore, the sampling loop may already be exited. In the convergence check, after a certain number of conformer generation cycles (default: 100 cycles) have passed, the amount of newly generated unique conformers is determined. This is done by first performing an energy-based removal of duplicate conformers (conformers with an energy difference <= 0.01 kcal/mol) from the working ensemble and then comparing the number of remaining conformers with the ensemble size obtained after the last duplicate removal step. If the ensemble size did not increase since the last check, the sampling process has converged and can be stopped. After leaving the iterative sampling loop, any conformers in the working ensemble with energies outside the energy window (see above) get discarded and the stochastic sampling process terminates.

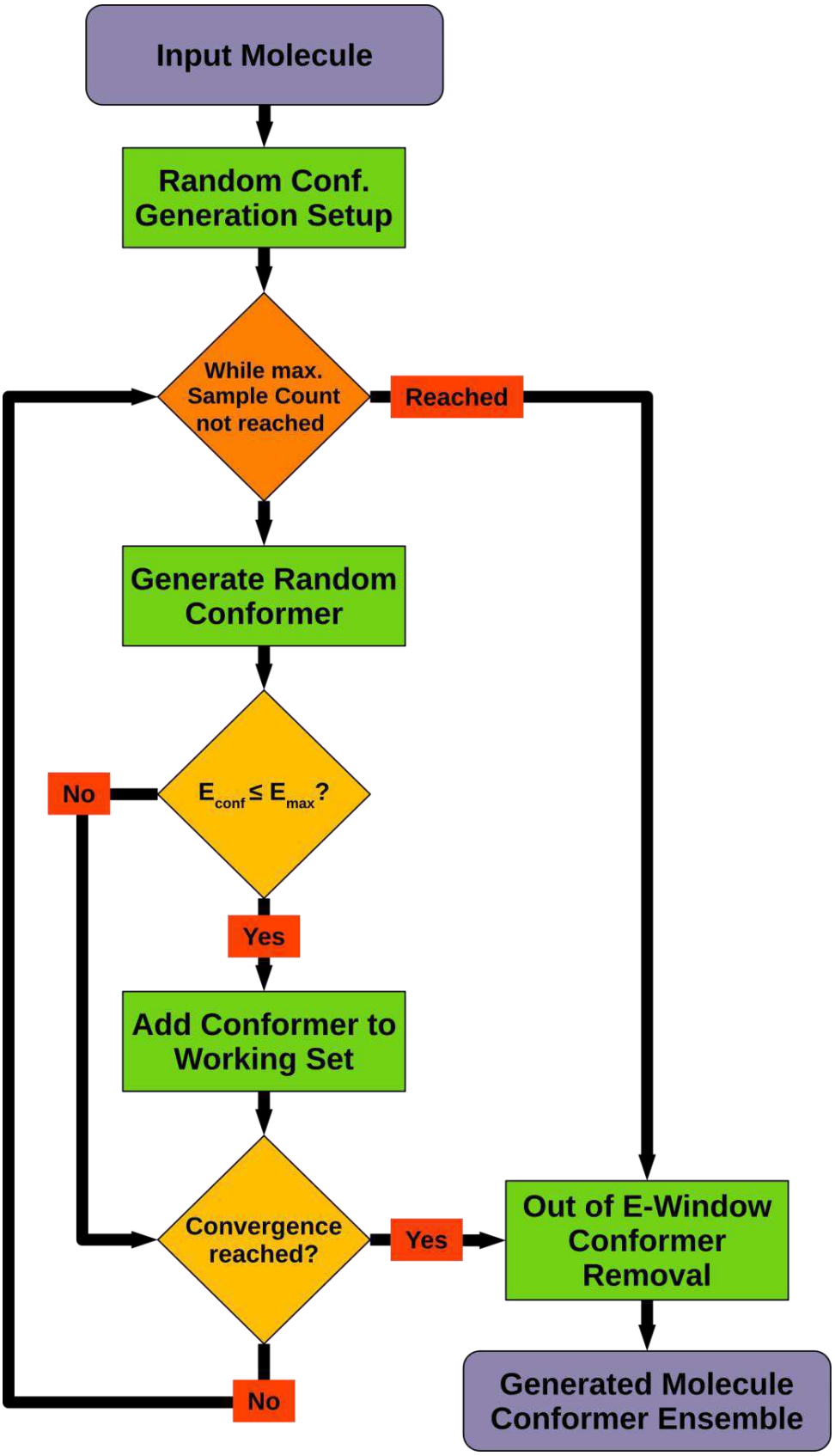

Input Molecule
Random Conf. Generation Setup
While max. Sample Count not reached
Reached
Generate Random Conformer
No
$E_{conf} \leq E_{max}$?
Yes
Add Conformer to Working Set
Convergence reached?
Yes
Out of E-Window Conformer Removal
No
Generated Molecule Conformer Ensemble

Figure 4 Stochastic conformer sampling workflow. E_{conf} = conformer energy, E_{max} = conformer energy threshold according to the current lowest energy conformer and energy window setting. Further details on the random conformer generation step can be found in Figure 5.

Random Conformer Generation. CONFORGE uses the random conformer generation functionality whenever arbitrary low-energy 3D structures of a molecule have to be generated from basic molecular graph information only (e.g. for stochastic conformer sampling, see previous section). The generation of a single random 3D structure comprises the following sequence of steps (Figure 5):

First, the 3D-coordinates of all heavy atoms and any hydrogens which are connected to stereocenters with defined configuration are initialized with random values lying in a range equaling the atom count times a structure type specific factor (e.g. 0.5 for macrocycle sampling). Hydrogens usually represent the majority of the atoms in typical organic molecules and their exclusion allows for a significant speedup of the subsequent DG-based raw 3D structure generation step. Here, the initial random atom positions are optimized by a coordinate embedding procedure [58] until they meet certain pairwise distance and tetrahedral volume constraints. The geometric distance range constraint assigned to an atom pair $p_{ij} = (a_i, a_j)$ depends on its topological distance TD_{ij}: For directly bonded atoms ($TD_{ij} = 1$) the upper and lower distance limit is set to the MMFF94 bond length taken from the assigned bond stretching interaction parameters. The upper and lower distance limit for a geminal atom pair ($TD_{ij} = 2$) is calculated from the respective MMFF bond lengths and the equilibrium bond angle specified by the assigned bond stretching and angle bending interaction parameters. For vicinal atom pairs ($TD_{ij} = 3$), where the central bond is not a double bond with a defined configuration, the lower limit is set to the calculated distance of the atoms in a coplanar and the upper limit to the distance in an anti-coplanar arrangement taking into account the MMFF94 bond lengths and angles of the involved bonds. Vicinal atoms connected to a central double bond with a defined configuration are assigned a fixed distance corresponding to the spacing of the atoms in coplanar arrangement if the central bond's configuration is *cis* (with regard to the atom pair), and the corresponding distance in anti-coplanar arrangement if the configuration is *trans*. All remaining atom pairs ($TD_{ij} > 3$) obtain a lower distance limit which is calculated as the sum of the covalent atom radii

plus an additional safety spacing of 1.5 Å, and an upper limit equal to the total sum of all MMFF94 bond lengths. Volume constraints are generated for atoms bonded to tetrahedral stereogenic centers and for groups with known planar atom arrangements. For stereogenic centers, the sign of the volume spanned by the neighbor atoms depends on the specified configuration and is enforced by setting the corresponding volume range limits to ±0.5 and ±1000 Å³ (empirical values), respectively. Planar groups are e.g. formed by amide Nitrogens, sp^2-hybridized or aromatic atoms with three incident bonds. Furthermore, by atoms connected to amide, aromatic and double bonds. To enforce a planar arrangement of such atom groups, the upper and lower volume range limits are set to zero.

Once a raw 3D structure has been obtained from the embedding process, a first check is made whether the configurations of defined stereogenic centers are correct with respect to the generated coordinates. If the check fails for at least one stereocenter, a new attempt is made to generate a valid raw structure starting from a different set of random atom positions. If, after a certain number of trials (default: 10), still no valid structure could be obtained, the random conformer generation procedure terminates and reports an error. Otherwise, processing continues with the next step, where the coordinates of removed hydrogens are calculated according to the hybridization state of the connected heavy atoms and already assigned atom positions. After that, the geometry of the hydrogen complete raw 3D structure is further refined by iterative minimization of its MMFF94 energy until the energy gradient norm or the change of energy is below a certain target threshold (the stopping criterion and threshold value have to be specified by the caller and are context-dependent). For energy minimization, the algorithm devised by Broyden, Fletcher, Goldfarb and Shanno (BFGS)[59–62] is used. The CDPKit-implementation of the BFGS-algorithm is based on code that has been taken from the GNU Scientific Library (GSL).[63] If the energy-driven structure refinement procedure fails, the random conformer generation procedure terminates immediately and reports an error. In the concluding step of the workflow, all defined stereocenters are once again checked for the correctness of their configurations. If no errors are found, the refined set of atom coordinates is output and the workflow terminates with success. Otherwise, as described previously, a new attempt to obtain a valid 3D structure will be made.

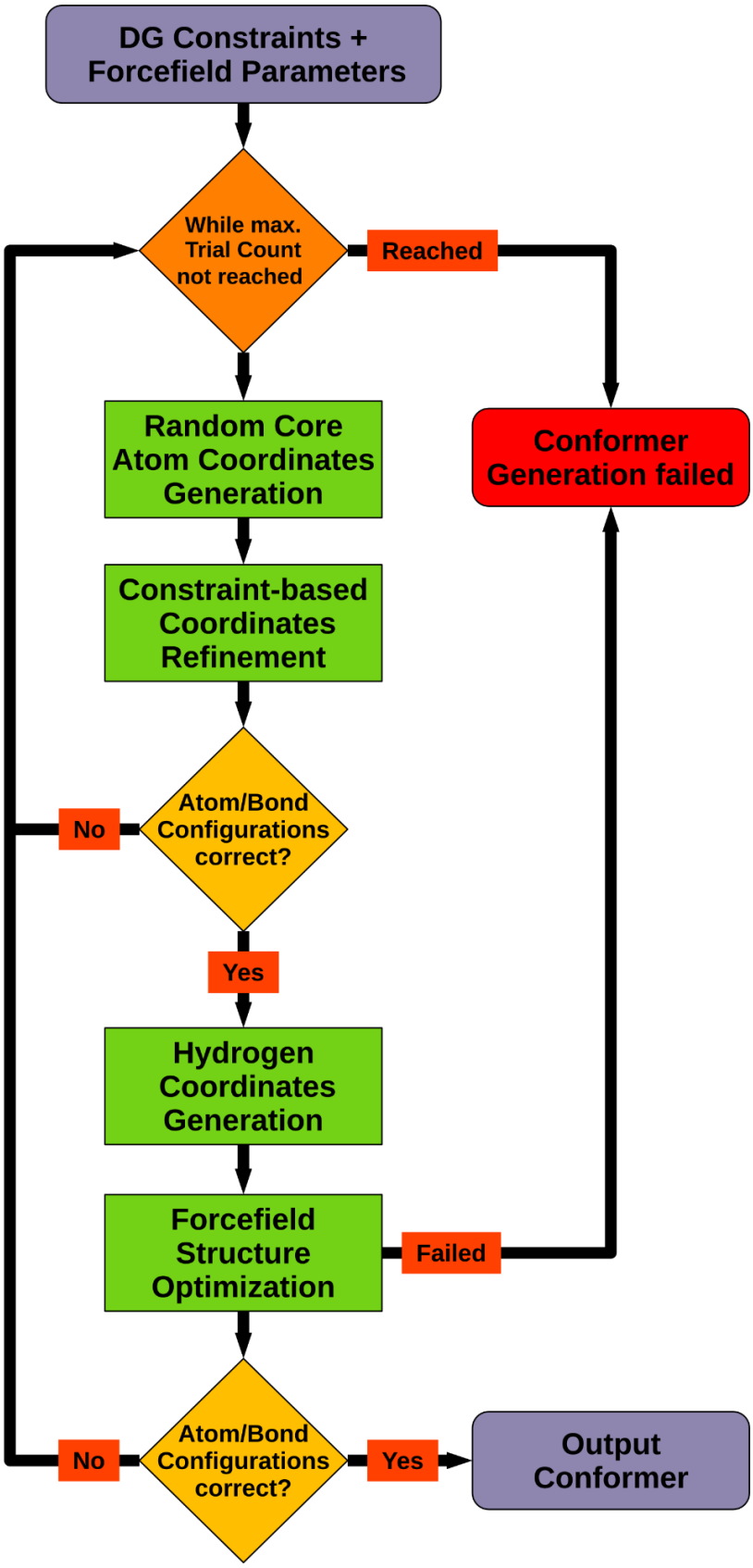

DG Constraints + Forcefield Parameters
While max. Trial Count not reached
Reached
Conformer Generation failed
Random Core Atom Coordinates Generation
Constraint-based Coordinates Refinement
Atom/Bond Configurations correct?
No
Yes
Hydrogen Coordinates Generation
Forcefield Structure Optimization
Failed
Atom/Bond Configurations correct?
No
Yes
Output Conformer

Figure 5. Random conformer generation workflow based on DG and subsequent force field-based structure optimization.

Systematic Conformer Sampling. CONFORGE performs systematic conformer sampling by applying different combinations of torsion angles at the rotatable bonds of the processed molecule on beforehand generated conformer 3D structure templates. Compared with stochastic sampling, the systematic approach usually demands much less processing time for small drug-like molecules since all conformers of a given 3D structure template can be generated by relatively fast rigid body transformations instead of having to repeatedly perform a time-consuming ab initio generation of energy-minimized random conformers.
For the construction of 3D structure templates, CONFORGE employs a fragment-based approach in which the overall 3D structure is assembled from smaller structural building blocks like chain fragments and ring systems. The idea behind this approach is to pre-generate reasonable 3D structures of frequently occurring molecular fragments by an external program ('genfraglib', can be found in the application directory of CDPKit) and store the resulting atom 3D coordinate sets in a permanent library for later use. Via this strategy, a considerable speedup of the 3D structure template buildup process can be achieved since a time consuming ab initio generation of 3D coordinates (see section Random Conformer Generation) can be circumvented for many of the fragments found in typical drug-like organic molecules.
Figure 6 illustrates the major processing steps performed for a given input molecule in the implemented systematic conformer sampling workflow. In the first step, the provided molecule is broken down into smaller structural building blocks by cutting specific bonds that have been identified by a set of fragmentation rules (details on molecule fragmentation can be found in the next section). Afterward a conformer ensemble is generated for each obtained fragment, which, depending on conformational flexibility, may consist only of a single 3D structure or multiple conformers. For rigid fragments like purely aromatic ring systems and chains/moieties lacking rotatable bonds, a single 3D structure is generated. Chains comprising rotatable bonds and flexible ring systems are sampled more thoroughly until a representative set of low-energy conformers has been obtained. As pointed out before, for each processed fragment, an attempt is made to obtain pre-generated 3D coordinates from an on-disk fragment library to speed up the

overall conformer generation process. If a suitable library entry is not available, conformers will be generated on-the-fly using CONFORGE's random conformer generation (see previous section) and torsion driving functionality. More details on fragment library lookup, on-the-fly conformer generation and fragment conformer post-processing can be found in section Fragment Conformer Ensemble Generation.

In the next step, a set of fragment conformer combinations (FCC) is generated. FCCs are represented by *N*-tuples of fragment conformer indices where *N* corresponds to the total number of resulting input molecule fragments. The energy of an FCC is calculated as the sum of the MMFF94 energies of the fragment conformers referenced by the index-tuple. To prevent the exhaustive consumption of main memory and an exaggerated processing time by very high numbers of possible FCCs (e.g. for larger peptides), the FCC generation process stops once a certain amount of stored FCCs has been reached (100,000 in the current implementation). Furthermore, only FCCs with energies lower or equal to the calculated energy threshold are stored for further processing. The energy threshold corresponds to the minimum FCC energy plus the specified energy window size times the safety factor 1.5.

In the final step of the workflow, each FCC serves as a 3D structure template for the generation of derived molecule conformers by systematic torsion driving at rotatable bonds linking the fragments. By means of this divide-and-conquer strategy, it is possible to sample a representative share of the lowest energy conformers in an acceptable amount of time, even in the case of large and flexible molecules. Further details on the assignment of torsion angles to rotatable bonds and the implementation of the torsion driving process can be found in the section Fragment Conformer Combination Torsion Driving.

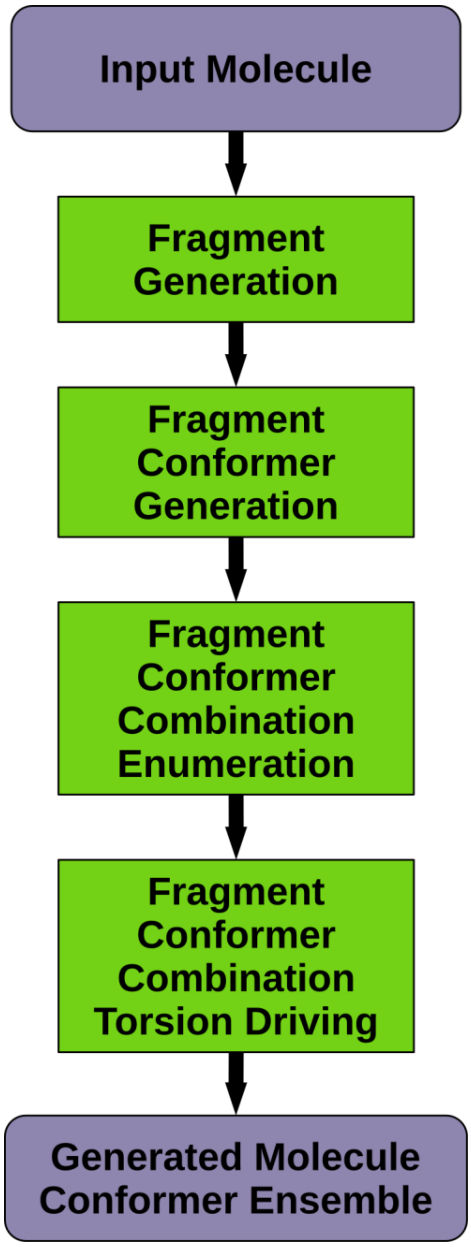


Figure 6. Systematic conformer sampling workflow. Further details on individual processing steps can be found in Figure 7 (Fragment Generation), 8 (Fragment Conformer Generation) and 10 (Fragment Conformer Combination Torsion Driving).

Molecule Fragmentation. As described in the previous section, the systematic conformer sampling procedure generates molecule conformers by assembling selected pre-computed conformers of molecular fragments. These fragments are derived from the molecular graph by

cutting specific carbon-heteroatom bonds and bonds to ring system substituents. The pseudo-code function `isCutBond()` shown in Listing 1, implements the rules by which the bonds being cut are identified:

```
bool isCutBond(Bond b):
  if isInRing(b) or connectsHydrogenAtom(b):
    return false

  if degree(b.atom1) < 2 or degree(b.atom2) < 2:
    return false

  if isInRing(b.atom1) or isInRing(b.atom2):
    return true

  if order(b) != 1:
    return false

  if element(b.atom1) == 'C':
    if element(b.atom2) not in { 'N', 'O', 'S', 'P', 'Se' }:
      return false
  elif element(b.atom2) == 'C':
    if element(b.atom1) not in { 'N', 'O', 'S', 'P', 'Se' }:
      return false
  else:
    return false

  if hasNonSingleBondTo(b.atom1, { 'N', 'O', 'S' }) or
     hasNonSingleBondTo(b.atom2, { 'N', 'O', 'S' }):
    return false

  return true
```

Listing 1. Pseudo-code implementing rules for the identification of bonds to cut in the molecule fragmentation process.

In the resulting fragments, bonds to previously connected fragments of the parent molecule are preserved. Their atoms are replaced by corresponding pseudo atoms encoding chemical element, hybridization state, formal charge and membership in aromatic rings. These pseudo atoms

provide enough local structural context for a later library lookup/on-the-fly generation of proper fragment 3D structures. Figure 7 shows an example of the fragmentation of a typical drug-like organic molecule and the list of resulting molecular fragments obtained by cutting bonds identified with the rules mentioned above.

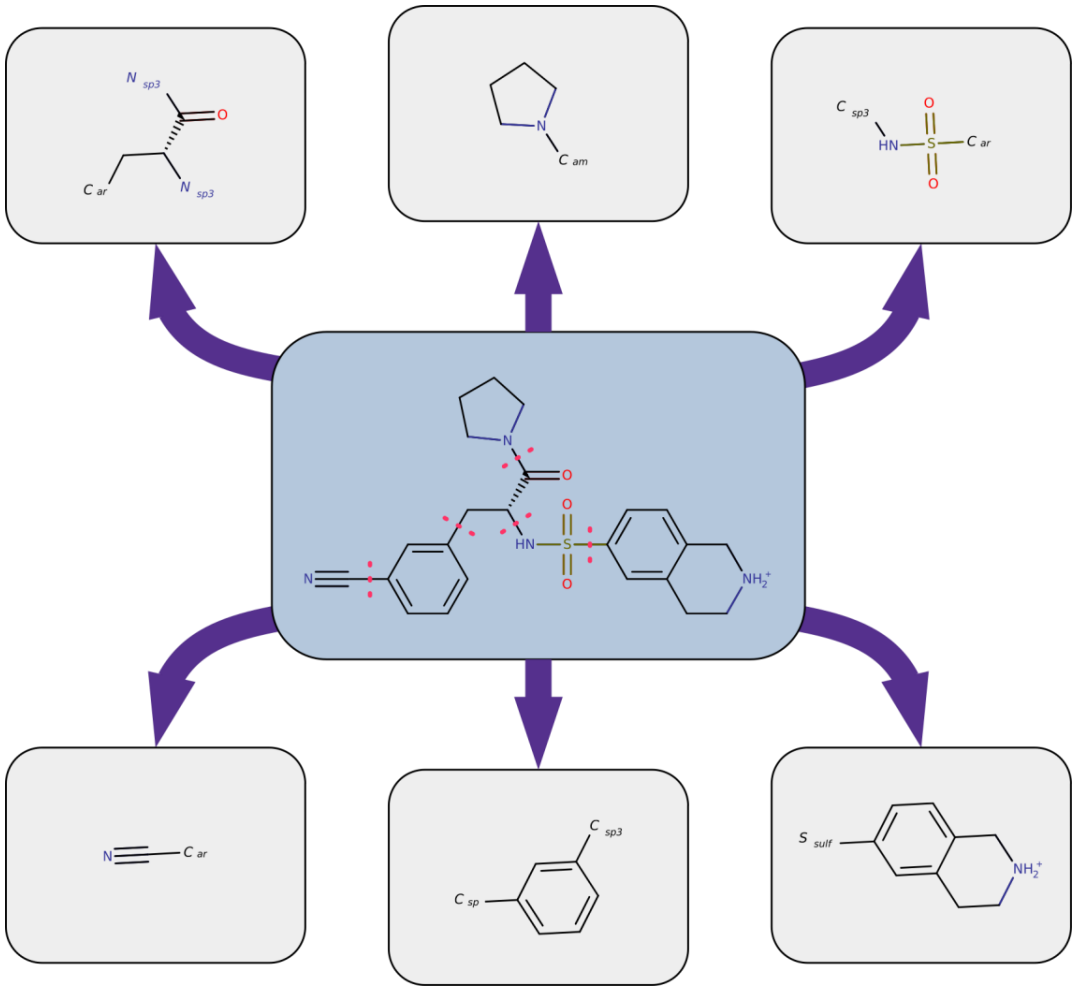


Figure 7. Example for the generation of fragments by splitting specific bonds of a molecule (marked by dashed red lines). Information about the type and nature of formerly connected atoms is carried over to the resulting fragments by the introduction of pseudo atoms which encode chemical element, hybridization state, formal charge and membership in aromatic rings.

Fragment Conformer Ensemble Generation. Depending on specified settings, data availability, and structural type, fragment conformer ensembles are either derived from provided input 3D coordinates, retrieved/derived from an entry in the built-in or user-specified fragment library, taken from the runtime cache, or have to be generated from scratch.

Figure 8 outlines the conformer ensemble generation workflow for a given input fragment. The workflow starts with checking whether the conformers should (by default, input coordinates are discarded) and can be derived from the provided atom 3D coordinates. Supplied atom 3D coordinates are only considered if they are present for at least all heavy atoms of the fragment, and if the fragment is not a flexible ring system or ring conformer enumeration has been disabled. If so, all present 3D coordinates are extracted and a calculation of missing hydrogen coordinates is performed (the hydrogen 3D coordinates calculation procedure is briefly described in section Random Conformer Generation). Afterward, the resulting complete fragment 3D structure is forwarded to the conformer post-processing step (see next section).

If input coordinates should not or cannot be considered according to the initial checks, the workflow proceeds with the fragment canonicalization step that comprises three sub-steps: In the first step, terminal heavy atoms connected to atoms of aromatic rings are replaced by hydrogen. Fragments differing only in their aromatic ring system substituent pattern thus converge into the same canonical fragment structure. This measure helps to increase the fragment library/runtime cache hit rate for quite common aromatic fragments like substituted phenyl rings. The resulting incorrect lengths of bonds between aromatic ring and replaced substituent atoms are corrected later in the conformer post-processing step (see next section).

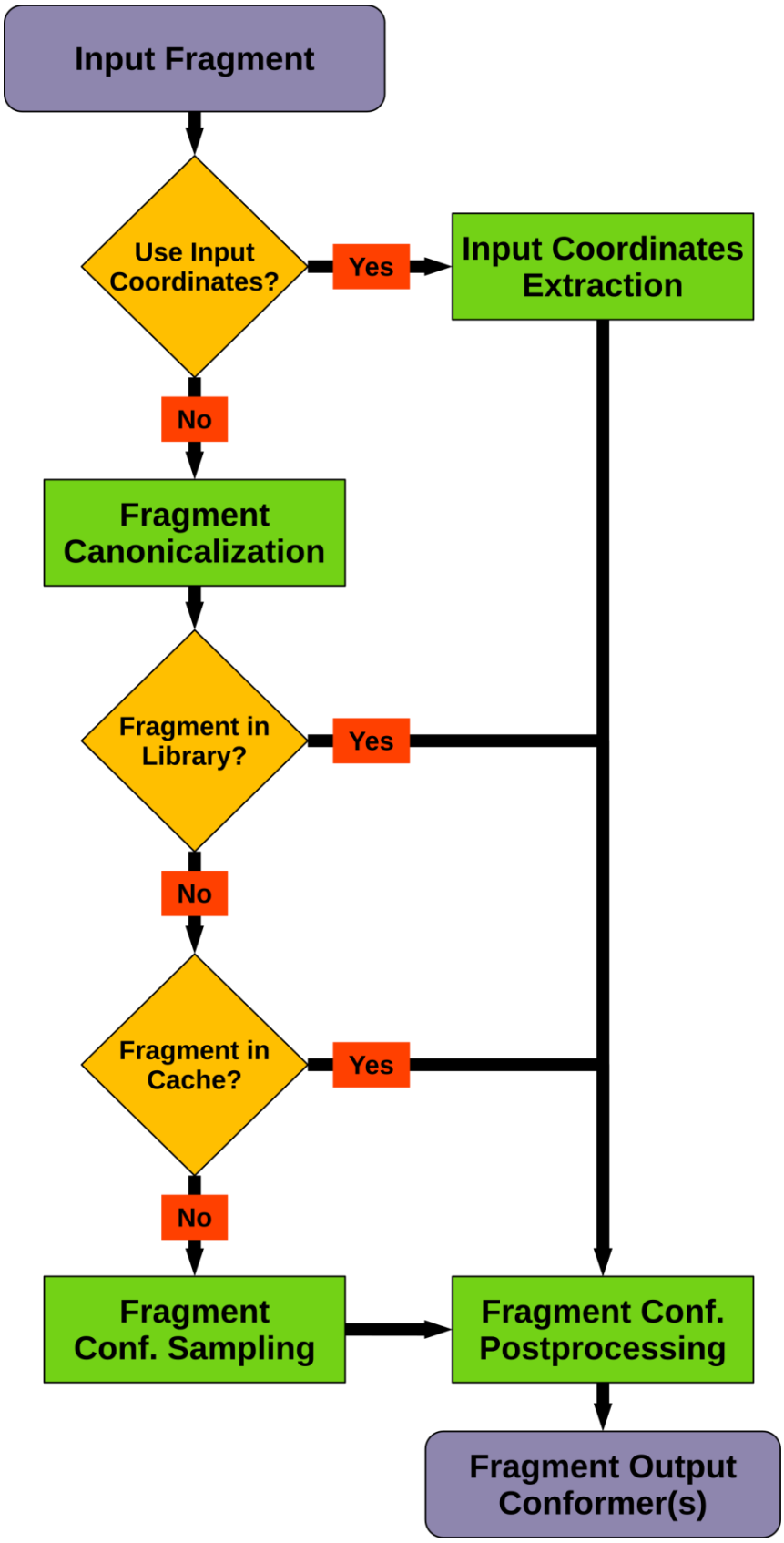

Input Fragment
Use Input Coordinates?
Yes
Input Coordinates Extraction
No
Fragment Canonicalization
Fragment in Library?
Yes
No
Fragment in Cache?
Yes
No
Fragment Conf. Sampling
Fragment Conf. Postprocessing
Fragment Output Conformer(s)

Figure 8. Fragment conformer ensemble generation workflow. Further details on the fragment conformer post-processing step can be found in Figure 9.

In the next step, free valences of pseudo atoms (see previous section) are compensated by adding explicit hydrogens and a calculation of canonical atom labels using CDPKit's implementation of the McKay algorithm[54] is performed.
Finally, a binary representation of the fragment's connection table with atom and bond lists ordered according to the previously calculated canonical atom labels is generated. The calculated SHA1 hash code[64] of the connection table data is then converted to a 64 bit integer key which serves as a unique fragment identifier (ID) in subsequent processing steps.
In the canonical atom labeling and fragment ID calculation procedure, stereodescriptors of defined stereocenters are only considered if the processed fragment represents a ring system. Stereochemistry is disregarded for acyclic fragments since configurations of tetrahedral stereocenters and double bonds can be corrected easily by fast geometric operations in the conformer post-processing step (see next section). Furthermore, for each acyclic fragment only a single random 3D structure is stored in the fragment library and runtime cache to reduce per-fragment memory consumption and thus allow for a higher number of entries. Complete conformer ensembles are then generated in the post-processing step by performing torsion driving on the saved 3D structure. The increased post-processing effort resulting from these measures is condoned to attain higher library and runtime cache hit rates for this structurally diverse fragment type. High hit rates are key for achieving low average molecule processing times by avoiding the relatively slow DG-based random conformer generation procedure for as many encountered fragments as possible.
After the fragment canonicalization procedure, the calculated 64 bit fragment ID is used as a unique key for querying the loaded fragment libraries. Aside from the built-in fragment library which gets loaded on program startup, CONFORGE also supports multiple user-specified external fragment libraries which are searched one by one for an entry matching the supplied fragment ID. If one of the performed library lookups is successful, the fragment conformers deposited in the library entry are extracted and forwarded to the final post-processing step after which the workflow terminates.

Should all library lookups fail, an attempt is made to find a matching entry in the dynamic runtime cache. The runtime cache is implemented as a Least Recently Used (LRU) list (limited to 10,000 entries in the current implementation) and stores conformers of previously processed fragments that likewise could not be found in any of the searched libraries. An identified matching cache entry is then processed in the same way as a corresponding library hit.
If library and cache lookups both fail, the requested fragment conformers are generated from scratch based on the information provided by the molecular graph of the derived canonical fragment. 3D structures are generated by means of the previously described random conformer generation functionality (see section Random Conformer Generation). For acyclic fragments and fragments representing purely aromatic ring systems, just a single output 3D structure is generated. Conformers of fragments representing flexible ring systems are sampled stochastically using a procedure similar to the one described in section Stochastic Conformer Sampling. Depending on actual ring system flexibility, this usually leads to the output of multiple structurally diverse (default ring atom RMSD = 0.1 Å) low-energy 3D structures which cover the conformational space of the fragment in the effective energy window (default values: small ring systems 8 kcal/mol, macrocycles 25 kcal/mol). Afterward, the generated 3D structures are deposited in the runtime cache for potential later reuse and then get post-processed in the workflow's final step.

Fragment Conformer Post-Processing. Fragment conformers resulting from the procedure described in the previous section need to undergo further checks, corrections and modifications before they can be used as building blocks for molecule 3D structure templates (see section Systematic Conformer Sampling). Depending on fragment type and source of the input conformers, required post-processing steps comprise: Correction of aromatic ring substituent bond lengths, inversion of stereocenters for the correction of detected configuration errors, enumeration of invertible nitrogen states and generation of additional conformers by torsion driving. The workflow of the executed post-processing procedure is shown in Figure 9. Herein, the first two processing steps are concerned with required structural corrections resulting from molecular graph modifications and the disregard of stereochemistry for acyclic fragments in the fragment canonicalization procedure (see previous section).

In the first correction step, for every input conformer, the lengths of bonds between aromatic ring atoms and atoms of exocyclic substituents (which were replaced by hydrogen in the fragment canonicalization procedure, see previous section) are scaled to match the corresponding MMFF94 equilibrium bond lengths in the structural context of the parent molecule. In the second step, atom and bond stereocenters of acyclic fragments whose calculated configuration does not match the desired output configuration are corrected by exchanging stereocenter substituent positions and applying geometric transformations on the involved atom 3D coordinates.
Since systematic errors introduced in the fragment canonical procedure do not apply to fragment conformers extracted from a supplied 3D structure of the parent molecule (see Figure 8), the two correction steps can be bypassed for this sort of conformers which is realized by a dedicated check at the beginning of the post-processing workflow.
The next processing step performs a systematic enumeration of all possible invertible nitrogen configuration combinations and generates corresponding 3D structures derived from the supplied input conformers. For a fragment with N invertible nitrogen atoms this will lead to a multiplication of the number of fragment conformers by a factor of 2^N. In the current implementation a non-planar nitrogen atom is considered invertible if it has three single-bonded neighbors, is not a member of more than one ring and is connected to at least two heavy atoms. CONFORGE supports three nitrogen configuration enumeration modes: i) no enumeration - if specified, the enumeration procedure will be skipped and the fragment conformers are left unaltered, ii) only invertible nitrogens with undefined configuration are considered (default setting) and iii) all invertible nitrogens. Algorithmically, the configuration of a nitrogen is inverted by rotating the 3D coordinates of the atoms of a selected substituent to the exact opposite position on the other side of the plane spanned by the bonds to the remaining two substituents. If the inverted nitrogen atom is a ring member, the exocyclic substituent will be chosen for rotation. If the nitrogen is acyclic, the substituent comprising the smallest number of atoms is selected.
If the fragment contains rotatable acyclic bonds, each conformer in the current set will be subjected to torsion driving, further expanding the current fragment conformer ensemble by systematic sampling. Usually, torsion driving only needs to be carried out for flexible acyclic fragments for which just a single low-energy conformer is handed over by the parent workflow (see previous section). Conceptually, the employed torsion driving workflow is largely similar to

the one described for the generation of molecule conformers from a set of FCCs (see next section) and, therefore, will not be outlined in more detail.
After torsion driving has finished, generated conformers which are out of the specified molecule conformer energy window get discarded and the remaining conformers are then forwarded to the last workflow step.
If no rotatable fragment bonds are present, the MMFF94 energy of each conformer using the force field parameterization of the parent molecule is calculated (the torsion driving procedure performs this step internally).
In the last post-processing step, the list of final fragment conformers is ordered by increasing energy and, if necessary, reduced to the maximum allowed output ensemble size (default setting: 10,000 conformers) by removing the necessary amount of high-energy conformers from the end of the list.

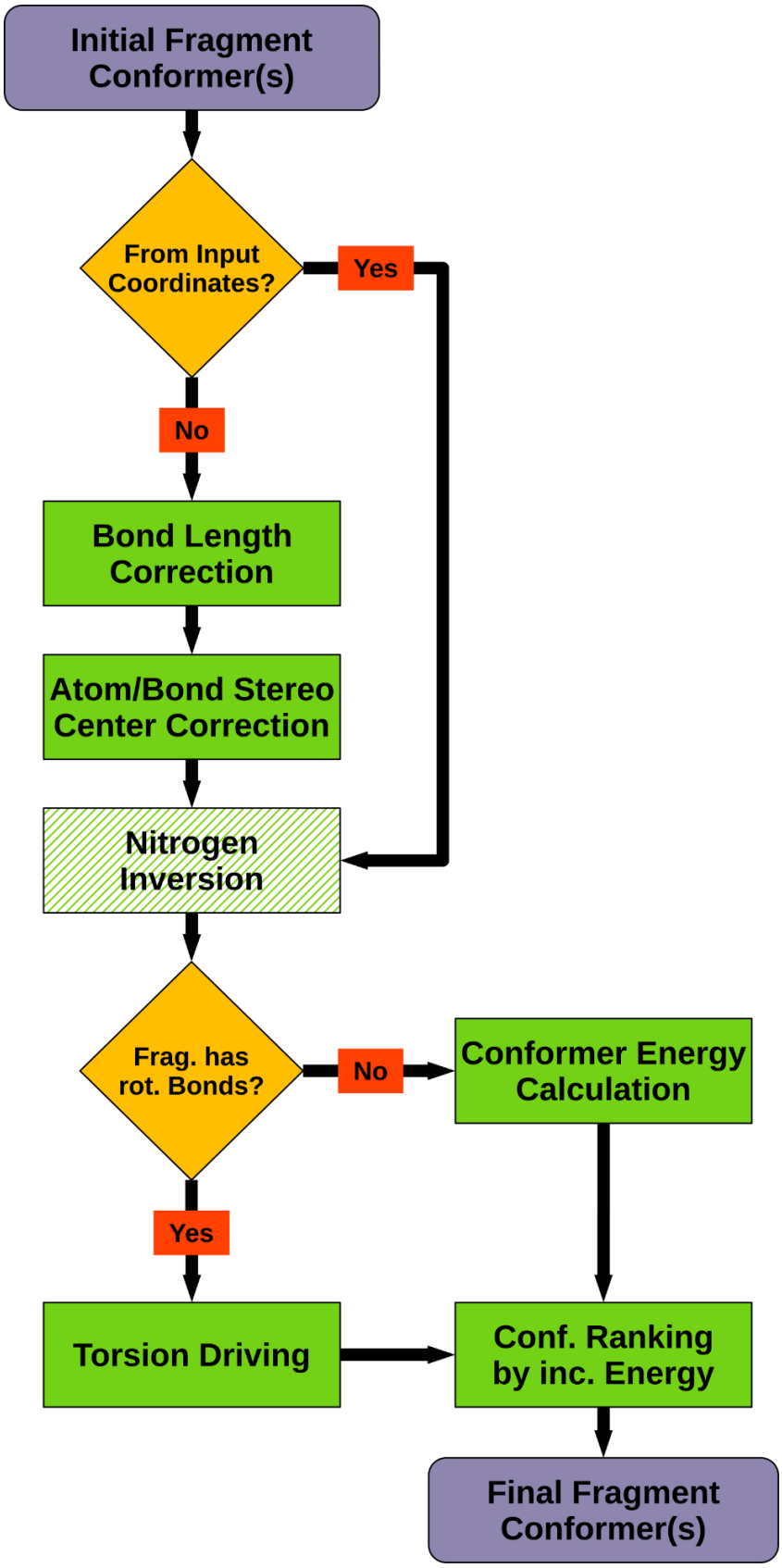

Initial Fragment Conformer(s)
From Input Coordinates?
Yes
No
Bond Length Correction
Atom/Bond Stereo Center Correction
Nitrogen Inversion
Frag. has rot. Bonds?
No
Conformer Energy Calculation
Yes
Torsion Driving
Conf. Ranking by inc. Energy
Final Fragment Conformer(s)

Figure 9. Fragment conformer post-processing workflow.

Fragment Conformer Combination Torsion Driving. This processing step represents the heart of the systematic conformer sampling workflow (see section Systematic Conformer Sampling) and produces an ensemble of output conformers from a set of energy ordered input FCCs. 3D structures of the output conformers are generated by a torsion driving procedure that aligns individual fragment conformers specified by the processed input FCCs along rotatable bonds linking the fragments (see section Molecule Fragmentation). The relative spatial orientation of the aligned fragment conformers is dictated by rotatable bond-specific dihedral angles which are retrieved from matching torsion library entries. Depending on the number of distinct torsion angle combinations possible for the present set of rotor bonds, a corresponding number of output conformers will usually be generated for each processed FCC. For big and quite flexible molecules, the total number of possible conformers can quickly reach rather high figures and an attempt to generate all of these conformers will fail due to excessive main memory and processing time consumption. However, carrying out the processing of FCCs in order of increasing energy and the early pruning of high-energy conformers during torsion driving ensures that also in cases where the conformational space cannot be explored exhaustively, most of the low-energy conformers will be captured and end up in the generated output ensembles (see section Results and Discussion).

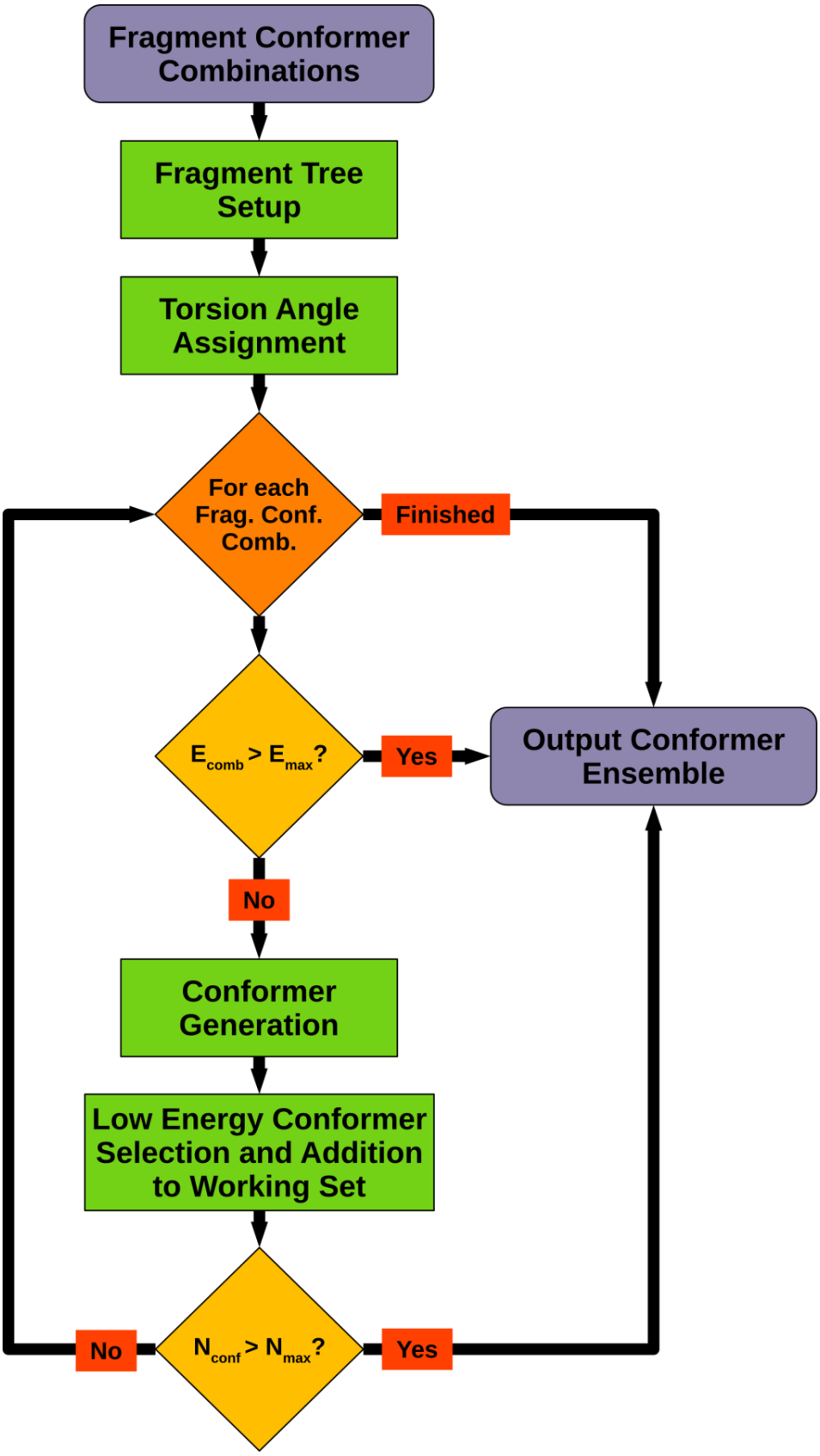

Fragment Conformer Combinations
Fragment Tree Setup
Torsion Angle Assignment
For each Frag. Conf. Comb.
Finished
$E_{comb} > E_{max}$?
Yes
Output Conformer Ensemble
No
Conformer Generation
Low Energy Conformer Selection and Addition to Working Set
$N_{conf} > N_{max}$?
No
Yes

Figure 10. Fragment conformer combination torsion driving workflow. E_{comb} = sum of fragment conformer energies of the current combination, E_{max} = upper energy limit according to the energy of the combination that led to the current lowest energy conformer and energy window setting, N_{conf} = current number of generated output conformers, N_{max} = maximum number of output conformers.

The first step of the overall torsion driving workflow (Figure 10) is concerned with the setup of the fragment tree data structure. This data structure represents the in-memory foundation of the torsion driving algorithm and comprises a set of linked nodes organized in a tree-like manner. Each non-leaf node of the tree references a particular rotatable bond and provides storage for conformers that can be generated by aligning all possible pairs of child node conformers along the referenced bond applying the dihedral angles specified by an assigned torsion library entry. The leaf nodes are associated with the fragments constituting the parent structure (root node) and each stores a particular fragment conformer as specified by the currently processed FCC. For the sake of better understanding, Figure 11 shows an example of a typical fragment tree constructed by the setup procedure for the molecule shown in the root node.

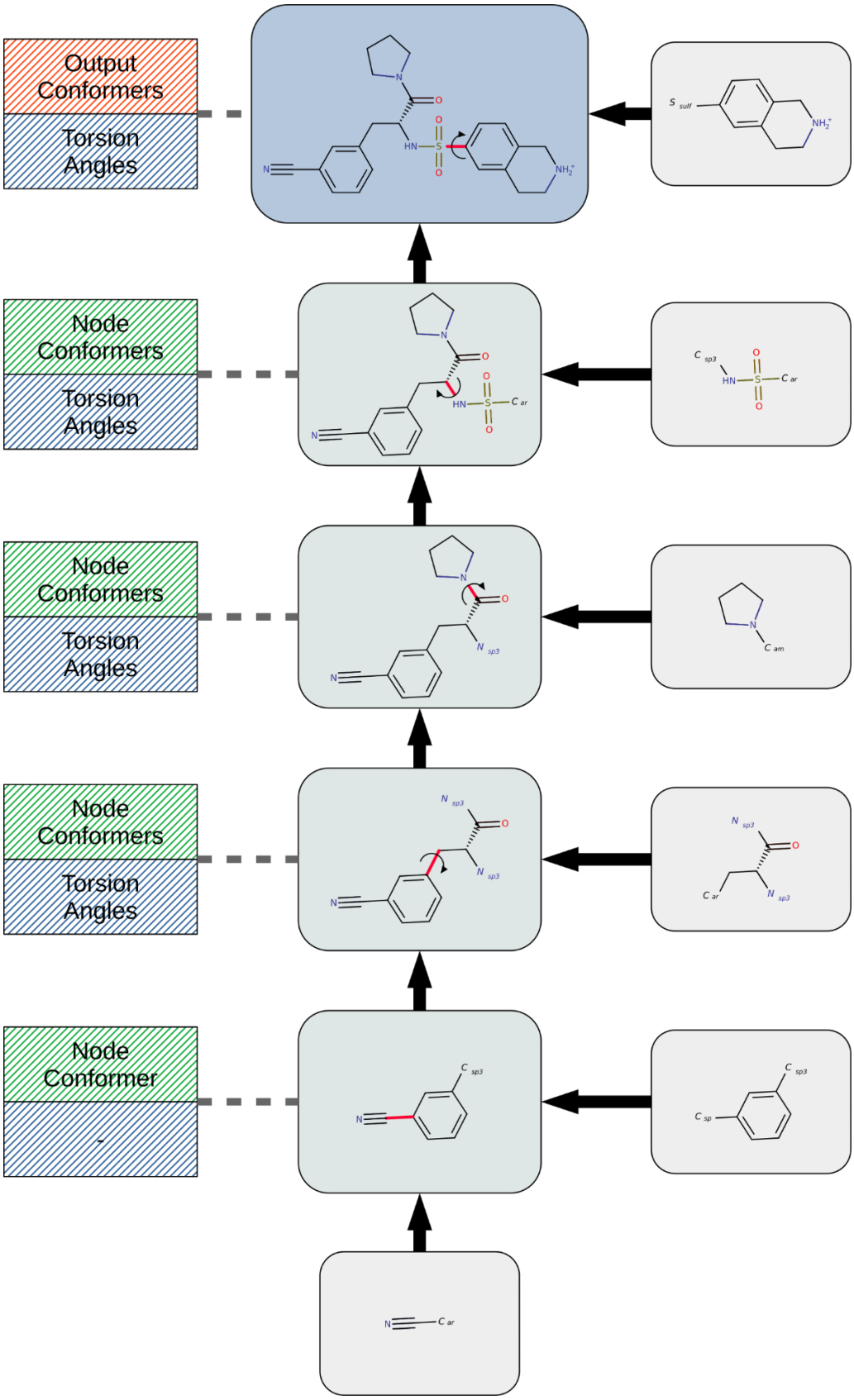

Output Conformers
Torsion Angles
Node Conformers
Torsion Angles
Node Conformers
Torsion Angles
Node Conformers
Torsion Angles
Node Conformer
-

Figure 11. Example of a fragment tree built for the molecule shown in the root node (top of the figure). This data structure serves as the basis for the systematic generation of molecule conformers by performing torsion driving on a given FCC (represented by the fragment conformers stored in the leaf nodes). For every rotatable bond of the molecule (marked in red), a dedicated internal node stores the conformers that can be generated by twisting all possible child node conformer combinations around the rotatable bond according to the dihedral angles provided by the assigned torsion library entry.

Once the fragment tree data structure has been set up, an assignment of proper dihedral angles for the rotatable bonds referenced by the non-leaf nodes is carried out. Here, for each rotatable bond, a lookup for a matching entry in the loaded torsion libraries will be performed. Torsion library entries specify matching rotatable bonds by a single SMARTS pattern[65] which describes a linear path of three or four atoms and provides lists of preferred dihedral angles and associated tolerance intervals. The torsion library used by CONFORGE has been derived from the collection of rotatable bond patterns and corresponding dihedral angles compiled by Schärfer et al.[66] and Guba et al.[67] which resulted from statistical analyses of torsion angle preferences in the drug-like chemical space. Based on the set of rotatable bond patterns in the most recent version of the library by Guba et al., we performed our own torsion angle analysis using a database of experimentally-determined receptor-bound ligand 3D structures extracted from the PDB.[68] Based on the obtained results, several new library entries have been added, and various modifications to the angle and tolerance lists of existing entries in the library by Guba et al. were made. Aside from the built-in library, CONFORGE also supports multiple user-specified external torsion libraries which are included in the search for matching entries. These will take precedence over the entries in the built-in library and thus enable customization of the dihedral angles used in the torsion driving core procedure.

Depending on the conformer generation settings in effect and the presence of local symmetry, dihedral angles provided by the looked-up library entries may undergo further processing before they are assigned to the corresponding tree nodes: i) In the case of rotatable bonds which represent an axis of 1-, 2- or 3-fold rotational symmetry of at least one of the connected fragments, all redundant angles that would lead to the generation of conformer duplicates are

removed. Often occurring fragments with known rotational symmetry (e.g. Trifluormethyl, Phenyl, *t*-Butyl, …) have been tabulated as a list of SMARTS-patterns and are identified at runtime by substructure searching. ii) If torsion angle tolerance range sampling has been enabled (default: not enabled), two additional angles per listed dihedral angle will be calculated that mark the beginning and end of the dihedral angle's first tolerance range (see ref. [67] for the definition of the first tolerance range).

After the torsion angle assignment step, the workflow enters a loop in which a set of output conformer candidates will be generated for each specified input FCC. The loop is exited either after all FCCs have been processed or if one of the two early exit conditions is fulfilled. The first early exit check is carried out before entering the torsion driving core procedure and evaluates whether the energy of the currently processed FCC is above a certain upper energy limit. The energy limit is calculated as the sum of the energy of the FCC that resulted in the lowest energy output conformer generated so far and the specified energy window size. A generation of novel low-energy output conformers from FCCs above this energy limit is very unlikely and the processing of additional input FCCs will have little impact on the final output conformer ensemble. Hence, the main loop can already be exited at this point with, on average, only little losses in output conformer ensemble quality.

If the FCC passes the energy limit check, the torsion driving core procedure will be entered. Herein, the leaf nodes of the fragment tree are first initialized with the fragment conformers specified by the newly processed FCC and then a recursive generation of conformers at the non-leaf nodes is carried out. The conformers of a non-leaf node are generated by overlaying pairs of left and right child node conformers at the referenced rotatable bond and then applying rotations to the coordinates of one child conformer according to the torsion angles assigned during setup. This process is repeated for all possible child node conformer pair and torsion angle combinations and finally results in a set of conformers which represents the conformational space of the molecule substructure covered by the fragments referenced in the subtree rooting at the currently processed node.

The MMFF94 energy $E_{A\text{-}B}$ of a conformer *A*-*B* that was generated from two child node conformers *A* and *B* for the torsion angle Θ can be calculated as (Eq. 1):

$$E_{A\text{-}B}(\Theta) = E_A + E_B + E_{BS} + E_T(\Theta) + E_C(\Theta) + E_{VdW}(\Theta) \qquad \text{Eq. 1}$$

E_A and E_B denote the MMFF94 energies of the child node conformers *A* and *B*, respectively, E_{BS} represents the bond stretching interaction energy of the rotatable bond, E_T represents the sum of the energies of torsion interactions involving the rotatable bond, E_C denotes the sum of the energies of electrostatic interactions between the child node substructures, and E_{VdW} denotes the sum of corresponding Van der Waals interaction energies. As can be seen from Eq. 1, E_A, E_B as well as E_{BS} are torsion angle independent constants whose values, once calculated, can be reused in the calculation of other energy terms they are involved in. The torsion driving procedure exploits this fact and is thus able to perform a fast energy calculation of newly generated conformers since every parameterized forcefield interaction energy term, in the worst case, needs to be evaluated only once per generated conformer.

For each newly generated conformer a first check is made to determine whether its energy is exceeding the current energy threshold which equals the sum of the minimum conformer energy encountered so far and the specified energy window size. If so, the generated conformer is discarded. Otherwise, it will be added to the node's current working set. After all possible conformers of a node have been generated, each saved conformer is again checked whether its energy is within the allowed energy window and if not, gets discarded. Finally, the remaining conformers are ordered by increasing energy and, if the number of conformers exceeds the maximum allowed pool size (default value: 10,000 conformers), the amount of conformers is reduced by pruning excessive high-energy conformers. These post-processing steps ensure that potential high-energy molecule conformers get identified and removed from the processing pipeline already early on and, in case of molecules comprising many fragments and/or high numbers of torsion angles per rotor bond, a combinatorial explosion of live conformers will be avoided.

After the bottom-up generation of conformers has finished at the root node of the fragment tree, the torsion driving procedure terminates. The root node now stores a set of final low-energy molecule conformers derived from the currently processed FCC. For each conformer a check is then made whether its energy is above an upper limit calculated as the sum of the minimum conformer energy encountered so far for all processed FCCs and the specified energy window size. If the conformer's energy is above the threshold, the conformer gets discarded. Otherwise, it is added to the output conformer working set. If, during the processing of the root node conformer ensemble, a new energetic minimum conformer has been encountered, the energetic

minimum gets adapted and all conformers in the current working set which are now above the new upper energy limit are discarded.
After all conformers have been processed, a check is made whether the second early loop exit condition is met. The condition will be fulfilled if the number of conformers in the working set is greater or equal to a specified non-zero maximum pool size (default value: 10,000 conformers). If so, the loop will be exited and the torsion driving workflow terminates. Otherwise, the next input FCC (if available) will be processed as described above.

Molecule Output Conformer Ensemble Compilation. Sets of low-energy molecule conformers obtained from the systematic or stochastic sampling procedures usually do not yet fulfill all user-specified output ensemble characteristics (structural diversity, max. output ensemble size and energy window size) and may contain a significant amount of duplicates or clusters of structurally too similar conformers. Figure 12 shows the workflow of the post-processing procedure which takes care of compiling a final output conformer ensemble with desired characteristics from a set of supplied input conformers. The procedure starts with ranking the supplied input conformers by increasing energy to ensure a preference of low-energy conformers in the later output conformer picking stage. Afterward, a check is made whether a present 3D structure of the molecule that has been provided with the processed molecule on input shall be added to the output conformer ensemble (default setting: not included). If so, the input 3D structure gets extracted and, after calculating its MMFF94 energy, it is added to the output conformer working set.
Processing then enters the main section of the workflow where final output conformers will be selected iteratively from the energy ordered set of input conformers based on mutual structural dissimilarity. The picking loop exits if either all input conformers have been processed, the maximum output ensemble size has been reached or the energy of the currently processed conformer exceeds the energy limit which is calculated as the sum of the minimum conformer energy (= energy of the first conformer) and the specified energy window size. Within the loop a processed input conformer is selected as an output conformer only if the heavy atom RMSD between the current conformer and any of the previously selected output conformers is not below a specified threshold value (default setting: 0.5 Å). The RMSD of an evaluated conformer pair is calculated by performing RMSD minimizing 3D alignments using CDPKit's implementation of

the Kabsch algorithm[69,70] for all possible homomorphic atom mappings that might result from the presence of topological symmetry. To avoid an excessive memory and processing time consumption in case of highly symmetric molecules, the number of processed mappings has been limited to a hardcoded maximum value of 131,072. If this limit is exceeded, further processing of the current molecule will be suspended and a corresponding error is reported.
When the RMSD value resulting from one of the performed 3D alignments falls below the specified threshold, further processing stops and the evaluated input conformer gets rejected. If, otherwise, all performed pairwise RMSD checks have been passed successfully, the conformer is added to the output conformer ensemble and processing continues with the next input conformer in line.
Although the implemented output conformer selection algorithm is quite simple, the obtained ensembles nevertheless fulfill all major quality criteria, which is proven by the benchmarking results presented in the Results and Discussion section.

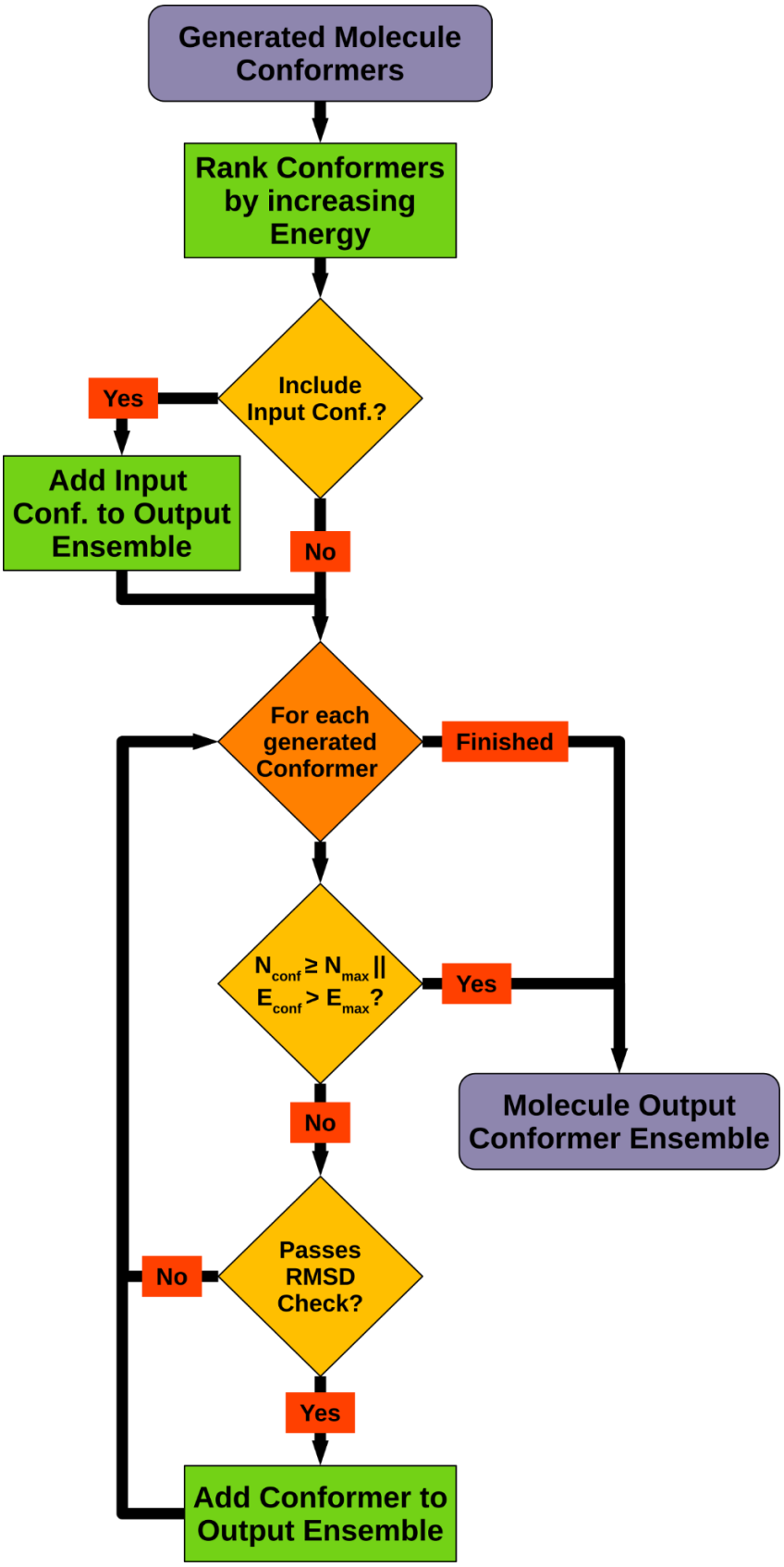

Generated Molecule Conformers
Rank Conformers by increasing Energy
Include Input Conf.?
Yes
Add Input Conf. to Output Ensemble
No
For each generated Conformer
Finished
$N_{conf} \geq N_{max}$ || $E_{conf} > E_{max}$?
Yes
Molecule Output Conformer Ensemble
No
Passes RMSD Check?
No
Yes
Add Conformer to Output Ensemble

Figure 12. Molecule output conformer ensemble compilation workflow based on conformer energy ranking and pairwise RMSD calculations. E_{conf} = conformer energy, E_{max} = upper conformer energy limit according to the current lowest energy conformer and energy window setting, N_{conf} = current number of generated output conformers, N_{max} = maximum number of output conformers.

Multi-Molecule Output Conformer Ensemble Generation. For compounds that consist of multiple components an additional processing step is required which merges the separately generated molecule conformer ensembles into a single compound output conformer ensemble (see also section Top-level Conformer Generation Workflow). The compound conformer generation method implemented in CONFORGE builds composite conformers from sets of selected molecule conformers by lining them up along the X-axis of the coordinate system. For each placed molecule conformer, an axis-aligned bounding box (AABB) is first calculated which specifies the minimum $[X_{min}, Y_{min}, Z_{min}]$ and maximum $[X_{max}, Y_{max}, Z_{max}]$ values of the atom coordinates in each dimension of space. The position $[X_{min}, (Y_{min} + Y_{max}) * 0.5, (Z_{min} + Z_{max}) * 0.5]$ then serves as an anchor point for calculating a translation vector to a particular location on the X-axis. The placement X-position starts at 0 and is incremented by $X_{max} - X_{min} + 4.0$ after each conformer has been placed. The constant 4.0 (in Å) represents an additional safety distance and makes sure that no atom Van der Waals sphere clashes occur between successively placed conformers. For the generation of the ith compound conformer, the molecule conformers at index i of the respective ensembles serve as input and its energy represents the total of the molecule conformer energies. If a conformer at index i does not exist, the molecule conformer with the highest index will be selected instead. Compound conformers are generated until all conformers of the largest molecule ensemble(s) have been consumed. A schematic representation of the compound conformer generation workflow can be found in Figure 13.

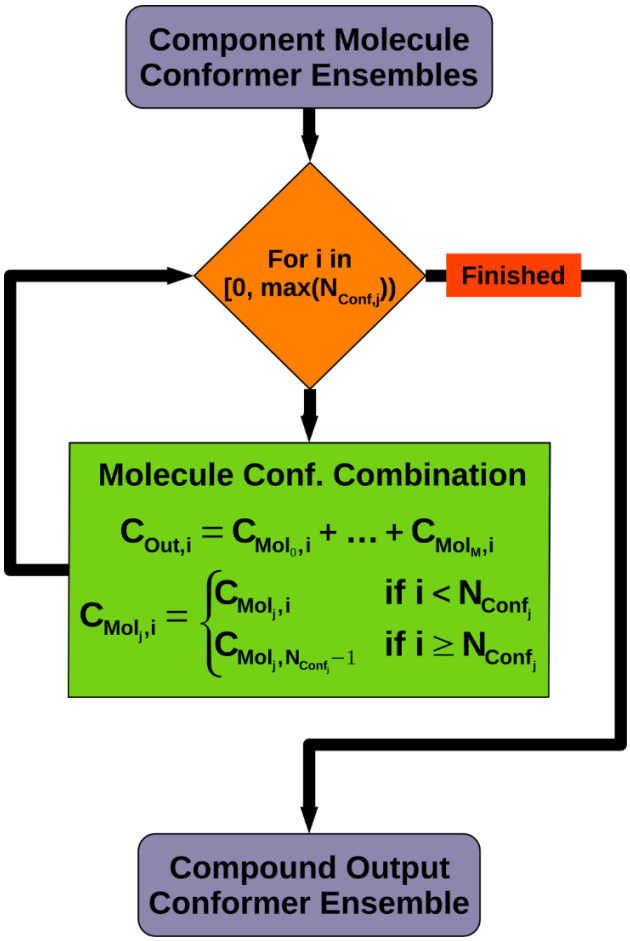


Figure 13. Output conformer ensemble generation workflow for multi-component compounds. $N_{Conf,j}/N_{Conf_j}$= number of conformers of component j, $C_{Out,i}$= ith compound output conformer, $C_{Mol_j,i}$= ith conformer of component j, M = number of components.

Results and Discussion

The ability of CONFORGE to reproduce experimental 3D structures as well as its computational efficiency and robustness has been thoroughly assessed both for typical drug-like molecules and macrocyclic structures. The calculation of various performance metrics and the visual presentation of the obtained results largely follows the established protocol developed by Friedrich et al.[27] with some meaningful additions for the presentation of the macrocycle sampling benchmarks. For reference, several well-known commercial (iCon;[7] two modes), non-open-source (Conformator;[11] two modes) and open-source conformer generators (Balloon,[22] RDKit;[26]

two modes/parameterizations) were assessed in addition to CONFORGE employing the same benchmarking protocol. The corresponding results will be presented and discussed along the ones obtained for CONFORGE in the next sections.

Small Molecule Dataset. For assessing the small molecule conformer sampling performance of the herein evaluated generators the Platinum Diverse Dataset[27] was used. The dataset consists of 2,859 high-quality protein-bound ligand conformations extracted from X-ray structural data provided by the PDB and has been designed and compiled especially for the benchmarking of conformer generators.[11,27,28]

Macrocycle Dataset. To evaluate the conformational sampling of macrocycles, we used a dataset consisting of 208 macrocyclic structures taken from the Cambridge Structural Database (CSD, 130 structures),[71] the PDB (60 structures) and the Biologically Interesting Molecule Reference Dictionary (BIRD) dataset (18 structures).[72] The dataset (from now on referred to as *Prime* dataset) was compiled by Sindhikara et al. for benchmarking the conformer generator Prime[51] (later also used for benchmarking other generators[46]) and contains structurally diverse, challenging macrocyclic structures (featuring disulfide bridges, cross-linking amide bonds, polycyclic rings - including cyclodextrins, polyglycines, cycloalkanes and peptidic macrocycles) with high crystallographic quality (low-temperature factors and/or resolutions). More details about the composition of the full dataset can be found in the supporting information of.[51]

Dataset Preparation. In order to prevent any evaluated conformer generator from taking advantage of the 3D structural information present in the original dataset files, we converted both datasets into isomeric SMILES format by means of a small Python script using functionality provided by CDPKit. The thus generated SMILES files then served as input for the conformer generators in all performed benchmarking runs.

Benchmarking Software and Hardware. The benchmarking runs for each dataset were performed in a fully automated manner using a set of Bash and Python scripts on a dedicated workstation equipped with an Intel Xeon E5-2630 V3 CPU (8 x 2.4 GHz) and 64 GB of DDR3 RAM running CentOS 8 Linux. User interaction was only required for starting the conformer

ensemble generation process and, once finished, for analyzing the produced conformer generator output. In each run, all evaluated conformer generators were executed in sequence without user interaction and indefinite time gaps in between. The source code of the developed benchmarking suite has been licensed under the GNU GPL V2 and is available for download (see Availability of data and materials).

Processing Time Measurement and RMSD Calculation. Measured per-molecule processing times and total program execution times were averaged over five runs to level out runtime differences caused by background system activity and first-time program startup. Furthermore, data was read and written only from/to local hard disk drives. Thus, any influence on I/O-speed and processing times resulting from changes in network latency has been excluded. Depending on availability, per-molecule processing times were either directly extracted from the saved conformer generator log-files in a post-processing step or have been determined indirectly by first time-stamping each line of log-output generated during program execution and then calculating individual molecule processing times from the time-stamp differences of characteristic generator specific log-file lines. In order to eliminate systematic processing time errors due to internal multi-threaded execution (e.g. performed in iCon), single-threaded execution was enforced for all conformer generators by tying started generator processes to a single CPU-core using the Linux command 'taskset'.
The RMSD of generated molecule conformers from the reference 3D structure in the processed dataset has been calculated after performing an RMSD minimizing 3D alignment using CDPKit's implementation of the Kabsch algorithm.[69,70] For 3D alignment and RMSD calculation only heavy atoms were considered and all possible topological symmetry mappings have been taken into account. The lowest RMSD that could be obtained in this way among all conformers of an evaluated ensemble was then reported as 'best RMSD'. Since we encountered slight differences between the conformer ensembles generated by iCon in the five performed runs, calculated best ensemble RMSDs and ensemble sizes were averaged over the five runs for all assessed conformer generators.

Statistical Analysis. Paired Wilcoxon Sign Rank Tests[73] were carried out to check whether the observed differences between the calculated smallest reference 3D structure RMSDs of

conformer ensembles generated by CONFORGE and the corresponding values obtained for the other herein tested generators are of statistical significance ($\alpha = 0.05$). The p-Values were adjusted for groupwise comparisons using the Bonferroni correction.[74] Results of all performed tests together with a short discussion can be found in the Supplementary Information document.

Assessed Generators. Aside from CONFORGE, we benchmarked four additional well-known conformer ensemble generators - Conformator, iCon, Balloon, and RDKit - and compared their conformer sampling performance with the results obtained for CONFORGE.
All assessed generators support different conformer sampling modes, methods, or parameterizations that have an impact on the size and quality of the generated conformer ensembles as well as on sampling speed. Hence, we also considered different modes of operation in the performed benchmarks: Conformator and iCon were both run with two sampling quality/speed settings (best and fast) in the small molecule and macrocycle dataset benchmarks. For Balloon, we evaluated two sampling algorithms and the RDKit conformer generator was assessed using two different built-in parameter set versions in all performed benchmarks. For the sake of comparability of our benchmarking results with the values published by Friedrich et al. (see notes below), the UFF forcefield was chosen to be used by the RDKit conformer generator also in our benchmarks (see Table 1). In the Platinum Diverse benchmarks CONFORGE was run in systematic sampling mode with a) default settings for all other sampling-related parameters ('CONFORGE Systematic Default') and b) enabled exhaustive torsion sampling and defaults for the remaining settings ('CONFORGE Systematic Best'). In the Prime dataset benchmarks stochastic sampling and default settings for all other sampling parameters were used. A comprehensive overview of all evaluated conformer generators, the applied settings and other relevant information can be found in Table 1.
Notes on Omega and other conformer generators: The commercial conformer ensemble generator Omega[8] by OpenEye Scientific Software - which has shown exceptional performance in several previously published benchmarks [7,28] - would have been a perfect additional candidate for inclusion in our benchmarking studies. Unfortunately, our academic software license does not allow for publishing Omega benchmarking results without OpeEyes's explicit permission.[75] To enable at least a partial comparison of CONFORGE with Omega in the Platinum Diverse benchmarks, we provided the benchmarking results (see Table 2) recently published by Friedrich

et al.[11] In our studies we used the same dataset and RMSD calculation method as Friedrich et al. and the results we obtained internally for Omega differed only insignificantly (e.g. deviations between the calculated mean accuracies were 0.012 Å for max. ensemble size = 50 and 0.009 Å for max. ensemble size = 250, respectively). Hardware independent performance indicators of other commercial/open source conformer generators (ConfGenX, ConfGen, CONFECT, MOE, Frog2, etc.) that were determined and published by Friedrich et al. for the Platinum Diverse Dataset[27,28] can be compared with the corresponding values obtained for CONFORGE in the same way as it was done for Omega.

Table 1. Conformer Generators and associated Settings employed in the performed Benchmarking Studies

Generator	Settings	Clustering[a]	Forcefield	Prog. Version
Balloon	DG[b]	RMSD	MMFF94	1.7.0
Balloon	GA[c]	RMSD	MMFF94	
CONFORGE	Systematic Best[d]	RMSD	MMFF94s_RTOR_NO_ESTAT[m]	CDPKit source 2021/02/27
CONFORGE	Systematic Default[e]	RMSD	MMFF94s_RTOR_NO_ESTAT[m]	
CONFORGE	Stochastic[f]	RMSD	MMFF94s_RTOR[l]	
Conformator	Best[g]	RMSD	- / MCOS[j]	1.1.0
Conformator	Fast[g]	RMSD	- / MCOS[j]	

iCon	Best[g]	RMSD	MMFF94s[k]	4.4.7
iCon	Fast[g]	RMSD	MMFF94s[k]	
RDKit	KDG[h]	-	UFF[n]	RDKit source 2020/09/2
RDKit	ETKDGv3[i]	-	UFF[n]	

[a]Conformer similarity measure used in the compilation of diverse output ensembles. [b]Genetic algorithm (GA) disabled (--noGA flag), conformers generated via DG only. [c]GA enabled (default). [d]Enforced systematic conformer sampling mode with larger energy window (20 kcal/mol), lower RMSD-threshold (0.3 Å) and enabled torsion angle tolerance range sampling. [e]Enforced systematic conformer sampling mode using default settings for energy window (15 kcal/mol) and RMSD-threshold (0.5 Å). [f]Enforced stochastic conformer sampling using default parameter settings. [g]Predefined conformer generation modes provided by the developers. [h]Conformer generation using DG embedding parameters for the KDG method.[76] [i]Conformer generation using DG embedding parameters for the ETKDGv3 method including improvements for small rings.[26,76] [j]Macrocycle Optimization Score (MCOS), only used for macrocycle optimization.[11] [k]MMFF94 parameter set variant which enforces planarity of delocalized trigonal Nitrogen atoms.[77] [l]MMFF94s using a refined torsion interaction parameter set.[78] [m]MMFF94s_RTOR excluding electrostatic interaction terms. [n]Universal Force Field (UFF).[79]

Small Molecule Conformer Sampling Performance.

As shown in Table 2, our new conformer generator CONFORGE is able to retrieve the bio-active conformations for the input molecules more accurately on average than all of our competitors. For a maximum ensemble size of 50 the improvement of CONFORGE Systematic Best over all competitors is of high statistical significance ($\alpha = 0.05$, supplementary Table S2), while for a

maximum ensemble size of 250 (supplementary Table S4) we cannot reject the hypothesis that the RMSD results of Conformator Best follow the same distribution as our results. As for the other methods at an ensemble size of 250, we can again claim better accuracy with high statistical significance ($\alpha = 0.05$). This is achieved while beating them regarding runtime, most methods even by more than an order of magnitude. Conformator is able to find better conformations when only regarding the best conformations from their produced ensembles, but as the baseline conformation would not be available in practical applications these better conformations would not be reliably retrievable. Conformator also produces smaller ensemble sizes when run with its 'Fast' presets, which may be generally desirable, but as the user of a given method will have to continue working with the generated conformations the on average worse results are a significant downside. The performance benefits of CONFORGE over Conformator are also large enough to have real-world impact with a speedup of 21.8x on average, when comparing CONFORGE's 'Best' to Conformator's 'Fast' variant, which is the worst-case comparison for our method. The runtimes of our benchmarks also show that no other openly available method can compete with our new implementation regarding time spent, with the best competitor, the commercially available iCon conformer generator with 'Fast' presets, still requiring 3.2x the processing time of our method.

Table 2. Conformer Generator Performance Comparison for the Platinum Diverse Dataset

Generator	Maximum ensemble size 50				Maximum ensemble size 250			
	Mean	**Median**	**Min.**	**Max.**	**Mean**	**Median**	**Min.**	**Max.**
RMSD (Å)								
Balloon DG	0.967	0.788	0.027	4.564	0.806	0.613	0.025	3.996
Balloon GA	0.989	0.858	0.027	4.606	0.833	0.711	0.027	4.336
CONFORGE Systematic Best	**0.669**	**0.486**	0.030	3.919	**0.559**	**0.416**	0.030	3.675

CONFORGE Systematic Default	0.683	0.548	0.036	**3.140**	0.616	0.524	0.036	**2.793**
Conformator Best	0.680	0.589	**0.020**	3.255	0.568	0.474	**0.020**	2.931
Conformator Fast	0.747	0.654	**0.020**	3.511	0.637	0.538	**0.020**	3.329
iCon Best	0.701	0.588	0.030	3.662	0.639	0.569	0.030	3.662
iCon Fast	0.719	0.535	0.030	3.921	0.598	0.474	0.030	3.662
RDKit ETKDGv3	0.711	0.590	0.038	4.386	0.604	0.518	0.038	3.367
RDKit KDG	0.738	0.596	0.038	4.156	0.621	0.515	0.038	3.638
Conformator Best[a]	*0.68*	*0.58*	-	-	*0.57*	*0.47*	-	-
Conformator Fast[a]	*0.75*	*0.66*	-	-	*0.64*	*0.53*	-	-
Omega[a]	*0.67*	*0.51*	-	-	*0.57*	*0.46*	-	-
Ensemble Size								
Balloon DG	49.09	50	-	-	241.55	250	-	-
Balloon GA	38.19	43	-	-	180.33	210	-	-
CONFORGE Systematic Best	39.05	50	-	-	149.26	212	-	-
CONFORGE Systematic Default	28.80	29	-	-	82.07	**29**	-	-
Conformator Best	38.60	42	-	-	167.37	189	-	-
Conformator Fast	**20.53**	**19**	-	-	**71.46**	55	-	-
iCon Best	28.96	32	-	-	71.99	32	-	-

iCon Fast	35.07	50	-	-	122.29	89	-	-
RDKit ETKDGv3	50.00	50	-	-	249.99	250	-	-
RDKit KDG	50.00	50	-	-	250.00	250	-	-
Conformator Best[a]	*38*	*42*	-	-	*166*	*187*	-	-
Conformator Fast[a]	*20*	*19*	-	-	*70*	*54*	-	-
Omega[a]	*34*	*50*	-	-	*118*	*74*	-	-
Processing Time (s)								
Balloon DG	16.939	14.432	0.711	89.403	81.763	70.543	1.285	461.502
Balloon GA	12.596	10.982	0.042	49.986	66.587	58.269	0.472	262.118
CONFORGE Systematic Best	0.164	0.021	**0.001**	**14.295**	0.334	0.07	**0.001**	**16.431**
CONFORGE Systematic Default	**0.102**	**0.012**	**0.001**	19.769	**0.224**	**0.014**	**0.001**	19.942
Conformator Best	4.028	0.344	0.009	268.220	6.579	2.160	0.009	353.579
Conformator Fast	3.581	0.220	0.008	262.060	4.240	0.516	0.008	293.616
iCon Best	0.652	0.199	0.002	64.742	0.755	0.275	0.002	65.130
iCon Fast	0.527	0.174	0.002	30.975	0.652	0.269	0.002	31.132
RDKit ETKDGv3	5.539	2.962	0.102	932.943	27.377	14.903	0.541	3657.464
RDKit KDG	4.075	2.910	0.077	26.434	20.590	14.664	0.423	132.641

The best values for RMSD, ensemble size, and molecule processing time obtained by any assessed generator are written in bold letters. [a]The values listed for Conformator were taken from

ref. [11] and included for reference to demonstrate the correctness of our benchmarking code by the close reproduction of the previously published results for this generator. As a side effect, this allows us to also include the in ref. [11] published results for Omega which we could not assess in our study due to licensing reasons.

It is also important to note that most other conformer generators were not able to handle all of the presented molecules and failed to produce any results for some molecules. The overall program execution times and number of molecules for which processing failed are listed in Table 3. Only iCon was also able to compute conformers for all molecules regardless of ensemble size, while again, all competitors were significantly slower than our method. The computation of the conformers for the whole dataset took the more thorough CONFORGE Systematic Best variant under 9 minutes while the commercially available iCon took 26 minutes with the 'Fast' variant. Freely available algorithms required at least 2 hours and 50 minutes (Conformator Fast) with many of them taking significantly longer than even that method. This comparison nicely visualizes the real impact of these runtimes on a researcher whose work depends on the generation of conformers.

Table 3. Total Program Execution Times and Molecule Processing Failures recorded for the Platinum Diverse Dataset

Generator	Maximum ensemble size 50		Maximum ensemble size 250	
	Total execution time (hh:mm:ss)[a]	Number of failed molecules	Total execution time (hh:mm:ss)[a]	Number of failed molecules
Balloon DG	13:26:45	0	64:55:43	1
Balloon GA	10:18:53	3	53:12:55	3
CONFORGE Systematic Best	00:08:55	0	00:20:23	0

CONFORGE Systematic Default	00:05:43	0	00:13:14	0
Conformator Best	03:11:41	4	05:13:06	4
Conformator Fast	02:50:25	4	03:21:48	4
iCon Best	00:31:59	0	00:37:46	0
iCon Fast	00:26:07	0	00:33:44	0
RDKit ETKDGv3	04:36:43	1	22:48:01	1
RDKit KDG	03:34:20	1	18:03:37	1

[a]Wall time elapsed between program start and termination.

As can be seen in Table 4, the largest improvement of CONFORGE regarding accuracy can be identified in the number of molecules for which conformer generation produces especially similar results to the true bioactive conformation (RMSD < 0.5 Å). CONFORGE Systematic Best is thereby able to produce more conformers close to this ground truth, while still performing similar to the competitors for larger RMSD accuracy ranges.

Table 4. Percentile of Platinum Diverse Dataset Structures successfully reproduced below specified RMSD thresholds

Generator	**Maximum ensemble size 50**				**Maximum ensemble size 250**			
	RMSD threshold (Å)							
	0.5	**1.0**	**1.5**	**2.0**	**0.5**	**1.0**	**1.5**	**2.0**

Balloon DG	32.4	61.1	78.5	89.6	41.8	69.7	84.5	93.6
Balloon GA	26.6	58.2	79.2	91.1	34.2	67.6	86.6	94.7
CONFORGE Systematic Best	**51.4**	78.8	90.3	96.4	**59.5**	86.4	95.0	98.3
CONFORGE Systematic Default	44.5	80.0	92.2	97.2	47.1	85.2	95.6	99.0
Conformator Best	41.8	78.0	**93.2**	**98.4**	52.9	85.9	**96.5**	99.0
Conformator Fast	36.8	73.8	91.8	98.0	45.9	83.2	95.5	98.8
iCon Best	36.5	**80.8**	93.1	97.9	38.0	**86.6**	96.2	99.0
iCon Fast	46.1	76.5	89.5	96.5	54.3	85.3	95.1	98.4
RDKit ETKDGv3	40.9	75.7	92.0	97.8	47.6	84.9	96.5	**99.4**
RDKit KDG	40.7	75.0	90.3	96.6	48.5	83.3	95.3	98.5
Conformator Best[a]	*42*	*78*	*94*	*98*	*53*	*86*	*97*	*99*
Conformator Fast[a]	*37*	*73*	*91*	*98*	*46*	*83*	*95*	*99*
Omega[a]	*49*	*80*	*92*	*97*	*56*	*87*	*96*	*99*

The best value for each RMSD threshold obtained by any assessed generator is written in bold letters. [a]The values listed for Conformator were taken from ref. [11] and included for reference to demonstrate the correctness of our benchmarking code by the close reproduction of the previously published results for this generator. As a side effect, this allows us to also include the

in ref. [11] published results for Omega which we could not assess in our study due to licensing reasons.

In practice, the ensemble size produced during conformer generation is also an important factor. If a large number of conformers is produced, it is more likely that one of them is close to the bioactive conformation, as a fundamental goal of conformer generation is to produce a sufficiently diverse set of results. Still, this increase in the number of conformations comes with the downside of having to search through this larger solution space for the conformer(s) desirable for a given use-case and results in a larger uncertainty of whether or not the correct one was chosen. For this reason, we visualized in Figure 14 how the accuracy, ensemble size and processing times compare between the different conformer generation methods. We limited the ensemble sizes to 50 and 250 for the different measurements, and it can be seen that CONFORGE lies approximately in the middle regarding how much of the ensemble size limit it uses, with the 'Best' variant producing smaller ensembles than the 'Default' variant. Generally, all methods would be allowed to output maximally large ensemble sizes (exactly the maximum allowed number for each molecule), but, with the exception of RDKit, they all discard too similar solutions. Even though CONFORGE produces similarly sized ensembles, its runtime is unparalleled by the other methods and the accuracy is nearly always better. This shows that the aforementioned benefits in accuracy and especially runtime are not an artifact of larger ensembles.

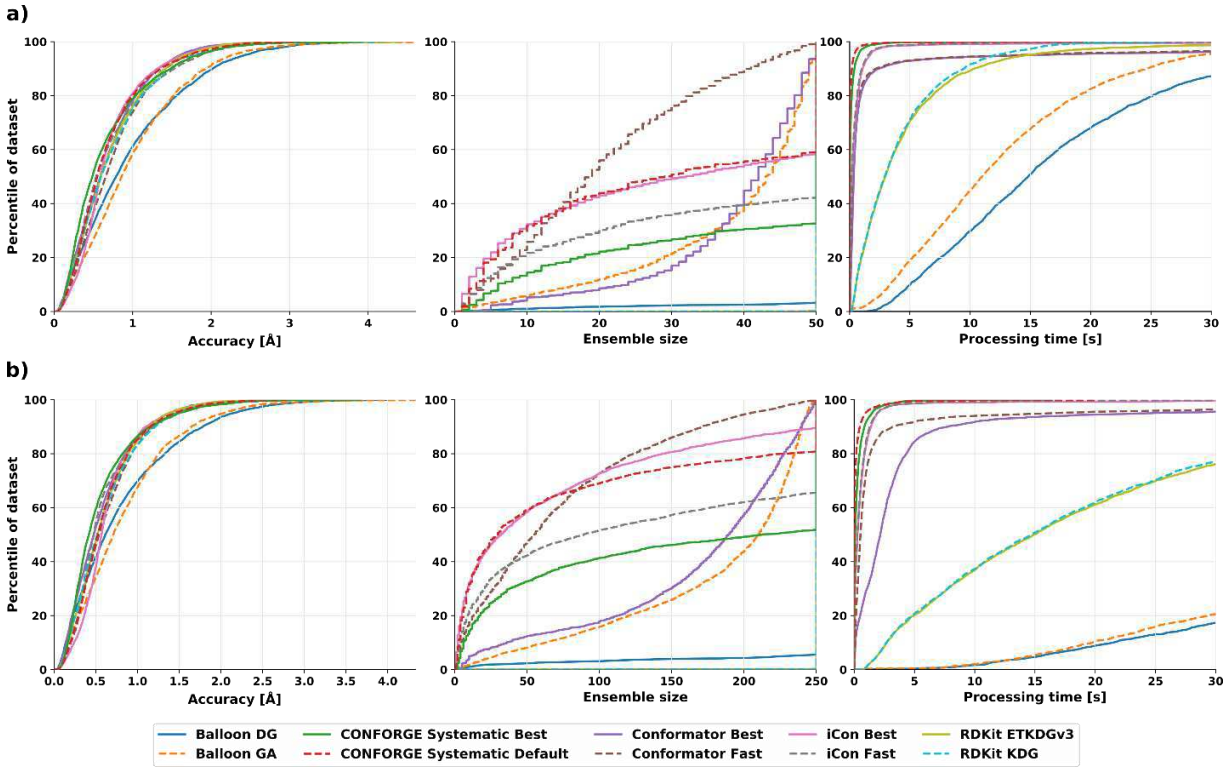


Figure 14. Percentile of Platinum Diverse Dataset molecules as a function of accuracy in the reproduction of the protein-bound ligand conformation (left), output ensemble size (middle), and molecule processing time (right) at maximum ensemble sizes of (a) 50 and (b) 250 conformers. For each of the diagrams, quicker rising curves indicate better performance.

The size of ensembles usually relates to the degrees of freedom that can be sampled during conformer generation. Rotatable bonds are thereby the largest contributing factor as the 3D position of molecule substructures connected to each of these bonds can vary, leading to a combinatorial problem when assembling the final conformer. Figure 15 shows this property nicely, as most conformer generators produce larger ensembles with an increasing number of rotatable bonds. Exceptions to this are the variants of Balloon, which seem to be agnostic to this number, and Conformator's 'Best' variant, which does not use the full amount of allowed conformers, even for complex molecules.

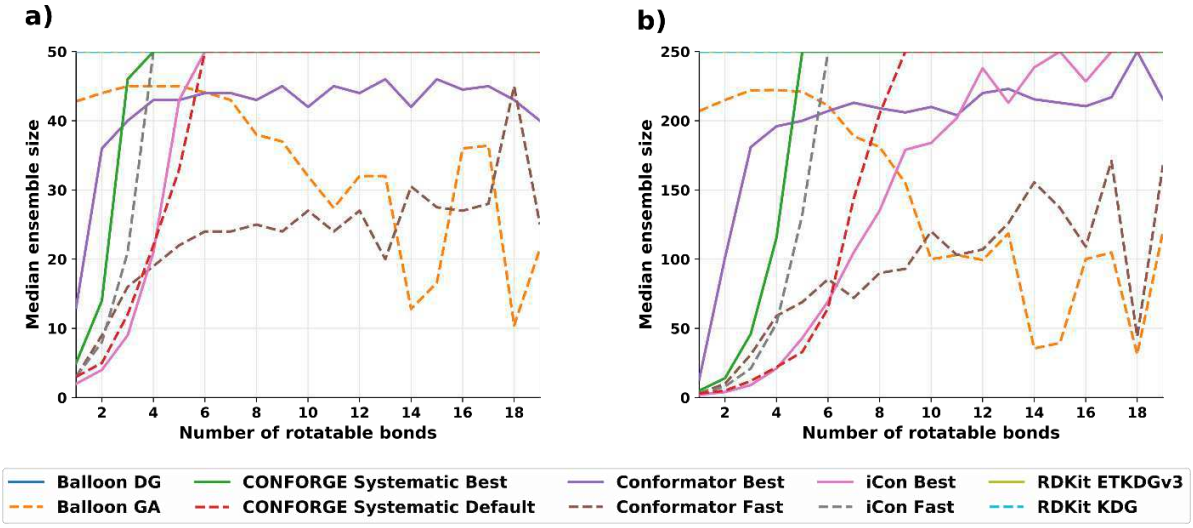


Figure 15. Median ensemble size as a function of the number of rotatable bonds at maximum ensembles of (a) 50 and (b) 250 conformers generated for the Platinum Diverse Dataset. Curves with lower median ensemble sizes indicate better performance with respect to this criterion.

The molecules are also likely to contain more atoms with an increasing number of rotatable bonds. This introduces more possibilities of errors in the relative position of the atoms compared to the crystal structure, leading to a higher expected RMSD than for small molecules when using the same conformer generation method and, therefore, lower accuracy. With increasing molecule size, this inevitably leads to a decreasing number of molecules retrieved with any fixed RMSD precision. Figure 16 visualizes this relationship for the different benchmarked methods and shows how the different algorithms are affected by this phenomenon. While all methods retrieve less molecules within a fixed threshold with an increasing number of rotatable bonds, our CONFORGE variants are usually able to handle this increase better than the competitors. Not only is the retrieval of molecules with few rotatable bonds more accurate, larger ones with more rotatable bonds also experience less of an accuracy loss compared to the other benchmarked methods. Other methods we could identify to handle larger numbers of rotatable bonds well include 'iCon Best' (1 Å), 'Conformator Best' (1.5-2 Å) and 'RDKit ETKDGv3' (2.5 Å) which, as can be seen in Table 4, show good reproduction rates with increasing RMSD thresholds. For the percentage of molecules retrieved within the accuracy threshold of RMSD <= 0.5 Å 'CONFORGE Systematic Best' outperformed all other methods.

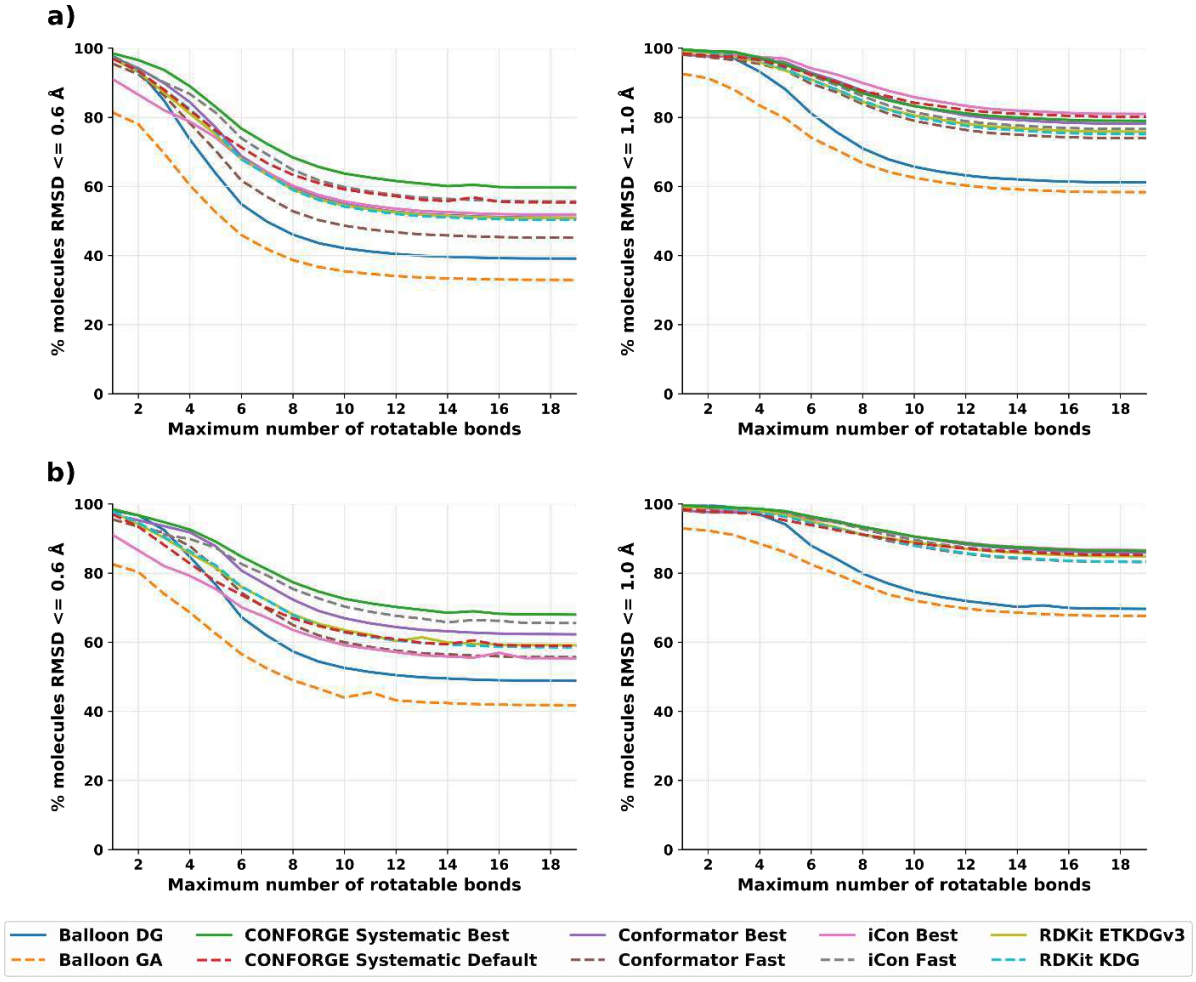


Figure 16. Percentile of Platinum Diverse Dataset molecules reproduced by the assessed generators with a maximum RMSD of 0.6 (left) and 1.0 Å (right) as a function of the maximum number of rotatable bonds at maximum ensemble sizes of (a) 50 and (b) 250 conformers.

Macrocycle Conformer Sampling Performance.

Macrocyclic compounds usually pose an even larger challenge for conformer generators due to the large number of (partially) rotatable bonds contributed by flexible side chains and the macrocyclic ring system itself. For this reason, we separately compare the performance of the different conformer generation methods to our CONFORGE Stochastic variant, which has been specifically designed for macrocycles and large/complex compounds.

Table 5 shows the performed measurements. CONFORGE Stochastic achieves on average higher accuracy than any other herein benchmarked method, with high statistical significance ($\alpha = 0.05$, supplementary Table S6). When only measuring the accuracy for atoms constituting the macrocyclic ring, it can be observed that the relative improvement over other methods is even larger. Still, this improved conformer quality comes at the cost of longer runtimes compared to other methods. As the quality of the results is significantly better and the relevance of macrocyclic compounds for drug development is continuously rising, our new method will provide drastic improvements when working with such molecules. The only notable competitor in this domain is the RDKit variant ETKDGv3. This method is able to outperform CONFORGE Stochastic regarding runtime while producing conformers that closely resemble experimental 3D structures. Still, the conformers produced by RDKit are more difficult to use in practice as the method always generates conformers up to the maximum allowed ensemble size. This larger ensemble size, compared to our method's 255.66 on average, may also be part of the reason why this competitor produces more sensible results than the other tested methods.

Table 5. Conformer Generator Performance Comparison for the Prime Macrocycle Dataset

Generator	Mean	Median	Min.	Max.
Overall RMSD (Å)[a]				
Balloon DG	1.782	1.434	0.043	7.637
Balloon GA	1.851	1.592	**0.032**	5.829
CONFORGE Stochastic	**1.153**	**0.942**	0.042	5.249
Conformator Best	1.563	1.187	0.180	6.478
Conformator Fast	1.615	1.252	0.221	6.468
iCon Best	1.472	1.115	0.052	6.372

iCon Fast	1.479	1.173	0.052	5.928
RDKit ETKDGv3	1.247	0.978	0.052	4.842
RDKit KDG	1.439	1.305	0.049	**4.543**
Macrocycle RMSD (Å)[b]				
Balloon DG	0.938	0.736	0.029	3.855
Balloon GA	1.140	0.917	**0.025**	4.528
CONFORGE Stochastic	**0.586**	**0.473**	0.028	3.155
Conformator Best	0.974	0.808	0.126	5.193
Conformator Fast	1.026	0.857	0.126	3.841
iCon Best	0.913	0.714	0.035	4.753
iCon Fast	0.950	0.762	0.035	5.240
RDKit ETKDGv3	0.672	0.584	0.027	**3.141**
RDKit KDG	0.833	0.747	0.027	3.379
Ensemble Size				
Balloon DG	321.20	500	-	-
Balloon GA	145.28	64	-	-
CONFORGE Stochastic	255.66	242	-	-

Conformator Best	236.02	233	-	-
Conformator Fast	99.53	59	-	-
iCon Best	**69.22**	**43.8**	-	-
iCon Fast	130.72	69	-	-
RDKit ETKDGv3	500.00	500	-	-
RDKit KDG	500.00	500	-	-
Processing Time (s)				
Balloon DG	262.62	174.45	9.87	1453.09
Balloon GA	494.88	343.06	40.58	2602.07
CONFORGE Stochastic	977.88	432.91	12.10	10609.27
Conformator Best	471.40	169.61	7.19	43358.54
Conformator Fast	186.90	111.53	1.54	5615.60
iCon Best	28.53	30.17	0.72	**492.15**
iCon Fast	**26.05**	**24.91**	**0.68**	574.71
RDKit ETKDGv3	677.82	159.89	7.25	34714.72
RDKit KDG	312.48	148.09	7.37	2962.38

The best values for RMSD, ensemble size, and molecule processing time obtained by any assessed generator are written in bold letters. [a]RMSD taking into account all heavy atoms of the

molecule. [b]RMSD taking into account only the heavy atoms constituting the macrocyclic ring system.

The difficulty of this task can be seen once again in Table 6 where the number of molecules for which the conformer generation failed is shown. Besides CONFORGE Stochastic, only RDKit was able to complete the computations without failure, while the other methods could not produce any results for at least one molecule. Notably, Conformator even completely crashed with a segmentation fault when presented 6 of the molecules in the dataset (VEVHAF, POTTEY, PRD_000785, 1YND_SFA, 1NMK_SFM, 3I6O_GR6).

Table 6. Total Program Execution Times and Molecule Processing Failures recorded for the Prime Macrocycle Dataset

Generator	Total execution time (hh:mm:ss)[a]	Number of failed molecules
Balloon DG	14:31:35	9
Balloon GA	27:20:21	11
CONFORGE Stochastic	56:30:54	0
Conformator Best	26:29:35	7
Conformator Fast	10:46:42	9
iCon Best	01:48:45	1
iCon Fast	01:40:27	1
RDKit ETKDGv3	39:09:47	0
RDKit KDG	18:03:17	0

[a]Wall time elapsed between program start and termination.

Just as for the previous benchmark, we computed the percentiles of molecules for which conformers within a certain RMSD threshold have been found by the individual methods. Table 7 shows the resulting values, with CONFORGE Stochastic finding more high-quality conformers for any of the presented thresholds. The next best percentile values were measured for RDKit ETKDGv3.

Table 7. Percentile of Prime Macrocycle Dataset Structures successfully reproduced below specified RMSD thresholds

Generator	RMSD threshold (Å)							
	0.5	**1.0**	**1.5**	**2.0**	**2.5**	**3.0**	**3.5**	**4.0**
Balloon DG	16.3	35.6	49.0	61.1	70.7	77.9	82.7	87.5
Balloon GA	5.8	23.6	43.8	59.6	71.6	79.3	85.6	90.4
CONFORGE Stochastic	**25.5**	**52.4**	**75.0**	**81.7**	**92.3**	**95.7**	**98.6**	**98.6**
Conformator Best	11.5	39.9	60.6	70.7	80.3	86.5	89.4	92.3
Conformator Fast	7.2	36.1	55.3	69.2	78.4	84.6	88.9	90.9
iCon Best	19.2	45.7	60.6	72.6	80.8	90.4	93.7	95.7
iCon Fast	25.0	44.7	60.6	70.7	80.8	88.0	93.8	95.7

RDKit ETKDGv3	22.6	51.4	67.8	79.8	89.4	94.7	97.6	**98.6**
RDKit KDG	12.5	38.9	61.5	74.0	87.0	94.2	97.6	**98.6**

The best value for each RMSD threshold obtained by any of the assessed generators is written in bold letters.

The accuracy benefits of CONFORGE can, once again, be seen in Figure 17. The produced ensembles are of average size compared to our competitors, while the method needs more time to produce the results than others. This increase in computation time should be more than made up for by the increase in quality over the other algorithms.

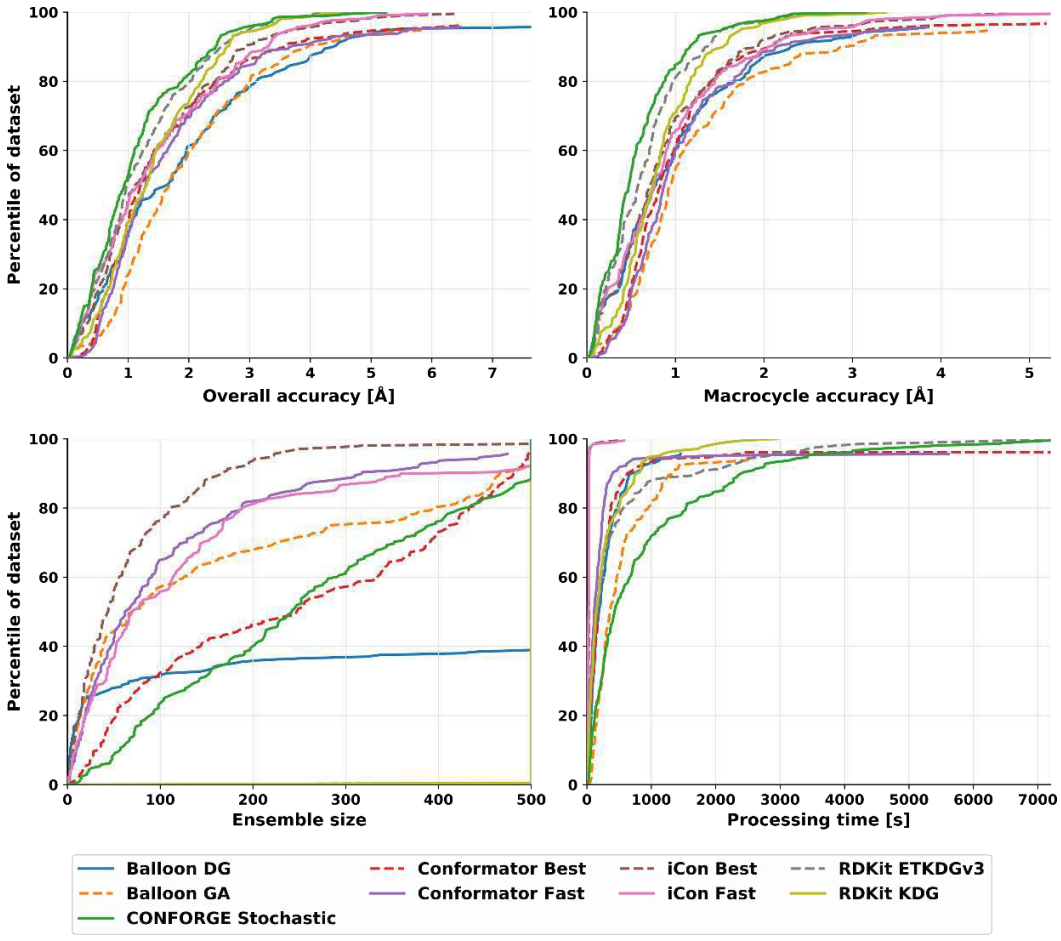


Figure 17. Percentile of Prime Macrocycle Dataset molecules as a function of accuracy in the reproduction of the crystallographic 3D-structure (left), output ensemble size (middle), and molecule processing time (right). For each of the diagrams, quicker rising curves indicate better performance.

The quality of CONFORGE Stochastic's results is visualized cumulatively over the number of rotatable bonds and size of the macrocycle in Figure 18. There, the plots show that for nearly all possible thresholds for those two properties, our method drastically outperforms any other algorithm regarding the accuracy of generated conformers.

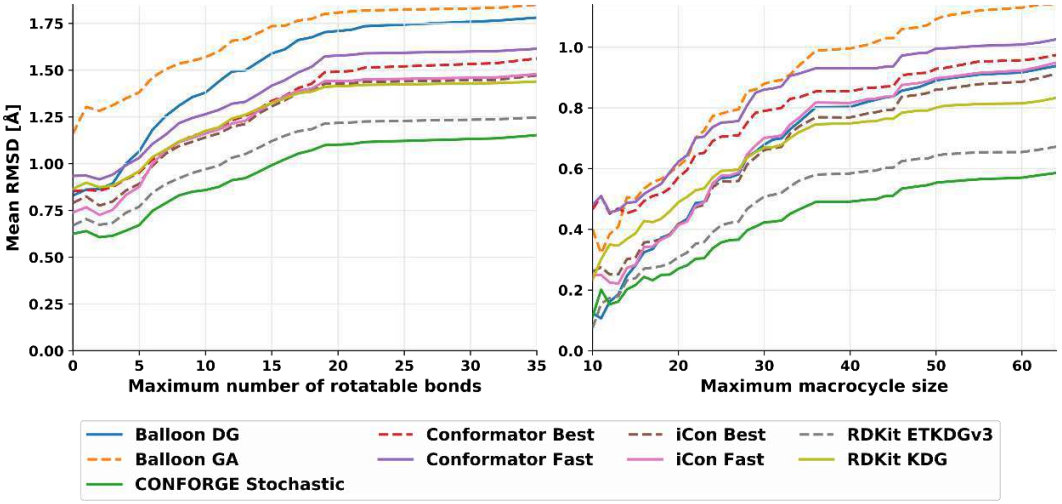


Figure 18. Mean overall RMSD as a function of the maximum number of rotatable side chain bonds (left) and maximum size (atom count) of the macrocyclic ring system (right) for the Prime Macrocycle Dataset. Curves with lower mean RMSD indicate better performance with respect to this criterion.

Conclusions

We introduced and described the implementation of the novel conformer ensemble generator CONFORGE which has been designed and developed to deliver top-level performance for all types of organic molecules in the drug-like chemical space. CONFORGE is fully open-source and available as part of the Chemical Data Processing Toolkit in the form of a versatile command-line tool that accepts a wide panel of input data formats and a set of classes and

functions provided by CDPKit's C++/Python-API. CONFORGE's implementation is based on established concepts and algorithms which have proven their power with respect to conformer ensemble generation in other well-known software tools for this purpose. For the computationally efficient and accurate conformational sampling of drug-like small molecules a knowledge-based systematic approach is employed, which makes extensive use of pre-generated fragment and torsion angle libraries that were derived from experimental 3D structures. For the sampling of macrocycle conformers, CONFORGE implements a purely stochastic approach based on a combination of DG and 3D structure refinement by iterative MMFF94 energy minimization. Irrespective of the sampling approach, CONFORGE does not require any input atom 3D coordinates and is able to generate conformer ensembles solely from molecular graph connection table information. The conformer sampling approach best suited for a processed input molecule is either chosen automatically or can be specified by the user in advance for all molecules to process. Furthermore, CONFORGE correctly handles compounds consisting of multiple molecules like salts and mixtures by a separate generation and later combination of individual component conformer ensembles.

CONFORGE's capability to reproduce experimental 3D structures as well as its computational efficiency and robustness has been assessed for typical drug-like organic molecules using the Platinum Diverse Dataset and for macrocyclic systems using a dataset of 208 molecules compiled by Sindhikara et al.[51] The calculation of performance metrics and the visual presentation of the obtained results largely followed the established protocol developed by Friedrich et al.[27] with extensions for the presentation of macrocycle sampling results. For comparison, several well-known commercial (iCon; two modes), non-open-source (Conformator; two modes) and open-source conformer generators (Balloon, RDKit; two modes/parameterizations) were benchmarked in addition to CONFORGE using the same benchmarking protocol. For the Platinum Diverse Dataset benchmarks, two runs with maximum output ensemble sizes (MES) of 50 and 250 representative conformers were performed, and for the testing of the macrocycle sampling performance, a MES of 500 was chosen. In the Platinum Diverse Dataset benchmarks, CONFORGE achieved a median accuracy in the reproduction of bioactive conformations of 0.486 Å (MES = 50) and 0.416 Å (MES = 250), respectively, which were the best values among all tested generators (see Table 2). At the same time, CONFORGE had the lowest mean processing time per molecule (12 ms for MES = 50 and 14 ms for MES =

250, respectively) and the lowest mean output ensemble size of 29 conformers in case of a MES of 250 and the second lowest for a MES of 50, respectively. The mean accuracy of 1.153 Å achieved by CONFORGE for the macrocycle dataset was again the best among all benchmarked conformer ensemble generators (see Table 5). However, CONFORGE also showed the highest mean per molecule processing time (432.91 s) and produced output ensembles with a mean size of 242 conformers (see Table 5). These results indicate that there is still room for improvements when it comes to macrocycle conformer sampling and CONFORGE will be best suited for low throughput applications which favor accuracy over speed.

In summary, the presented open-source conformer ensemble generator CONFORGE has proven to be able to deliver excellent performance in several aspects of relevance. To our knowledge, CONFORGE is the first open-source conformer generator that is able to compete with leading commercial software for the conformational sampling of drug-like small molecules and at the same time also shows very good performance when it comes to the conformational sampling of macrocyclic systems. It will provide the scientific community with a truly free alternative of high quality that facilitates open research due to the absence of any restrictions on the use of the generated data. As part of the CDPKit project, CONFORGE will be actively maintained and undergo further development. Planned future improvements of CONFORGE include speed optimizations regarding stochastic sampling, the possibility to use other general-purpose force fields like the Open Force Field[80] and functionality enabling a restriction of conformer sampling to only particular user-specified parts of the processed molecule.

Data and Software Availability

The source code of the developed benchmarking suite, both test datasets in SDF and SMILES format, the generated conformer ensembles (multi-conf. SD-files) and benchmarking result files (CSV-files and figures) for all assessed generators, the CDPKit source code[52] in the version at the time of benchmarking and an installer for the corresponding CDPKit binaries (compiled for RHEL 8.x based systems) can be downloaded from https://zenodo.org/record/7817208 (DOI: 10.5281/zenodo.7817208).

Supplementary Information

A description of all options supported by CONFORGE's command-line interface (Table S1) and tables/text summarizing the results of the carried out Paired Wilcoxon Sign Rank Tests (Tables S2, S3, S4, S5 and S6; PDF).

Author Contributions

TS implemented CONFORGE and parts of the benchmarking code, is the main developer of CDPKit, carried out and supervised the conformer generator benchmarks, analyzed the results and wrote the manuscript. CP is the main developer of the benchmarking code, analyzed benchmarking results and contributed to writing of the manuscript. SK, OW and TL contributed to the writing and proof reading of the manuscript. All authors have given approval to the final version of the manuscript.

Competing Interests

The authors declare no competing financial interest.

Funding Sources

The authors gratefully acknowledge funding by the University of Vienna Research Platform NeGeMac (Next Generation Macrocycles to Address Challenging Protein Interfaces).

Acknowledgments

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

References

(1) Perola, E.; Charifson, P. S. Conformational Analysis of Drug-like Molecules Bound to Proteins: An Extensive Study of Ligand Reorganization upon Binding. *J. Med. Chem.* **2004**,

47 (10), 2499–2510. https://doi.org/10.1021/jm030563w.

(2) Lyne, P. D. Structure-Based Virtual Screening: An Overview. *Drug Discov. Today* **2002**, *7* (20), 1047–1055. https://doi.org/10.1016/s1359-6446(02)02483-2.

(3) Venkatraman, V.; Pérez-Nueno, V. I.; Mavridis, L.; Ritchie, D. W. Comprehensive Comparison of Ligand-Based Virtual Screening Tools against the DUD Data Set Reveals Limitations of Current 3D Methods. *J. Chem. Inf. Model.* **2010**, *50* (12), 2079–2093. https://doi.org/10.1021/ci100263p.

(4) Hu, G.; Kuang, G.; Xiao, W.; Li, W.; Liu, G.; Tang, Y. Performance Evaluation of 2D Fingerprint and 3D Shape Similarity Methods in Virtual Screening. *J. Chem. Inf. Model.* **2012**, *52* (5), 1103–1113. https://doi.org/10.1021/ci300030u.

(5) Langer, T.; Wolber, G. Pharmacophore Definition and 3D Searches. *Drug Discov. Today Technol.* **2004**, *1* (3), 203–207. https://doi.org/10.1016/j.ddtec.2004.11.015.

(6) Strategies for 3D Pharmacophore-Based Virtual Screening. *Drug Discov. Today Technol.* **2010**, *7* (4), e203–e270. https://doi.org/10.1016/j.ddtec.2010.11.004.

(7) Poli, G.; Seidel, T.; Langer, T. Conformational Sampling of Small Molecules With iCon: Performance Assessment in Comparison With OMEGA. *Front Chem* **2018**, *6*, 229. https://doi.org/10.3389/fchem.2018.00229.

(8) Hawkins, P. C. D.; Skillman, A. G.; Warren, G. L.; Ellingson, B. A.; Stahl, M. T. Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database. *J. Chem. Inf. Model.* **2010**, *50* (4), 572–584. https://doi.org/10.1021/ci100031x.

(9) Watts, K. S.; Dalal, P.; Murphy, R. B.; Sherman, W.; Friesner, R. A.; Shelley, J. C. ConfGen: A Conformational Search Method for Efficient Generation of Bioactive Conformers. *J. Chem. Inf. Model.* **2010**, *50* (4), 534–546. https://doi.org/10.1021/ci100015j.

(10) Li, J.; Ehlers, T.; Sutter, J.; Varma-O'brien, S.; Kirchmair, J. CAESAR: A New Conformer Generation Algorithm Based on Recursive Buildup and Local Rotational Symmetry Consideration. *J. Chem. Inf. Model.* **2007**, *47* (5), 1923–1932. https://doi.org/10.1021/ci700136x.

(11) Friedrich, N.-O.; Flachsenberg, F.; Meyder, A.; Sommer, K.; Kirchmair, J.; Rarey, M. Conformator: A Novel Method for the Generation of Conformer Ensembles. *J. Chem. Inf. Model.* **2019**, *59* (2), 731–742. https://doi.org/10.1021/acs.jcim.8b00704.

(12) Sadowski, P.; Baldi, P. Small-Molecule 3D Structure Prediction Using Open Crystallography Data. *J. Chem. Inf. Model.* **2013**, *53* (12), 3127–3130. https://doi.org/10.1021/ci4005282.

(13) Labute, P. LowModeMD—Implicit Low-Mode Velocity Filtering Applied to Conformational Search of Macrocycles and Protein Loops. *Journal of Chemical Information and Modeling*. 2010, pp 792–800. https://doi.org/10.1021/ci900508k.

(14) *Molecular operating environment (MOE)*. https://www.chemcomp.com/Products.htm (accessed 2023-03-30).

(15) Cleves, A. E.; Jain, A. N. ForceGen 3D Structure and Conformer Generation: From Small Lead-like Molecules to Macrocyclic Drugs. *J. Comput. Aided Mol. Des.* **2017**, *31* (5), 419–439. https://doi.org/10.1007/s10822-017-0015-8.

(16) Jain, A. N.; Cleves, A. E.; Gao, Q.; Wang, X.; Liu, Y.; Sherer, E. C.; Reibarkh, M. Y. Complex Macrocycle Exploration: Parallel, Heuristic, and Constraint-Based Conformer Generation Using ForceGen. *J. Comput. Aided Mol. Des.* **2019**, *33* (6), 531–558. https://doi.org/10.1007/s10822-019-00203-1.

(17) Leite, T. B.; Gomes, D.; Miteva, M. A.; Chomilier, J.; Villoutreix, B. O.; Tufféry, P. Frog: A FRee Online druG 3D Conformation Generator. *Nucleic Acids Res.* **2007**, *35* (Web Server issue), W568–W572. https://doi.org/10.1093/nar/gkm289.

(18) Miteva, M. A.; Guyon, F.; Tufféry, P. Frog2: Efficient 3D Conformation Ensemble Generator for Small Compounds. *Nucleic Acids Res.* **2010**, *38* (Web Server issue), W622–W627. https://doi.org/10.1093/nar/gkq325.

(19) O'Boyle, N. M.; Vandermeersch, T.; Flynn, C. J.; Maguire, A. R.; Hutchison, G. R. Confab - Systematic Generation of Diverse Low-Energy Conformers. *J. Cheminform.* **2011**, *3* (1), 8. https://doi.org/10.1186/1758-2946-3-8.

(20) Kothiwale, S.; Mendenhall, J. L.; Meiler, J. BCL::Conf: Small Molecule Conformational Sampling Using a Knowledge Based Rotamer Library. *J. Cheminform.* **2015**, *7*, 47. https://doi.org/10.1186/s13321-015-0095-1.

(21) Landrum, G. *RDKit*. http://www.rdkit.org (accessed 2023-03-24).

(22) Vainio, M. J.; Johnson, M. S. Generating Conformer Ensembles Using a Multiobjective Genetic Algorithm. *J. Chem. Inf. Model.* **2007**, *47* (6), 2462–2474. https://doi.org/10.1021/ci6005646.

(23) Sauton, N.; Lagorce, D.; Villoutreix, B. O.; Miteva, M. A. MS-DOCK: Accurate Multiple Conformation Generator and Rigid Docking Protocol for Multi-Step Virtual Ligand Screening. *BMC Bioinformatics* **2008**, *9*, 184. https://doi.org/10.1186/1471-2105-9-184.

(24) Blaney, J. M.; Dixon, J. S. Distance Geometry in Molecular Modeling. In *Reviews in Computational Chemistry*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007; pp 299–335. https://doi.org/10.1002/9780470125823.ch6.

(25) Crippen, G. M.; Havel, T. F. Distance Geometry and Molecular Conformation. **1988**.

(26) Wang, S.; Witek, J.; Landrum, G. A.; Riniker, S. Improving Conformer Generation for Small Rings and Macrocycles Based on Distance Geometry and Experimental Torsional-Angle Preferences. *J. Chem. Inf. Model.* **2020**, *60* (4), 2044–2058. https://doi.org/10.1021/acs.jcim.0c00025.

(27) Friedrich, N.-O.; Meyder, A.; de Bruyn Kops, C.; Sommer, K.; Flachsenberg, F.; Rarey, M.; Kirchmair, J. High-Quality Dataset of Protein-Bound Ligand Conformations and Its Application to Benchmarking Conformer Ensemble Generators. *J. Chem. Inf. Model.* **2017**, *57* (3), 529–539. https://doi.org/10.1021/acs.jcim.6b00613.

(28) Friedrich, N.-O.; de Bruyn Kops, C.; Flachsenberg, F.; Sommer, K.; Rarey, M.; Kirchmair, J. Benchmarking Commercial Conformer Ensemble Generators. *J. Chem. Inf. Model.* **2017**, *57* (11), 2719–2728. https://doi.org/10.1021/acs.jcim.7b00505.

(29) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The Protein Data Bank. *Nucleic Acids Res.* **2000**, *28* (1), 235–242. https://doi.org/10.1093/nar/28.1.235.

(30) Yudin, A. K. Macrocycles: Lessons from the Distant Past, Recent Developments, and Future Directions. *Chem. Sci.* **2015**, *6* (1), 30–49. https://doi.org/10.1039/c4sc03089c.

(31) Marsault, E.; Peterson, M. L. Macrocycles Are Great Cycles: Applications, Opportunities, and Challenges of Synthetic Macrocycles in Drug Discovery. *J. Med. Chem.* **2011**, *54* (7), 1961–2004. https://doi.org/10.1021/jm1012374.

(32) Mallinson, J.; Collins, I. Macrocycles in New Drug Discovery. *Future Med. Chem.* **2012**, *4* (11), 1409–1438. https://doi.org/10.4155/fmc.12.93.

(33) Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development settings1PII of Original Article: S0169-409X(96)00423-1. The Article Was

Originally Published in Advanced Drug Delivery Reviews 23 (1997) 3–25.1. *Adv. Drug Deliv. Rev.* **2001**, *46* (1), 3–26. https://doi.org/10.1016/S0169-409X(00)00129-0.

(34) Bell, I. M.; Gallicchio, S. N.; Abrams, M.; Beese, L. S.; Beshore, D. C.; Bhimnathwala, H.; Bogusky, M. J.; Buser, C. A.; Christopher Culberson, J.; Davide, J.; Ellis-Hutchings, M.; Fernandes, C.; Gibbs, J. B.; Graham, S. L.; Hamilton, K. A.; Hartman, G. D.; Heimbrook, D. C.; Homnick, C. F.; Huber, H. E.; Huff, J. R.; Kassahun, K.; Koblan, K. S.; Kohl, N. E.; Lobell, R. B.; Lynch, J. J.; Robinson, R.; David Rodrigues, A.; Taylor, J. S.; Walsh, E. S.; Williams, T. M.; Blair Zartman, C. 3-Aminopyrrolidinone Farnesyltransferase Inhibitors: Design of Macrocyclic Compounds with Improved Pharmacokinetics and Excellent Cell Potency. *Journal of Medicinal Chemistry*. 2002, pp 2388–2409. https://doi.org/10.1021/jm010531d.

(35) Dougherty, P. G.; Qian, Z.; Pei, D. Macrocycles as Protein-Protein Interaction Inhibitors. *Biochem. J* **2017**, *474* (7), 1109–1125. https://doi.org/10.1042/BCJ20160619.

(36) Giordanetto, F.; Kihlberg, J. Macrocyclic Drugs and Clinical Candidates: What Can Medicinal Chemists Learn from Their Properties? *J. Med. Chem.* **2014**, *57* (2), 278–295. https://doi.org/10.1021/jm400887j.

(37) Rezai, T.; Bock, J. E.; Zhou, M. V.; Kalyanaraman, C.; Scott Lokey, R.; Jacobson, M. P. Conformational Flexibility, Internal Hydrogen Bonding, and Passive Membrane Permeability: Successful in Silico Prediction of the Relative Permeabilities of Cyclic Peptides. *Journal of the American Chemical Society*. 2006, pp 14073–14080. https://doi.org/10.1021/ja063076p.

(38) Dömling, A. Small Molecular Weight Protein–protein Interaction Antagonists—an Insurmountable Challenge? *Current Opinion in Chemical Biology*. 2008, pp 281–291. https://doi.org/10.1016/j.cbpa.2008.04.603.

(39) Doak, B. C.; Over, B.; Giordanetto, F.; Kihlberg, J. Oral Druggable Space beyond the Rule of 5: Insights from Drugs and Clinical Candidates. *Chemistry & Biology*. 2014, pp 1115–1142. https://doi.org/10.1016/j.chembiol.2014.08.013.

(40) Kwitkowski, V. E.; Prowell, T. M.; Ibrahim, A.; Farrell, A. T.; Justice, R.; Mitchell, S. S.; Sridhara, R.; Pazdur, R. FDA Approval Summary: Temsirolimus as Treatment for Advanced Renal Cell Carcinoma. *Oncologist* **2010**, *15* (4), 428–435. https://doi.org/10.1634/theoncologist.2009-0178.

(41) Raymond, E.; Alexandre, J.; Faivre, S.; Vera, K.; Materman, E.; Boni, J.; Leister, C.; Korth-Bradley, J.; Hanauske, A.; Armand, J.-P. Safety and Pharmacokinetics of Escalated Doses of Weekly Intravenous Infusion of CCI-779, a Novel mTOR Inhibitor, in Patients with Cancer. *J. Clin. Oncol.* **2004**, *22* (12), 2336–2347.

(42) Goodin, S. Novel Cytotoxic Agents: Epothilones. *Am. J. Health. Syst. Pharm.* **2008**, *65* (10 Suppl 3), S10–S15. https://doi.org/10.2146/ajhp080089.

(43) Goodin, S. Ixabepilone: A Novel Microtubule-Stabilizing Agent for the Treatment of Metastatic Breast Cancer. *Am. J. Health. Syst. Pharm.* **2008**, *65* (21), 2017–2026. https://doi.org/10.2146/ajhp070628.

(44) Marsault, E.; Peterson, M. L. *Practical Medicinal Chemistry with Macrocycles: Design, Synthesis, and Case Studies*; John Wiley & Sons, 2017.

(45) Hawkins, P. C. D. Conformation Generation: The State of the Art. *J. Chem. Inf. Model.* **2017**, *57* (8), 1747–1756. https://doi.org/10.1021/acs.jcim.7b00221.

(46) Reyes Romero, A.; Ruiz-Moreno, A. J.; Groves, M. R.; Velasco-Velázquez, M.; Dömling, A. Benchmark of Generic Shapes for Macrocycles. *J. Chem. Inf. Model.* **2020**, *60* (12), 6298–6313. https://doi.org/10.1021/acs.jcim.0c01038.

(47) Olanders, G.; Alogheli, H.; Brandt, P.; Karlén, A. Conformational Analysis of Macrocycles: Comparing General and Specialized Methods. *Journal of Computer-Aided Molecular Design*. 2020, pp 231–252. https://doi.org/10.1007/s10822-020-00277-2.

(48) Poongavanam, V.; Danelius, E.; Peintner, S.; Alcaraz, L.; Caron, G.; Cummings, M. D.; Wlodek, S.; Erdelyi, M.; Hawkins, P. C. D.; Ermondi, G.; Kihlberg, J. Conformational Sampling of Macrocyclic Drugs in Different Environments: Can We Find the Relevant Conformations? *ACS Omega* **2018**, *3* (9), 11742–11757. https://doi.org/10.1021/acsomega.8b01379.

(49) *Macrocycle conformations — applications, vDev build*. https://docs.eyesopen.com/applications/omega/theory/macrocycle_theory.html (accessed 2023-03-30).

(50) Watts, K. S.; Shawn Watts, K.; Dalal, P.; Tebben, A. J.; Cheney, D. L.; Shelley, J. C. Macrocycle Conformational Sampling with MacroModel. *Journal of Chemical Information and Modeling*. 2014, pp 2680–2696. https://doi.org/10.1021/ci5001696.

(51) Sindhikara, D.; Spronk, S. A.; Day, T.; Borrelli, K.; Cheney, D. L.; Posy, S. L. Improving

Accuracy, Diversity, and Speed with Prime Macrocycle Conformational Sampling. *J. Chem. Inf. Model.* **2017**, *57* (8), 1881–1894. https://doi.org/10.1021/acs.jcim.7b00052.

(52) Seidel, T. *CDPKit: The Chemical Data Processing Toolkit*. https://github.com/molinfo-vienna/CDPKit (accessed 2023-03-29).

(53) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5* (2), 107–113. https://doi.org/10.1021/c160017a018.

(54) McKay, B. Practical Graph Isomorphism. In *Numerical mathematics and computing*; 1981.

(55) Halgren, T. A. Merck Molecular Force Field. II. MMFF94 van Der Waals and Electrostatic Parameters for Intermolecular Interactions. *J. Comput. Chem.* **1996**, *17* (5-6), 520–552. https://doi.org/10.1002/(sici)1096-987x(199604)17:5/6<520::aid-jcc2>3.0.co;2-w.

(56) Floyd, R. W. Algorithm 97: Shortest Path. *Commun. ACM* **1962**, *5* (6), 345. https://doi.org/10.1145/367766.368168.

(57) Kearsley, S. *MMFF94 Validation Suite*. http://www.ccl.net/cca/data/MMFF94/ver.98.05.22/index.html (accessed 2023-03-30).

(58) Agrafiotis, D. K. Stochastic Proximity Embedding. *J. Comput. Chem.* **2003**, *24* (10), 1215–1221. https://doi.org/10.1002/jcc.10234.

(59) Broyden, C. G. The Convergence of a Class of Double-Rank Minimization Algorithms 1. General Considerations. *IMA J Appl Math* **1970**, *6* (1), 76–90. https://doi.org/10.1093/imamat/6.1.76.

(60) Fletcher, R. A New Approach to Variable Metric Algorithms. *Comput. J.* **1970**, *13* (3), 317–322. https://doi.org/10.1093/comjnl/13.3.317.

(61) Goldfarb, D. A Family of Variable-Metric Methods Derived by Variational Means. *Math. Comput.* **1970**, *24* (109), 23–26. https://doi.org/10.1090/s0025-5718-1970-0258249-6.

(62) Shanno, D. F. Conditioning of Quasi-Newton Methods for Function Minimization. *Math. Comput.* **1970**, *24* (111), 647–656. https://doi.org/10.1090/s0025-5718-1970-0274029-x.

(63) Galassi, M.; Davies, J.; Theiler, J.; Gough, B.; Jungman, G.; Alken, P.; Booth, M.; Rossi, F.; Ulerich, R. *The gnu scientific library reference manual, 2007*. https://www.gnu.org/software/gsl/ (accessed 2023-03-30).

(64) Eastlake, D.; Jones, P. RFC3174: US Secure Hash Algorithm 1 (SHA1), Internet Engineering Task Force. 2001.

(65) James, C. A.; Weininger, D.; Delany, J. SMARTS Theory. Daylight Theory Manual; Daylight Chemical Information Systems. *Laguna Niguel: CA* **2000**.

(66) Schärfer, C.; Schulz-Gasch, T.; Ehrlich, H.-C.; Guba, W.; Rarey, M.; Stahl, M. Torsion Angle Preferences in Druglike Chemical Space: A Comprehensive Guide. *J. Med. Chem.* **2013**, *56* (5), 2016–2028. https://doi.org/10.1021/jm3016816.

(67) Guba, W.; Meyder, A.; Rarey, M.; Hert, J. Torsion Library Reloaded: A New Version of Expert-Derived SMARTS Rules for Assessing Conformations of Small Molecules. *J. Chem. Inf. Model.* **2016**, *56* (1), 1–5. https://doi.org/10.1021/acs.jcim.5b00522.

(68) Westbrook, J. D.; Shao, C.; Feng, Z.; Zhuravleva, M.; Velankar, S.; Young, J. The Chemical Component Dictionary: Complete Descriptions of Constituent Molecules in Experimentally Determined 3D Macromolecules in the Protein Data Bank. *Bioinformatics* **2015**, *31* (8), 1274–1278. https://doi.org/10.1093/bioinformatics/btu789.

(69) Kabsch, W. A Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallogr. A* **1976**, *32* (5), 922–923. https://doi.org/10.1107/S0567739476001873.

(70) Kabsch, W. A Discussion of the Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallogr. A* **1978**, *34* (5), 827–828. https://doi.org/10.1107/S0567739478001680.

(71) Groom, C. R.; Bruno, I. J.; Lightfoot, M. P.; Ward, S. C. The Cambridge Structural Database. *Acta Crystallogr B Struct Sci Cryst Eng Mater* **2016**, *72* (Pt 2), 171–179. https://doi.org/10.1107/S2052520616003954.

(72) The Biologically Interesting Molecule Reference Dictionary (BIRD). *RCSB Protein Data Bank* **2019**. https://doi.org/10.2210/wwpdb/doc_bird.

(73) Blair, R. C.; Higgins, J. J. Comparison of the Power of the Paired Samples T Test to that of Wilcoxon's Signed-Ranks Test under Various Population Shapes. *Psychol. Bull.* **1985**, *97* (1), 119.

(74) Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scand. Stat. Theory Appl.* **1979**, *6* (2), 65–70.

(75) OpenEye Scientific Software. *Academic license agreement preface*. https://www.eyesopen.com/academic-license-preface (accessed 2021-10-11).

(76) *rdkit.Chem.rdDistGeom module — The RDKit 2021.03.1 documentation*. https://www.rdkit.org/docs/source/rdkit.Chem.rdDistGeom.html (accessed 2023-03-30).

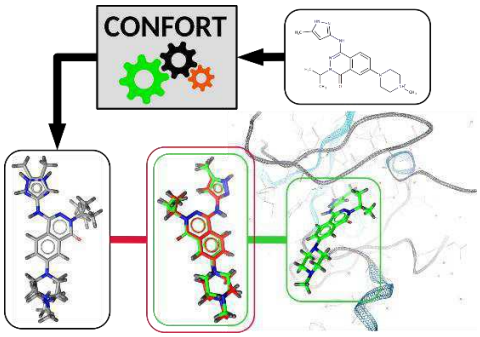
(77) Halgren, T. A. MMFF VI. MMFF94s Option for Energy Minimization Studies. *J. Comput. Chem.* **1999**, *20* (7), 720–729. https://doi.org/10.1002/(sici)1096-987x(199905)20:7<720::aid-jcc7>3.0.co;2-x.

(78) Wahl, J.; Freyss, J.; von Korff, M.; Sander, T. Accuracy Evaluation and Addition of Improved Dihedral Parameters for the MMFF94s. *J. Cheminform.* **2019**, *11* (1), 53. https://doi.org/10.1186/s13321-019-0371-6.

(79) Rappe, A. K.; Casewit, C. J.; Colwell, K. S.; Goddard, W. A., III; Skiff, W. M. UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. *J. Am. Chem. Soc.* **1992**, *114* (25), 10024–10035. https://doi.org/10.1021/ja00051a040.

(80) Qiu, Y.; Smith, D. G. A.; Boothroyd, S.; Jang, H.; Hahn, D. F.; Wagner, J.; Bannan, C. C.; Gokey, T.; Lim, V. T.; Stern, C. D.; Rizzi, A.; Tjanaka, B.; Tresadern, G.; Lucas, X.; Shirts, M. R.; Gilson, M. K.; Chodera, J. D.; Bayly, C. I.; Mobley, D. L.; Wang, L.-P. Development and Benchmarking of Open Force Field v1.0.0-the Parsley Small-Molecule Force Field. *J. Chem. Theory Comput.* **2021**, *17* (10), 6262–6280. https://doi.org/10.1021/acs.jctc.1c00571.

For Table of Contents Only



3.3 Conclusion

In this chapter, the topic of conformer generation for small molecules was discussed. Also, the novel conformer generation algorithm CONFORGE was presented, it is available as part of the open-source toolkit CDPKit [40]. This new method can generate conformer ensembles for small molecules both more accurately and faster than the tested competitors while improving upon the accuracy of conformers for macrocyclic compounds even more drastically. Note that not only openly available tools were tested during the benchmarking, but also commercial-only ones, which were also outperformed. This was achieved by applying various code/algorithmic optimization strategies and creating a refined fragment library. For the conformer generation of large compounds, such as molecules with macrocycles, a custom stochastic sampling strategy was implemented, resulting in unparalleled accuracy compared to the tested state of the art methods.

The open availability of this fast and accurate method will be of great benefit to the whole chemoinformatics research community, as many computer aided drug design methods require compounds' conformations and their overall result quality strongly depends on the quality of the conformer ensembles.

Chapter 4

Graph Edit Distance Approximation

4.1 Introduction

Graphs are data structures that contain objects, represented as vertices, and their relationships/connections, represented as edges. They can be used to describe a variety of complex network-like structures such as the Internet, social networks, road networks (intersections and roads), or molecules (atoms and bonds).

In many of these applications, a measure of dissimilarity between two graphs is of interest. This measure can be thought of as a distance. Two graphs that are similar can be considered "close" to each other and thereby have a small dissimilarity/distance. Conversely, very dissimilar graphs can be considered far apart and should therefore have a large distance value. An intuitive definition for the distance between two equal graphs would thereby be 0, with a distance value greater than 0 for any unequal pair, whereby larger values indicate more dissimilar graphs. While there exist a variety of graph distance measures [26], one of the most commonly used ones that adheres to this intuitive idea is Graph Edit Distance.

Graph Edit Distance (GED) is defined as the minimum required total "cost" of transforming one graph into another [12]. This cost is determined by the edit path, describing which series of edit operations needs to be performed to modify one graph into the other. Edit operations have a predefined cost, or cost function, and commonly include changes such as adding, removing, or relabeling a vertex/edge. The difficulty when computing GED is that precisely the cheapest (optimal) edit path is required during computation, which leads to the problem being NP-hard, meaning it can only be computed exactly in a reasonable time for very small-sized graphs. The problem of this theoretical complexity can also be seen with the current state of the art algorithms, as none of them can reliably compute exact GED on graphs with more than 16 nodes. [5] Still, there exist applications where GED is of great use, such as ligand-based virtual screening of molecules [13], where larger graphs can occur and runtime per molecule is of utmost importance.

Due to the restrictive complexity of exact GED, approximation methods are often used in practice, as small inaccuracies are acceptable in many applications. There exist various heuristics that approximate GED [5], the most prominent one being Bipartite Graph Matching (BGM) [38] due to its relatively low computational complexity. Still, the accuracy of BGM often suffers from its emphasis on relatively small local neighborhoods around vertices, as it only considers vertex pairs with their immediate incident edges to build an assignment.

While methods that consider larger neighborhoods with the BGM methodology exist [6], they usually require multiple orders of magnitude longer to produce a GED approximation.

Contribution made

The goal was to design a method that is able to approximate Graph Edit Distance (GED) more accurately than the classical Bipartite Graph Matching (BGM) approach while retaining a similar runtime or at least not being slower than more accurate heuristics. To achieve this, the idea was to consider larger neighborhoods of the vertices, while not increasing runtime drastically. Previous attempts at this idea used subgraphs which are compared with exact GED [6] or bags of walks [14]. The use of exact GED to compare subgraphs makes the method a lot slower (multiple orders of magnitude) than the classical BGM, while the bags of walks approach has the downside of being difficult to parameterize such that accuracy is gained (tottering phenomenon [14]).

To combat these pitfalls, the use of trees to compare the neighborhoods of vertices for the computation of assignment costs was investigated. A prominent tree variant, which has found application in other graph-related problems such as isomorphism testing [19] or graph kernels [30], is the Weisfeiler Lehman tree. While those trees look promising at first, their exponential growth in size with increasing graph diameter makes them unfeasible for comparing neighborhoods.

As a solution to this, a novel tree representation of the neighborhood of a vertex with the help of neighborhood trees (and compressed neighborhood trees) is proposed. These trees abstract the surroundings of a vertex by including all shortest paths from the observed vertex to all other vertices that are reachable within a fixed distance. Not limiting the distance thereby results in a tree that is limited in depth by the diameter of the graph. Also, the applied compression methodology ensures that the size of the representations is limited by the number of vertices and edges in the graph, by transforming the trees into directed acyclic graphs.

With this novel tree representation combined with the fast BGM approach to approximate GED, the new method should provide an intuitive tuning parameter to increase the accuracy of approximation compared to classical BGM, by increasing or decreasing the depth of the trees, while mostly retaining BGM's runtime benefits. This should also be very beneficial when using GED with molecules, especially when they are large, as is the case with macrocyclic molecules.

4.2 Neighborhood Trees

Approximating the Graph Edit Distance with Compact Neighborhood Representations

1st Franka Bause
Faculty of Computer Science
University of Vienna
Vienna, Austria

1st Christian Permann
Research Platform NeGeMac
Department of Pharmaceutical Sciences
University of Vienna
Vienna, Austria

3rd Nils Kriege
Faculty of Computer Science
University of Vienna
Vienna, Austria

***Abstract*—The graph edit distance is used for comparing graphs in various domains and is primarily approximated due to its high computational complexity. Widely-used heuristics search for an optimal assignment of vertices based on the distance between local substructures. While fast ones only consider vertices and their incident edges, leading to poor accuracy, other approaches require computationally intense exact distance computations between subgraphs. Our new method abstracts local substructures to neighborhood trees and compares them using efficient tree matching techniques. This results in a ground distance for mapping vertices that yields high quality approximations of the graph edit distance. By limiting the maximum tree height, our method supports steering between more accurate results and faster execution. We thoroughly analyze the running time of the tree matching method and propose techniques to accelerate computation in practice. We use compressed tree representations, recognize redundancies by tree canonization and exploit them via caching. Experimentally we show that our method provides a significantly improved trade-off between running time and approximation quality compared to existing state-of-the-art approaches.**

I. Introduction

In recent years graphs have been widely used to model structured data and methods for analyzing them have gained in popularity. Graphs are commonly used in web content mining [1], pattern recognition [2], molecular property prediction [3], [4], any many other domains. Various tasks and applications require quantifying the similarity of two graphs. One of the most prominent measures of graph similarity is the *graph edit distance* (GED), i.e., the cost for transforming one graph into another by applying predefined edit operations. Usually adding, removing and relabeling vertices and edges is allowed, where each operation is assigned a specific cost. The computation of the graph edit distance from G to H then consists of finding a sequence of edit operations of minimum total cost that can be applied to G and yields a graph G' isomorphic to H. In practice, the usefulness of the exact GED is limited, as its computation is NP-hard [5]. This lead to the development of fast heuristic algorithms, such as *bipartite graph matching* (BGM) [6]. This method derives an edit path from an optimal (linear) assignment between the vertices of the two graphs. The cost for assigning a vertex v in $V(G)$ to a vertex v' in $V(H)$ reflects the cost for editing v and its incident edges to match v' and its incident edges. Vertex insertion and deletion is encoded by introducing additional placeholder vertices. The derived edit path is not necessarily optimal, but its cost is used as an approximation of the GED in many domains [7]. However, there are applications, for which the approximation quality obtained using this method is not sufficient, or that could benefit from more accurate results, such as graph clustering, k-nn classification or similarity search in graph databases. Hence, extensions of BGM have been proposed, which take enlarged local substructures around vertices into account. However, these either capture only limited structural information using walks [8] or require exact GED computation of subgraphs [9] leading to a drastic increase in both memory and time complexity. For this reason, we propose a new GED heuristic following the concept of BGM, which includes local information in the form of *neighborhood trees* to drastically improve the trade-off between running time and approximation quality. We compare neighborhood trees by a *structure and depth preserving tree edit distance* (SDTED), which is highly efficient compared to computing the exact GED, while retaining the benefits of incorporating expressive structural information.

The concept of neighborhood trees is inspired by so-called *unfolding trees* related to the Weisfeiler-Leman heuristic for the graph isomorphism problem. The method iteratively refines vertex labels encoding vertex neighborhoods of increasing size. The technique has recently become popular in graph learning, where it forms the basis of successful graph kernels [10] and graph neural networks [10], [11]. Graph neural networks have been shown to be equivalent to the Weisfeiler-Leman method regarding their expressivity, i.e., the ability to distinguish non-isomorphic graphs [12], [13]. The discrete colors representing unfolding trees have been used to obtain a highly efficient heuristic for the GED ignoring their subtle structural differences at the expense of approximation quality [14]. Also, graph neural networks have been employed to predict the GED for graph pairs, but the technique does not allow to obtain a corresponding edit path [15]. Unfolding trees are a suitable concept for understanding the structure encoded by iterative methods based on direct neighborhoods, but are rarely generated explicitly. A main reason for this is that they quickly grow in size with increasing height. Our concept of

*Authors contributed equally to this work

neighborhood trees overcomes this problem by pruning repeated vertices and allowing compact representations without redundancy amenable to efficient tree matching.

Our contribution. We make the following contributions.

1) We propose to use tree structures encoding node neighborhoods and their SDTED in the BGM framework for approximating the graph edit distance.
2) We propose *neighborhood trees*, a compact and powerful variation of the Weisfeiler-Leman unfolding trees.
3) We thoroughly analyze the running time of the SDTED and improve it to $O(n^2\Delta)$ for two trees with n vertices and maximum degree Δ reducing it by a factor of $\Delta^2 h$ over the best previously known result [16], where h is the height of the tree.
4) We apply caching techniques and compressed representations to drastically speed-up the computation in practice.
5) We show that our new approach outperforms state-of-the-art approximation algorithms for the GED regarding approximation quality while being computed efficiently.

II. Preliminaries

In this section, we give an overview of the necessary definitions and the notation used throughout the article. We first introduce graphs, the graph edit distance and a standard approximation scheme for it. Then we provide an introduction to the Weisfeiler-Leman algorithm and the SDTED.

A. Graph Theory

A *graph* $G = (V, E, \mu, \nu)$ consists of a set of vertices V, a set of edges $E \subseteq V \times V$ between them and labeling functions $\mu\colon V \to L$ and $\nu\colon E \to L$ for the vertices and edges, respectively. In this article, we assume that L is a set of categorical labels. We consider undirected graphs and refer to an edge between u and v by $uv = vu$. The vertices and edges of a graph G are denoted by $V(G)$ and $E(G)$, respectively. The *neighbors* of a vertex $u \in V$ are denoted by $N(u) = \{v \mid uv \in E\}$. A *rooted tree* T is a connected, acyclic graph with a distinct vertex $r \in V(T)$ referred to as *root* denoted by $r(T) = r$. For $v \in V(T)\backslash\{r\}$ the *parent* $p(v)$ is defined as the unique vertex $u \in N(v)$ closer to r than v, and $\forall v \in V(T)$ the *children* are defined as $\mathrm{chi}(v) = N(v)\backslash\{p(v)\}$. An *isomorphism* between two graphs G and H is a bijection $\Phi\colon V(G) \to V(H)$ such that (i) $\forall u, v \in V(G)\colon \mu(v) = \mu(\Phi(v))$, and (ii) $\Phi(u)\Phi(v) \in E(H) \Leftrightarrow uv \in E(G)$, and (iii) $\forall uv \in E(G)\colon \nu(uv) = \nu(\Phi(u)\Phi(v))$. For two rooted trees T and T' in addition $\Phi(r(T)) = r(T')$ must hold. If an isomorphism between two graphs G and H exists, they are called *isomorphic*, denoted by $G \cong H$.

B. Graph Edit Distance

The graph edit distance (GED) is the cost of transforming one graph into the other by deletion, insertion and relabeling of vertices and edges. An *edit path* between G and H is a sequence $(e_1, e_2, \ldots, e_k)$ of such edit operations that, when applied one after the other to G, yield a graph $G' \cong H$. Edit operations have non-negative costs given by the edit cost function c. The GED is the cost of a cheapest edit path. Formally, the *graph edit distance* between two graphs G and H is defined as

$$\mathrm{GED}(G, H) = \min\left\{ \sum_{i=1}^{k} c(e_i) \;\middle|\; (e_1, ..., e_k) \in \Upsilon(G, H) \right\},$$

where $\Upsilon(G, H)$ denotes the set of all possible edit paths from G to H. Since computing the GED is NP-hard [5], it is often approximated. Many heuristics rely on finding an optimal assignment between the vertices of the graphs. Let A and B be two sets with $|A| = |B| = n$ and $c\colon A \times B \to \mathbb{R}$ a ground cost function. An *assignment* between A and B is a bijection $\pi\colon A \to B$. The *cost* of an assignment π is $c(\pi) = \sum_{a \in A} c(a, \pi(a))$. The *assignment problem* is to find an assignment with minimum cost and can be solved in cubic time using the Hungarian method [17].

C. Bipartite Graph Matching

An upper bound for the GED between two graphs can be obtained from any edit path between them. An edit path can be derived from an optimal assignment of their vertices regarding a cost matrix. Riesen and Bunke [6] constructed a $k \times k$ cost matrix C, with $k = n + m$ for two graphs with n and m vertices, respectively. For simplicity, we present a reduced matrix with $k = \max\{n, m\}$ that is sufficient for metric edit costs [18]. Due to the symmetry of the GED in this case no vertices need to be inserted. Assume w.l.o.g. that $n \geq m$, then we obtain a quadratic $n \times n$ cost matrix by adding $n - m$ columns representing deletion of vertices in G by defining

$$C = \left[\begin{array}{cccc|ccc} c_{1,1} & c_{1,2} & \ldots & c_{1,m} & c_{1,\epsilon} & \ldots & c_{1,\epsilon} \\ c_{2,1} & c_{2,2} & \ldots & c_{2,m} & c_{2,\epsilon} & \ldots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \ldots & c_{n,m} & c_{n,\epsilon} & \ldots & c_{n,\epsilon} \end{array}\right].$$

The entries of C can be interpreted as follows: Values $c_{i,j}$ (left part) represent the costs for replacing vertex i of G with vertex j of H. The entries $c_{i,\epsilon}$ (right part) are the costs for deleting vertex i of G and its incident edges. Based on this cost matrix an optimal assignment is computed. An equivalent solution can be obtained from an $n \times m$ cost matrix by a more involved reduction applicable even for non-metric edit costs [19].

An edit path can be derived from a vertex assignment $\pi\colon V(G) \to V(H)$ as follows [6]: Any vertex (i) $v \in V(G)$ with $\pi(v) \notin V(H)$ is deleted, (ii) $v \in V(H)$ with $\pi^{-1}(v) \notin V(G)$ is inserted, (iii) $v \in V(G)$ with $\pi(v) \in V(H)$ and $\mu(v) \neq \mu(\pi(v))$ is relabeled. Similarly any edge (i) $uv \in E(G)$ with $\pi(u)\pi(v) \notin E(H)$ is deleted, (ii) $uv \in E(H)$ with $\pi^{-1}(u)\pi^{-1}(v) \notin E(G)$ is inserted, (iii) $uv \in E(G)$ with $\pi(u)\pi(v) \in E(H)$ and $\nu(uv) \neq \nu(\pi(u)\pi(v))$ is relabeled. The sum of edit costs is the approximated GED. Since the edit path found might not be optimal, this method provides an upper bound.

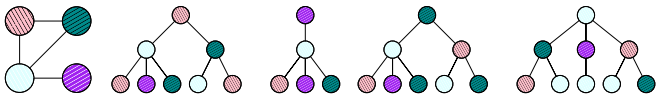


Fig. 1: Graph G and its unfolding trees $F_2^v, v \in V(G)$.

D. Weisfeiler-Leman Color Refinement

The 1-dimensional Weisfeiler-Leman (WL) algorithm or *color refinement* starts with all vertices having a color representing their label (or a uniform coloring in case of unlabeled vertices). In each iteration the color of every vertex v in $V(G)$ is updated based on the multiset of colors of its neighbors according to

$$c_{i+1}(v) = h\left(c_i(v), \{\!\{c_i(u) \mid u \in N(v)\}\!\}\right),$$

where h is an injective function. Since every graph isomorphism associates vertices having the same color only, the approach allows to recognize graphs as non-isomorphic. Non-isomorphic graphs with the same multiset of colors for all $i \in \mathbb{N}$ exist, but they are rare even for $i = 2$ [20] and many real-world graph datasets do not contain such graph pairs [21]. The expressivity and computational efficiency of the method makes it an important tool for graph isomorphism testing and machine learning [11], [21].

The colors encode neighborhoods of increasing radius by a tree structure, see Figure 1. Mathematically, the *unfolding tree* F_i^v with height i of the vertex $v \in V(G)$ is defined recursively as the tree with a root representing v and children $N(v)$. Each child $w \in N(v)$ is the root of the unfolding tree F_{i-1}^w and $F_0^w = (\{w\}, \emptyset)$. The labels of the original graph are preserved. The unfolding trees F_i^v and F_i^w of two vertices v and w are isomorphic if and only if $c_i(v) = c_i(w)$.

E. Structure and Depth Preserving Tree Edit Distance

For rooted trees we use the *structure and depth preserving tree edit distance* (SDTED) [16], which is defined as

$$\mathrm{SDTED}(T, T') := \min\{c'(M) \mid M \in \mathrm{SDM}(T, T')\},$$

where $\mathrm{SDM}(T, T')$ denotes the set of all structure and depth preserving mappings between T and T'. A mapping $M \subseteq V(T) \times V(T')$ is *structure and depth preserving*, iff (i) $\forall (u, u'), (v, v') \in M\colon\ u = v \Leftrightarrow u' = v'$, (ii) $(r(T), r(T')) \in M$, and (iii) $\forall (u, u') \in M\colon$ $(p(u), p(u')) \in M$. Let $M_1 = V(T) \setminus \{m_1 \mid (m_1, m_2) \in M\}$ and $M_2 = V(T') \setminus \{m_2 \mid (m_1, m_2) \in M\}$ be the vertices not in the mapping. The cost c' of the mapping M is defined as

$$c'(M) = \sum_{(u,v)\in M} c(\mu(u), \mu(v)) + \sum_{u \in M_1} c(u, \epsilon) + \sum_{v \in M_2} c(\epsilon, v).$$

We can respect edge labels by associating the relabeling costs with the endpoint that is the child of the other endpoint. Given two trees, the SDTED can be computed in a bottom-up fashion from optimal assignments between the children of two vertices at the same level until reaching the roots, see the Appendix for a recursive formulation.

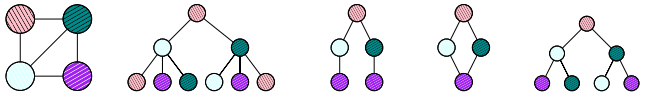
III. Related Work

Computing the GED is a combinatorial optimization problem and different standard approaches for such problems have been applied. Some exact methods are based on backtracking search, e.g., CSI-GED [22] or BSS_GED [23], while others use integer linear programming formulations such as BIP-GED [24]. The complexity, however, makes exact approaches infeasible for large graphs and approximate methods are widely-used [7].

Many approximation methods [5], [6], [25], [26], [9] are based on optimal assignments. Usually, faster methods are only suitable for graphs with discrete labels and uniform edit costs. A very general, tight lower bound, which can handle continuous labels and arbitrary (metric) edit costs, is Branch [25]. It follows the concept of BGM introduced by Kaspar Riesen and Horst Bunke [6], but guarantees that the assignment costs are a lower bound of the GED. Existing variants use larger neighborhood subgraphs by computing the exact GED between them with the downside of drastically increased computational complexity [9]. Others encode neighborhoods by bags of walks [8], where only the start and end vertices are considered. This comes at the cost of losing local structure information, as all intermediate vertices are omitted. Moreover, the approximation quality decreases for walks with five or more vertices due to the phenomenon of *tottering* [8].

Our idea of neighborhood trees is inspired by Weisfeiler-Leman unfolding trees. Graph and vertex similarities based on the WL algorithm have been studied extensively in machine learning [21]. Several graph kernels count matching WL colors [10], use them to compute optimal vertex assignments [27] or define similarities between colors, e.g., by matching unfolding trees [16]. A GED approximation based on finding an optimal vertex assignment regarding a tree metric generated by the WL algorithm was proposed by Kriege et al. [14]. This approach uses matching WL colors and implicitly compares the unfolding trees of two vertices level-wise (from root to leaves) and determines the similarity as the number of levels up to which the unfolding trees are isomorphic. This leads to a coarse similarity, in which vertices with different labels, but equal neighborhoods are considered less similar than vertices with the same label and completely different neighborhoods. While the technique is faster than comparable BGM methods, it is less accurate on some datasets.

GED approximation methods not based on BGM are often multiple orders of magnitude slower, while improving accuracy only marginally [26]. This includes methods such as the LP relaxations of ILP formulations [24], which may yield tighter lower bounds but, again, at drastically increased runtime. Graph neural networks have been used for predicting the GED for pairs of graphs based on their vertex- and graph-level embeddings [15]. This technique, however, does not yield an edit path and cannot guarantee that the result is a lower or upper bound for the GED.



(a) G (b) Unfolding tree (c) 0-NT (d) cNT (e) 1-NT

Fig. 2: Graph G, and height 2 trees of upper left vertex.

IV. Neighborhood Trees for Graph Matching

Accurate approximation of the GED through BGM requires representations of sufficiently large and complex vertex neighborhoods, which allow the efficient computation of suitable and expressive distances. We propose to use trees for representing neighborhoods and use the SDTED between such trees as a cost function for the assignment. Trees capture the vertex neighborhoods well and their structure can be exploited to speed up distance computation. This leads to significant speed-ups over neighborhood subgraphs. We investigate traditional WL unfolding trees for this purpose, which quickly grow in size with increasing height and store a large amount of redundant information. To mitigate this, we propose k-*redundant neighborhood trees*, which are equivalent to WL unfolding trees for sufficiently large k but exhibit less redundancy for smaller values of k leading to significantly more compact representations. Of particular interest are the 0-redundant neighborhood trees, which represent a vertex with its surroundings by a rooted tree of shortest paths up to a maximum length restricting the radius of the surroundings. Bounding the tree height allows to control the trade-off between more accurate results and faster execution.

A. Definitions and Properties

We formalize the concept of neighborhood trees. A k-redundant neighborhood tree can be constructed from the corresponding unfolding tree by deleting all subtrees with its roots that already occurred more than k levels before. Let $\mathrm{depth}(v)$ denote the length of the path from v to the root and $\phi(v)$ denote the original vertex in $V(G)$ represented by v in the unfolding or neighborhood tree.

Definition 1 (k-redundant Neighborhood Tree, k-NT)**.** *For $k \geq 0$, the* k-redundant neighborhood tree *of a vertex $v \in V(G)$ with height i, denoted by $T^v_{i,k}$, is defined as the subtree of the unfolding tree F^v_i induced by the vertices $u \in V(F^v_i)$ satisfying*

$$\forall w \in V(F^v_i)\colon \phi(u) = \phi(w) \Rightarrow \mathrm{depth}(u) \leq \mathrm{depth}(w) + k.$$

Note that for $k \geq i$ the k-redundant neighborhood tree is equivalent to the WL unfolding tree. We call the 0-redundant neighborhood tree simply *neighborhood tree* (NT), see Figure 2 for an example. In neighborhood trees, some edges of the graph are not represented, e.g., the diagonal edge between the blue and the green vertex of graph G in Figure 2. Hence, we also investigate 1-redundant neighborhood trees, which include such edges (see the Appendix for details).

Neighborhood trees may contain multiple instances of the same vertex on the same level only. In general k-redundant neighborhood trees, these duplicates are again the roots of isomorphic subtrees and increase the size further. Therefore, we propose a *compressed neighborhood tree* (cNT) to reduce duplicate vertices. Given two vertices u and v on the same level of a rooted tree T, by *merging u onto v* we refer to the operation that adds the edge $p(u)v$ to T and deletes the subtree rooted at u. When merging two vertices, we assume that no vertices with lower depth can be merged.

Definition 2 (Compressed Neighborhood Tree, cNT)**.** *The* compressed neighborhood tree $C^v_{i,k}$ *of $v \in V(G)$ is the reduction of $T^v_{i,k}$ obtained by merging vertices u onto vertices w whenever $\phi(u) = \phi(w)$ and $\mathrm{depth}(u) = \mathrm{depth}(w)$.*

Note that the rooted subtrees of two vertices that are merged are isomorphic leading to a well-defined compressed neighborhood tree independent of the order in which vertices at the same level are merged. The size and height of a cNT are bounded as follows.

Theorem 1. *The cNT of a k-redundant neighborhood tree rooted at any vertex of a graph G has size at most $O(|E(G)| \cdot (k+1))$ and height at most $\mathrm{diam}(G) + k$, where $\mathrm{diam}(G)$ is the diameter of G.*

Proof. A cNT $C^v_{\infty,1}$ rooted at vertex $v \in V(G)$ includes all shortest paths from v to any vertex u which have length at most $\mathrm{diam}(G)$. As the parameter k in k-NTs allows for up to k repetitions of vertices in subsequent layers, at most k layers are added to the compressed k-NT through repetition of the vertex $w \in V(G)$ where $\forall u \in V(G)\colon \mathrm{depth}(u) \leq \mathrm{depth}(w)$. Thereby, the maximum height is bounded by $\mathrm{diam}(G) + k$. Since no vertex can occur multiple times in a single layer and a vertex can only occur in $1 + k$ consecutive layers, we can infer an upper bound on the size of any compressed k-NT. Any vertex $u \in V(G)$ can occur at most $k + 1$ times together with its incident edges. As $|E(G)| \geq |V(G)| + 1$ the size of any $C^v_{i,k}$ is bounded by $O(|E(G)| \cdot k)$. □

Restricting the maximum height reduces the size of the tree further by pruning deeper levels. The compressed tree $C^v_{i,k}$ can be created directly instead of deriving it from the full tree by merging vertices.

B. Creating Neighborhood Trees

Algorithm 1 shows how to generate compressed neighborhood trees according to Definitions 1 and 2 with $k = 0$ and a maximum height h. For the creation of a tree the chosen vertex is set as its root and iterative refinement is performed. The iterative refinement method explores the neighborhood level-wise in a breadth-first search fashion. We build the cNT by appending newly found children to their parents instead of inserting them into a queue. If a child was already found at a layer and a corresponding vertex has been created in the cNT, only the missing edge is inserted. If the refinement did not extend the tree by a new level, its height remains unchanged. In this case the refinement procedure terminates immediately, otherwise it continues until the maximum height

Algorithm 1 Creation of compressed neighborhood trees

```
function BUILDNT(Graph G, height h)
    T ← ∅                                          ▷ T: all cNTs
    for each v ∈ V(G) do
        T ← cTree(v)                               ▷ initialize cNT
        i ← 0                                      ▷ refinement height
        while i < h and HEIGHT(T) = i do
            REFINE(T, G)
            i ← i + 1
        T ← T ∪ {T}
    return T

function REFINE(cTree T, Graph G)
    d ← HEIGHT(T)
    L_d ← {v ∈ V(T) | depth(v) = d}
    for each v ∈ L_d do
        for each e ← vu ∈ E(G) do ▷ edges incident to v
            if u ∉ V(T) then
                T.ADD(v, u, e)      ▷ add child u to v with e
            else if depth(u) = d + 1 then
                T.ADDEDGE(v, u, e)  ▷ add e from v to u
```

h is reached. Our datastructures for (compressed) neighborhood trees maintain information such as the vertex depth, the set L_d of potentially extendable leaves, and whether a vertex is already represented to achieve constant time look-ups. With these techniques Algorithm 1 constructs a compressed neighborhood tree of a given vertex in time $O(|E(G)|)$.

C. Neighborhood Tree Edit Distance

We compare pairs of (compressed) neighborhood trees using the SDTED, see Section II-D, to obtain a distance between vertex pairs reflecting the dissimilarity of the neighborhoods for use in the BGM framework. We introduce a weight w to control the contribution of layers based on their depth. The cost of an edit operation at depth l is multiplied by w^l. For $w = 1$ we obtain the unweighted case and we chose $w < 1$ to reduce the influence of deeper levels.

The SDTED can be solved recursively starting at the root vertices and computing optimal assignments between their children, where the cost for matching two children depends on the SDTED of their subtrees, see the Appendix for details and pseudocode. Schulz et al. [16] used this approach with memoization to obtain a running time of $O(nn'h\Delta^3)$ for two WL unfolding trees with n and n' vertices, height h and a maximum degree of Δ. We improve the running time by introducing techniques for efficient tree matching into the SDTED computation and refining the analysis.

Theorem 2. *The SDTED of two trees with n and n' vertices and maximum degree Δ can be computed in $O(nn'\Delta)$ time.*

Proof. Consider two trees $T = (V, E)$, $T' = (V, E')$ with n and n' vertices, respectively. We have to solve a linear assignment problem for each pair of vertices $(i, j) \in V \times V'$ with $\text{depth}(i) = \text{depth}(j)$. Let n_i and n_j denote the number of children of i and j, respectively. Introducing placeholder vertices for insertion and deletion leads to a squared cost matrix with $n_i + n_j$ rows and columns [16]. However, we can find an equivalent solution using the reduction of Bougleux et al. [19] resulting in an $n_i \times n_j$ cost matrix C. The arising problems can be solved by maximum weight matching algorithms for unbalanced bipartite graphs in time $O(ms + s^2 \log s)$, where m is the number of edges, i.e., non-zero entries in C, and s the smaller vertex set [28]. From this we obtain an upper bound of $O(n_i n_j \Delta)$ for our setting, since $m \leq n_i n_j$ and $s = \min\{n_i, n_j\} \leq \Delta$. This leads to a total running time of

$$\sum_{i \in V} \sum_{j \in V'} O(n_i n_j \Delta) = O\left(\sum_{i \in V} n_i \sum_{j \in V'} n_j \Delta\right) = O(nn'\Delta).$$

□

In the case of integral edit costs bounded by K, the costs in all assignment problems are bounded by $C = K(n + n')$. We can use the bipartite matching algorithm by Goldberg et al. [29] to obtain a total running time of $O(nn'\sqrt{\Delta}\log(\Delta C))$ using the same arguments.

D. GED Approximation with SDTED Caching

Based on the neighborhood tree edit distance we obtain a cost matrix, which is then used to compute a minimum assignment between the vertices of the two graphs. From the assignment an upper bound for the graph edit distance is then derived yielding our approximation. Combining Theorem 1 and Theorem 2 we obtain a total running time of $O(nn'mm'\Delta)$ for computing the cost matrix for two graphs with n and n' vertices, m and m' edges, and maximum degree Δ by applying the SDTED to all cNTs of vertex pairs. The running time for creating the cost matrix dominates the total running time of our approach. We describe implementation details and techniques to speed-up the computation in practice.

The most significant acceleration is achieved by storing and reusing the results of SDTED computations for neighborhood trees and subtrees. To this end, we use canonical string encodings of trees (further detailed in the Appendix), which are equal for two trees if and only if the trees are isomorphic. While no polynomial-time algorithm is known for the graph canonization problem [30], canonical encodings of unlabeled trees can be computed in linear-time by classical algorithms [31], which sort siblings level-wise in a bottom-up fashion. While for unlabeled graphs sorting can be realized in linear-time using bucket sort, for arbitrary label alphabets $O(n \log n)$ time is required to order the tree unambiguously. From this, a string encoding is derived via tree traversal serving as a unique identifier. In our method, this technique is used for accessing a hash-based data structure storing the SDTED between trees by using tuples of canonical encodings as key. Here, we exploit the symmetry of the SDTED by ignoring the order of the two encodings in the tuple.

We implemented the tree canonization procedure such that it generates canonical encodings of all subtrees as a byproduct, which are also cached in the same way. This means that, not only the encodings of subtrees can be reused for computing the encoding of all parents, but also the repeated computation of SDTED between subtrees can be avoided. During the first recursive step, where the overall encodings are computed, all subtree encodings are also computed and stored.

V. Experimental Evaluation

We compare our newly proposed technique to state-of-the-art approaches regarding approximation quality and runtime. Specifically, we address the following research questions:

Q1 How tight are the upper bounds of our method compared to the state-of-the-art?

Q2 How do our bounds perform when taking the trade-off between bound quality and runtime into account?

In the Appendix, we also investigate the speed-up gained by our caching strategy.

A. Setup

This section gives an overview of the datasets, the methods used in the experimental comparison and their implementation and configuration.

1) Methods and Distance Functions: We compare our approach to the state-of-the-art bipartite graph matching methods *BGM* [6], its extensions using bags of walks [8] and subgraphs [9], as well as to the state-of-the-art exact GED method *BSS_GED* [23], as we focus on solution quality while not being orders of magnitude slower than *BGM*. For our method, we compare the assignments under the SDTED using the traditional unfolding trees *WL* and neighborhood trees *NT*, while varying the height parameter. We employ the optimizations described in the previous section applying compression for *WL* and *NT*. The weight parameter w for the SDTED computation was set to 0.5 for all datasets, which was found to yield good results in preliminary experiments. Since the exact approach can handle uniform edit costs only, we use these in our experiments.

2) Implementation: We implemented our newly proposed methods, as well as *BGM* and its extensions, in Java. For our competitors methods, we followed the implementations presented by the authors. We used the C++ implementation of *BSS_GED* provided by the authors. All experiments were conducted on an Intel Xeon Gold 6130 machine at 2.1 GHz with 96 GB RAM. We report average execution time over 5 runs. Our implementation is available at *http://anonymized*.

3) Datasets: We tested all methods on a wide range of real-world datasets from the TUDataset collection [32] with different characteristics, see Table I. These datasets are widely used for graph similarity computation and have discrete vertex and edge labels. Attributes, if present, were removed prior to the experiments since not all methods support them. We randomly sampled 1000 graphs from the dataset *Protein Com* [33].

TABLE I: Characteristics of the datasets.

Name	$\lvert\mathbf{G}\rvert$	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	$\lvert L_V\rvert$	$\lvert L_E\rvert$	$\overline{diam.}$
Letter-med	2250	4.67	4.50	–	–	2.44
MUTAG	188	17.93	19.79	7	4	8.22
PTC_FM	349	14.11	14.48	18	4	7.37
Protein Com	1000	10.19	9.19	622	–	4.76
MSRC_9	221	40.58	97.94	10	–	7.00
NCI1	4110	29.87	32.30	37	–	13.33

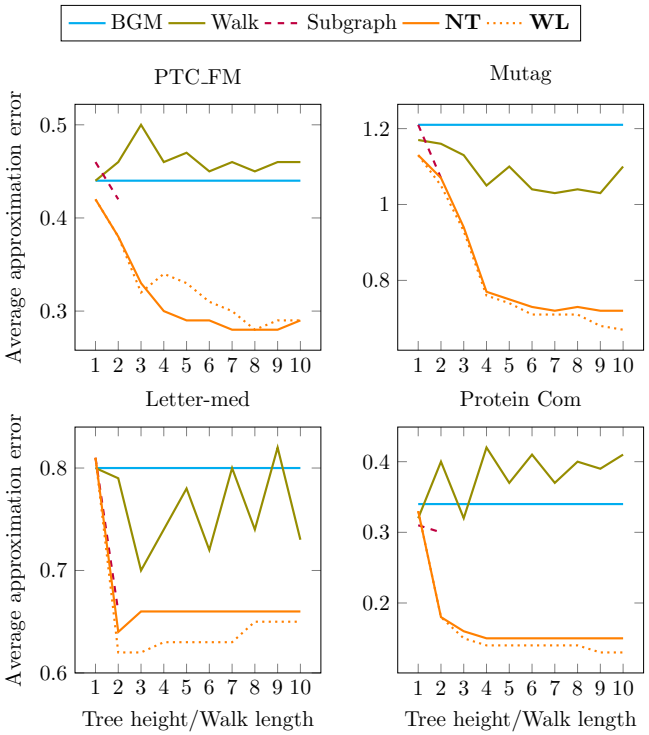


Fig. 3: Average relative error (proposed in bold).

B. Results

In the following, we report on our experimental results and discuss the different research questions.

1) Q1: Bound Quality: Tight bounds are essential, especially when the exact graph edit distance is small. In this section we compare our new approaches with state-of-the-art bipartite graph matching. We first investigate how many iterations are needed for the different tree structures to achieve the best accuracy. The average relative approximation error of the different methods regarding the exact graph edit distance is shown in Figure 3. Since *BGM* does not have any parameter to adjust the tree height, walk length or subgraph size, the approximation error is constant and it is used as a baseline. While we gain a huge advantage initially in all datasets, the approximations do not get much better after a certain tree height. An explanation for this is that many graphs are already fully explored by NTs of small height and not many vertices are added in later refinement steps. The *Walk* method performed sometimes worse and sometimes better than

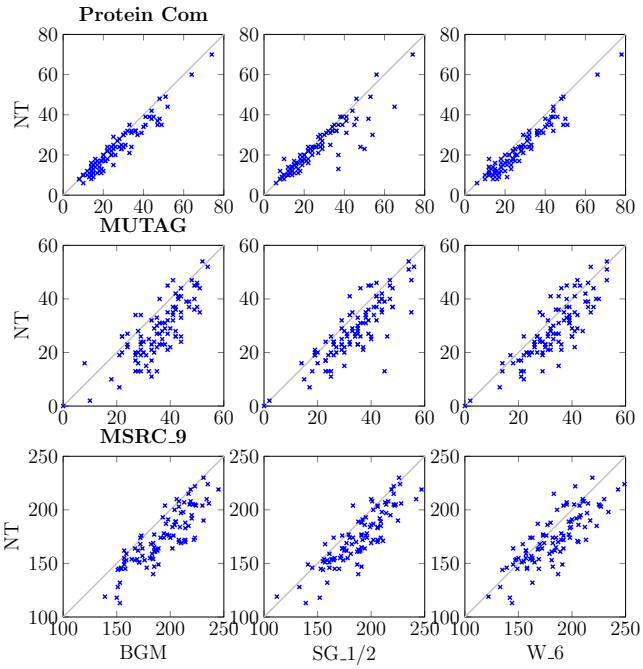


Fig. 4: Approximation quality of *NT_10* in comparison to the different state-of-the-art methods on the datasets MUTAG and MSRC_9. The methods are denoted with variant_height.

the baseline *BGM*, while always performing worse than our approach. Additionally, the method has unpredictably varying accuracy with increasing walk length, often even leading to degraded accuracy. The *Subgraph* method could only be tested with subgraph sizes of 1 and 2 due to the huge increase in runtime (see Figure 5). For these parameter settings, the results were comparable to or worse than our method.

We compare our approximations to the different state-of-the-art methods in Figure 4 and the corresponding figure in the Appendix. Points that are below the gray line indicate a better approximation by our approach. Since almost all points are below this line, we can conclude, that our method performs generally better, even on datasets with larger graphs.

2) Q2: Runtime: There is typically a trade-off between runtime and accuracy. In this section we evaluate our proposed approaches in terms of runtime. Figure 5 shows the average runtime for computing the distance between two graphs using the different methods. Exact computation is much slower than most approximations and shown as a baseline (for the datasets *MSRC_9* and *NCI1* no values are given, since the graphs are already too large for this method). Since *BGM* only computes one larger assignment (and one small assignment for each vertex pair), we can expect it to be faster than our newly proposed methods. In practice we see that the runtime difference between *BGM* and *NT* is quite small, while our *WL* variant is much slower. Since the unfolding trees in the *WL* variant contain many redundant vertices and grow very fast in size with increasing height, this is in accordance with our expectation. The plateauing of the runtime for *NT* can be

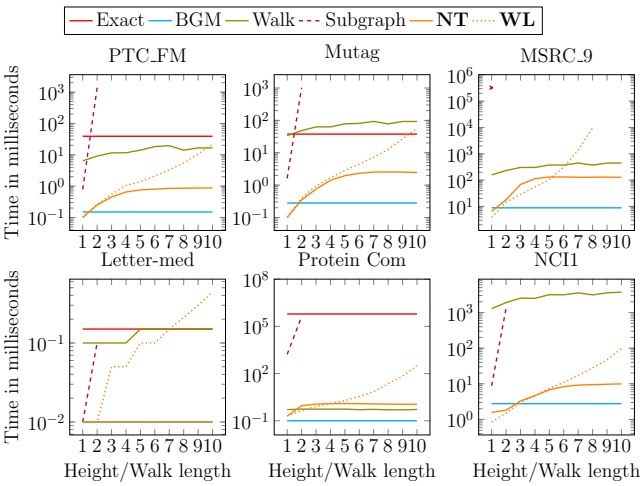


Fig. 5: Average runtime for computing the distance between two graphs (proposed in bold).

explained by the bounded size of the NTs. After every edge in the graph is explored, the tree will not grow in size anymore. The *Subgraph* method becomes very slow when increasing the size of the subgraphs, to the point where exact GED computation is faster than the approximate method. For this reason, we did not test subgraph sizes of 3 and larger. The runtime of *Walk* varies a lot between the datasets relative to the other methods. Seemingly, *Walk* is fastest when the number of distinct labels is large, due to the small number of matching node pairs in the method's product graph. While *Walk* was the fastest method other than *BGM* for the Protein Com dataset, it is important to note, that it performed worse than the baseline *BGM*.

3) Discussion: The experiments showed that, while our approach provides tighter bounds for the graph edit distance, it is only marginally slower than state-of-the-art bipartite graph matching. Especially for graphs with small diameter, a small tree height is often sufficient. We observed, that in contrast to bags of walks, where large walk lengths can decrease the method's accuracy due to the "tottering phenomenon" [8], the accuracy of our method increases with larger tree height. This indicates, that a good choice for our height parameter h can be easily and intuitively found by a user, as it trades accuracy for runtime. Our experiments have shown that not limiting h still results in a fast method, as the increase in computational complexity arising from the parameter h is bounded by the diameter of the graphs.

VI. Conclusions

We introduced a new bipartite graph matching based GED approximation. For the vertex assignment, neighborhood trees and their edit distance are used. Thereby, our approach takes complex structural information around the vertices into account without performing expensive exact GED computations for local subgraphs. As our trees can be limited in height, an individual trade-off between runtime and accuracy can

be achieved. In our experimental evaluation we showed that the proposed method is only slightly slower than the state-of-the-art approach, while providing great benefits regarding approximation accuracy.

In future work we will investigating other practical applications of our method, such as database search, where storing the computed SDTEDs globally seems promising. Furthermore, our method can be applied to molecular graphs for structure-based virtual screening of ligands [4]. The improved approximation quality with only small overhead is promising especially when working with larger molecules such as macrocyclic compounds [34]. The WL algorithm is fundamental for graph learning methods such as graph kernels and graph neural networks [21]. Various applications in this field could potentially benefit from replacing WL unfolding trees by more compact neighborhood trees.

REFERENCES

[1] A. Schenker, A. Kandel, H. Bunke, and M. Last, *Graph-Theoretic Techniques for Web Content Mining*, ser. Series in Machine Perception and Artificial Intelligence. WorldScientific, 2005, vol. 62.

[2] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *International Journal of Pattern Recognition and Artificial Intelligence*, 02 2014.

[3] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert, "Graph kernels for molecular structure-activity relationship analysis with support vector machines," *Journal of Chemical Information and Modeling*, vol. 45, no. 4, pp. 939–951, 2005.

[4] C. Garcia-Hernandez, A. Fernández, and F. Serratosa, "Ligand-based virtual screening using graph edit distance as molecular similarity measure," *Journal of Chemical Information and Modeling*, vol. 59, no. 4, pp. 1410–1421, 2019.

[5] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou, "Comparing stars: On approximating graph edit distance," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 25–36, 2009.

[6] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image Vision Comput.*, vol. 27, no. 7, pp. 950–959, 2009.

[7] M. Stauffer, T. Tschachtli, A. Fischer, and K. Riesen, "A survey on applications of bipartite graph edit distance," in *Graph-Based Representations in Pattern Recognition, GbRPR*, 05 2017.

[8] B. Gaüzère, S. Bougleux, K. Riesen, and L. Brun, "Approximate graph edit distance guided by bipartite matching of bags of walks," in *Structural, Syntactic, and Statistical Pattern Recognition, S+SSPR*, 08 2014, pp. 73–82.

[9] V. Carletti, B. Gaüzère, L. Brun, and M. Vento, "Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance," in *Graph-Based Representations in Pattern Recognition*, 2015, pp. 188–197.

[10] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Appl. Netw. Sci.*, vol. 5, no. 1, p. 6, 2020.

[11] C. Morris, Y. Lipman, H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. M. Borgwardt, "Weisfeiler and Leman go machine learning: The story so far," *CoRR*, vol. abs/2112.09992, 2021.

[12] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and Leman go neural: Higher-order graph neural networks," *AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4602–4609, Jul. 2019.

[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[14] N. M. Kriege, P. Giscard, F. Bause, and R. C. Wilson, "Computing optimal assignments in linear time for approximate graph matching," in *2019 IEEE International Conference on Data Mining, ICDM*, 2019, pp. 349–358.

[15] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "SimGNN: A neural network approach to fast graph similarity computation," in *ACM International Conference on Web Search and Data Mining*, 2019.

[16] T. H. Schulz, T. Horváth, P. Welke, and S. Wrobel, "A generalized Weisfeiler-Lehman graph kernel," *Machine Learning*, vol. 111, no. 7, pp. 2601–2629, Jul 2022.

[17] J. R. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, 1957.

[18] F. Serratosa, "Speeding up fast bipartite graph matching through a new cost matrix," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 29, no. 2, pp. 1 550 010:1–1 550 010:17, 2015.

[19] S. Bougleux, B. Gaüzère, D. B. Blumenthal, and L. Brun, "Fast linear sum assignment with error-correction and no cost constraints," *Pattern Recognit. Lett.*, vol. 134, pp. 37–45, 2020.

[20] L. Babai and L. Kucera, "Canonical labelling of graphs in linear average time," in *Symposium on Foundations of Computer Science*, 1979, pp. 39–46.

[21] C. Morris, M. Fey, and N. M. Kriege, "The power of the Weisfeiler-Leman algorithm for machine learning with graphs," in *IJCAI 2021*, 2021, pp. 4543–4550.

[22] K. Gouda and M. Hassaan, "Csi_ged: An efficient approach for graph edit similarity computation," *International Conference on Data Engineering, ICDE*, pp. 265–276, 2016.

[23] X. Chen, H. Huo, J. Huan, and J. S. Vitter, "An efficient algorithm for graph edit distance computation," *Knowledge-Based Systems*, vol. 163, pp. 762 – 775, 2019.

[24] J. Lerouge, Z. Abu-Aisheh, R. Raveaux, P. Héroux, and S. Adam, "New binary linear programming formulation to compute the graph edit distance," *Pattern Recognition*, vol. 72, pp. 254–265, 2017.

[25] D. B. Blumenthal and J. Gamper, "Improved lower bounds for graph edit distance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 3, pp. 503–516, 2018.

[26] D. B. Blumenthal, N. Boria, J. Gamper, S. Bougleux, and L. Brun, "Comparing heuristics for graph edit distance computation," *VLDB J.*, vol. 29, no. 1, pp. 419–458, 2020.

[27] N. M. Kriege, P.-L. Giscard, and R. C. Wilson, "On valid optimal assignment kernels and applications to graph classification," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, 2016, p. 1623–1631.

[28] L. Ramshaw and R. E. Tarjan, "On minimum-cost assignments in unbalanced bipartite graphs," HP Laboratories, Tech. Rep. HPL-2012-40, Jun. 2012.

[29] A. V. Goldberg, S. Hed, H. Kaplan, and R. E. Tarjan, "Minimum-cost flows in unit-capacity networks," *Theory Comput. Syst.*, vol. 61, no. 4, pp. 987–1010, 2017.

[30] M. Grohe and P. Schweitzer, "The graph isomorphism problem," *Commun. ACM*, vol. 63, no. 11, pp. 128–134, 2020.

[31] A. V. Aho and J. E. Hopcroft, *The Design and Analysis of Computer Algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1974.

[32] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. [Online]. Available: www.graphlearning.io

[33] B. K. Stöcker, T. Schäfer, P. Mutzel, J. Köster, N. M. Kriege, and S. Rahmann, "Protein complex similarity based on Weisfeiler-Lehman labeling," in *12th Int. Conf. Similarity Search and Applications, SISAP*, 2019, pp. 308–322.

[34] S. Sengupta and G. Mehta, "Macrocyclization via c–h functionalization: a new paradigm in macrocycle synthesis," *Organic & Biomolecular Chemistry*, vol. 18, no. 10, 2020.

APPENDIX

We give algorithmic details omitted in the main document and investigate 1-redundant neighborhood trees, which do not provide a huge benefit over 0-redundant neighborhood trees in most cases. In Section A we present the pseudocode for computing the SDTED. Section B gives an overview of the creation of k-redundant neighborhood trees, including implementation details. In Section C we give further details on implementation and optimization. We show the approximation results on the remaining datasets in Section D and evaluate the

Algorithm 2 Computation of SDTED

```
function SDTED(Tree T_1, Tree T_2)
    n ← max(|chi(r(T_1))|, |chi(r(T_2))|)
    PAD(T_1, n), PAD(T_2, n)        ▷ add undef. for equal size
    C ← R_{n×n}                     ▷ 0 initialized
    for each s_i ∈ chi(r(T_1)) do
        for each s'_j ∈ chi(r(T_2)) do
            if s_i is defined or s'_j is defined then
                if s_i is undefined then
                    C_ij ← DELETIONCOST(s'_j)
                else if s'_j is undefined then
                    C_ij ← DELETIONCOST(s_i)
                else
                    C_ij ← SDTED(s_i, s'_j)      ▷ child cost
    c_r ← COST(r(T_1), r(T_2))
    c_s ← LAPCOST(C)        ▷ min. subtree assignment cost
    return c_r + c_s
```

runtime and approximation quality of 1-NTs in Section E. In Section F we investigate the effect of our caching strategy on the runtime of our method.

A. SDTED

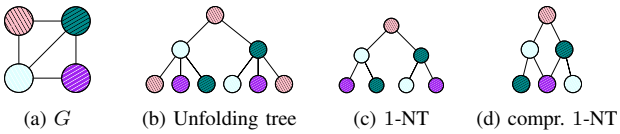
The structure and depth preserving tree edit distance can be used to compare two rooted trees. Intuitively, it compares the two trees layer by layer recursively by finding the minimum assignment cost between children and including that cost for assigning the parents in the layer above. A naive implementation of the SDTED can be seen in Algorithm 2. This simple variant does not include edge costs, but only vertex costs. Trees and subtrees with different numbers of children are padded with placeholder vertices such that deletion/insertion costs can be considered. The deletion cost of a whole subtree can be determined without additional recursion. For our implementation of the SDTED we included edge costs and store the costs for all computed pairs of trees and subtrees using a lexicographic encoding, see Section C.

B. k-redundant Neighborhood Trees

In variant 1-*NT* we investigate k-redundant neighborhood trees with $k = 1$, and allow for vertices to reappear one layer after their first occurrence (see Figure 6 for an example). Since vertices can reappear in different layers, the compressed tree includes the same vertex object in different parts of the tree. For this reason, vertex references are used such that every layer of our tree contains at most one distinct reference to a vertex, while two subsequent layers can contain two different references to the same vertex. Besides the usage of references, it only differs from *NT* regarding conditional checks due to allowing the additional vertices.

C. Lexicographic Encoding

In the computation of the SDTED, pairs of subtrees often occur multiple times. This is even more likely to happen if the original graph has symmetry or otherwise shows regularity.



(a) G (b) Unfolding tree (c) 1-NT (d) compr. 1-NT

Fig. 6: Graph G, and height 2 trees of upper left vertex.

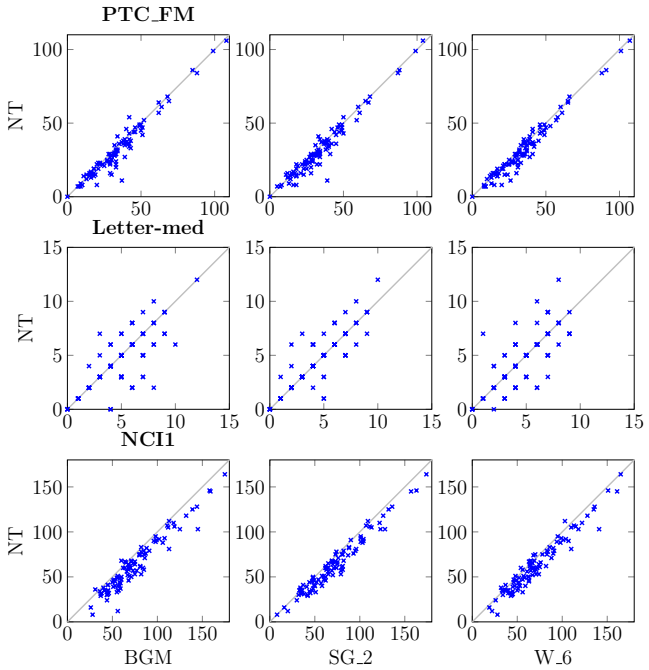


Fig. 7: Approximated graph edit distance of *NT_10* in comparison to the approximation computed by the state-of-the-art methods. The methods are denoted with variant_height.

Compression of trees alleviates some computational complexity of our method, but the repeated SDTED computation between the same subtrees is still costly. For this reason, we introduce a lexicographic encoding on our neighborhood trees similar to [31]. This encoding is used to store costs for pairs of trees or subtrees, by mapping their joint encoding to this value, thereby completely avoiding duplicate computations. As child encodings are stored during this computation, in practice, all (sub-)encodings of a tree are created with the first call for encoding the tree at its root.

Note that when using k-redundant neighborhood trees with $k > 1$, vertices are represented through references. This abstraction of the vertices does not impact the produced string of the lexicographic encoding but only avoids duplicate computation for references equal in memory address.

D. Approximation of the Graph Edit Distance

Here we show the results regarding the approximation of the graph edit distance for the remaining datasets. Figure 7 shows the relation between our approximation and the state-of-the-art methods.

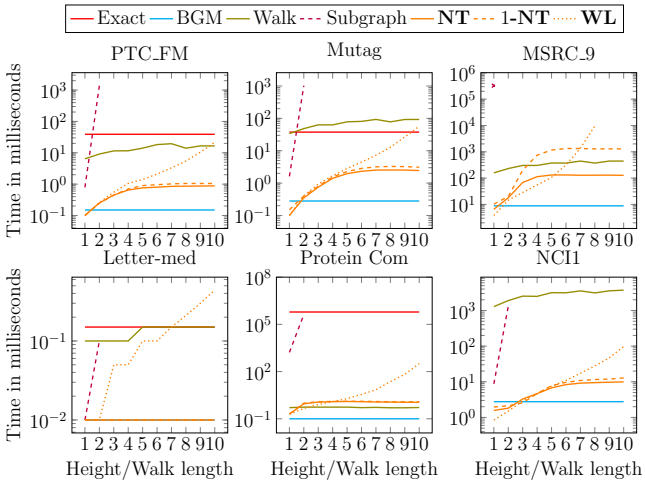


Fig. 8: Average runtime in milliseconds for computing the distance of two graphs of newly proposed (bold) and state-of-the art method.

E. Runtime and Approximation Error with 1-NTs

Figure 8 shows the runtime for the different methods, also including 1-NTs. Interestingly, 1-*NT* performs much worse on dataset *MSRC_9* than *NT*, while on the other datasets, there was not much of a difference between them. This could be due to the higher average vertex degree in this dataset, since 1-*NT* has some redundancy in the tree (by allowing vertices to appear in two consecutive levels).

Figure 9 shows the average approximation error of the different methods, also including 1-NTs. It can be seen, that they are only slightly better than 0-NTs in some cases. This, and the lower runtime, suggests that 0-NTs are sufficient for representing the neighborhood of a vertex accurately and efficiently.

F. Caching

In our extended experiments we evaluate the speed-up in runtime gained using our caching strategy for the different tree variants. For a fair comparison, we do not store the cached values for the whole dataset, but only during the pairwise approximation. If the values were kept over the computation of all or multiple approximations additional speedup at the cost of memory may be gained whenever subtree pairs reoccur for different graph pairs.

Figure 10 shows the average runtime for computing our approximations of the graph edit distance with and without using our caching strategy and varying tree height. Since the runtime for the uncached *WL* increased very quickly with larger height, only values up to a tree height of 5 are given. On almost all datasets we can see a huge runtime benefit for all methods. Because the size of the *WL* trees is not limited by the graph size and quickly grows, this method benefits the most from caching. The other two methods benefit more on datasets with higher average degree and larger diameter. One reason for this, might be the higher number of redundant

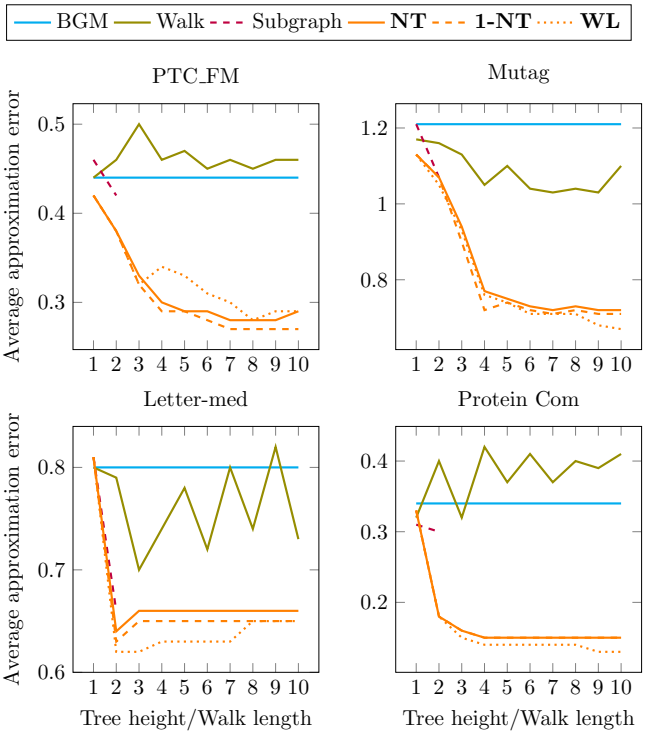


Fig. 9: Average relative error (proposed in bold).

computations occurring in these graphs, which can be avoided by our caching strategy.

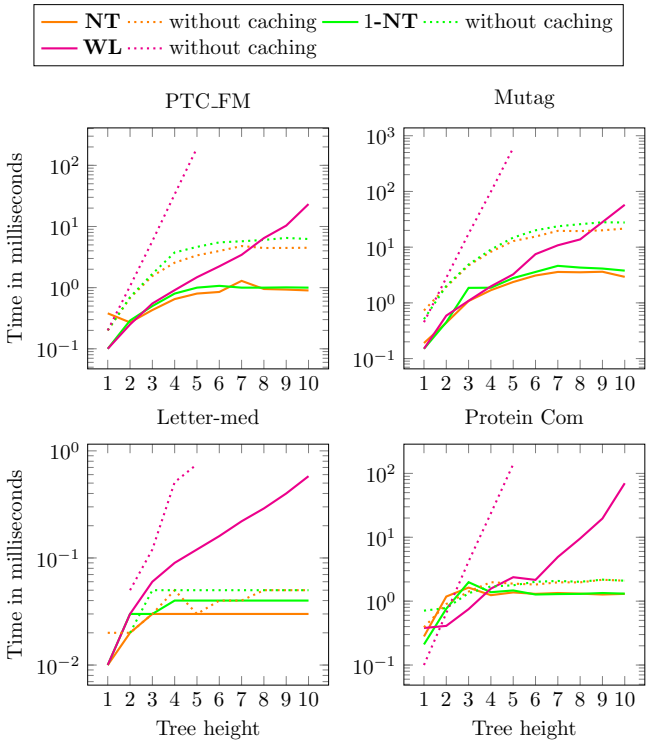


Fig. 10: Average runtime in milliseconds for computing the distance of two graphs using our methods with and without our caching.

4.3 Conclusion

In this chapter, a novel approach for approximating Graph Edit Distance with the help of new neighborhood trees was presented. Neighborhood trees abstract the local neighborhood of a vertex by storing all shortest paths to other reachable vertices, with their intermediate vertices and edges, up to a maximum length. Thereby, the depth of each tree is limited by the distance to the furthest away vertex and the overall diameter of the graph. This representation is a lot smaller than Weisfeiler Lehman trees and can even be further reduced in size by compressing the neighborhood trees into directed acyclic graphs (compressed neighborhood trees), resulting in trees that at most contain as many vertices and edges as the original graph.

To approximate GED, the same methodology as in classical Bipartite Graph Matching is applied, whereby the representations of the vertices of both graphs are compared pairwise to create a cost matrix from which a minimal assignment is found. To obtain the cost values for this assignment, Structure and Depth preserving Tree Edit Distance (SDTED) is used with the compressed neighborhood trees of various sizes. With the optimal assignment of the trees, the edit path and the resulting GED can be derived. The result section shows that this newly created method is able to outperform BGM in accuracy, while mostly staying within the same order of magnitude regarding runtime.

The concept of neighborhood trees and the proposed compression should also be applicable in other graph-related domains. For example, many graph learning methods use Weisfeiler Lehman (WL) trees and kernels, where optimizations regarding runtime may be possible using the novel neighborhood tree representation. Another advantage of these trees is that they encode less redundant information than WL trees, which may allow machine learning methods to more easily derive relevant information from the data and thereby produce more accurate models.

Chapter 5

Conclusion

In this dissertation, important topics of computer aided drug design and chemoinformatics were touched on with a focus on domains that need special attention due to the rising importance of novel macrocyclic compounds. In each of those domains, novel methods and algorithms were presented that are able to improve upon the state of the art and thereby facilitate further research on large compounds and compounds with macrocyclic core.

A deeper look into virtual molecule screening methods revealed that current state of the art algorithms often miss molecules even though they should fit the presented screening model. This is the case due to the overemphasis on secondary fitness scores, which are not in line with the primary goal of finding a transformation. Therefore, a novel alignment algorithm (G3PS) was proposed that avoids these issues by iteratively searching for valid transformations, without overfitting towards such secondary scores. Also, other additional model restrictions, such as exclusion volumes, are considered already during the process of finding the transformation, instead of only considering them in post-processing as a filter. The results show that this new approach is able to retrieve molecules with pharmacophore screening both more accurately and with lower runtime than the previous state of the art. When the model allows for omitted features (only partial matches of the molecules) the runtime improvement of G3PS grows exponentially in the number of allowed omitted features, as its complexity does not grow with omitted features. The increase in accuracy and runtime should prove especially useful when working with macrocycles, as those are especially difficult to find fitting transformations for and require a lot of processing time due to their size.

Conformer generation was another topic of interest, as there are no algorithms publicly available that can compete with commercial ones regarding both runtime and accuracy. The generation of high-quality conformers for macrocyclic structures thereby proved especially challenging, as even only commercially available generators often struggle with such structures. For this reason, an openly available novel conformer generation algorithm (CONFORGE) has been developed that implements various optimizations such as refined fragment libraries and caching strategies to provide an attractive alternative to the scientific community. The performed benchmarks show that CONFORGE is able to produce conformer ensembles both faster and with higher accuracy than other openly available methods while also outperforming the tested commercial ones. For the generation of conformers for compounds with macrocycles, a stochastic sampling strategy is employed with which the produced ensembles outperform all of the tested competitors regarding accuracy. This new method should drastically improve the state of the art and facilitate future research when working with macrocycles.

Graph edit distance (GED) and approximation thereof were also explored. As the com-

putation of exact GED is computationally infeasible for even only medium-sized graphs due to its high algorithmic complexity, it needs to be approximated in practice. Still, GED is of great interest when working with molecules, as there exist various methods depending on it, and becomes even more difficult to use with large compounds or compounds with macrocycles. Therefore, an investigation on how such an approximation can be obtained more accurately while not requiring unreasonable amounts of time to compute was necessary. As a result, a novel local neighborhood representation in the form of (compressed) neighborhood trees was designed which can be compared and used for creating a vertex assignment. With this assignment, a suboptimal edit path and thereby upper bound on the GED can be computed. The results show that this new method is able to approximate GED with this upper bound more accurately than fast state of the art methods while staying within the same order of magnitude regarding runtime, which other methods don't. The proposed neighborhood tree concept could also be applied in other domains. It should be able to replace Weisfeiler Lehman trees in some applications such as graph neural networks, where both accuracy and performance could be gained in the future.

Overall, proposed methods and algorithms presented in this dissertation should facilitate future research in the domain of macrocycles and computer aided drug discovery in general. For all of the touched topics, the newly developed methods bring significant improvements regarding result quality, computation time, or in many cases even both. The straightforward methodologies and explainable nature of the methods should also provide an attractive starting point for further research and additional improvement upon this thesis' work.

Appendices

Appendix A

Abstract

A.1 English abstract

With the recently fast-growing interest in macrocyclic compounds and the novel synthesis methods which enable their creation, new computer aided drug design methods are needed to improve the drug design process. For this reason, existing methods need to be reevaluated and improved upon to handle both larger and more complex molecules. In this dissertation, the current state of the art in three application domains, namely virtual screening, small molecule conformer generation, and graph edit distance approximation, were investigated.

For virtual screening, fundamental flaws with existing methods were identified that lead to degraded performance and accuracy in hit retrieval. With a major refocus of the optimization goal a novel alignment algorithm, Greedy 3-Point Search, was developed, which is faster and more accurate in retrieving hits than previous methods. With this new method, the accuracy and therefore scientific value of pharmacophore-based virtual screening should increase significantly.

Conformer generation methods are widely available, though the openly accessible methods produce worse results than commercial-only ones while being computationally less efficient. The newly created CONFORGE method aims to solve this issue by improving upon the performance and accuracy of publicly available methods, bringing them closer to or even surpassing commercially-only ones. It is available for anyone to use with the CDPKit, which can be downloaded from the public GitHub repository [40].

Graph Edit Distance (GED) computation is NP-hard, making it unfeasible to use with larger-sized graphs/molecules. While various approximate methods exist, increasing their accuracy would be beneficial for the use with many follow-up methods. For this reason, a novel representation of local node surroundings with neighborhood trees was developed, which can be used in conjunction with the Bipartite Graph Matching (BGM) methodology. This enables more accurate GED estimation while staying in the same order of magnitude as standard BGM regarding runtime. Other methods, which provide better accuracy than BGM, take multiple orders of magnitude longer to run. This new method, thereby, fills an accuracy/runtime trade-off spot, that was previously not achievable.

With these novel methods, future macrocycle and chemoinformatics research in general will be able to be conducted more accurately and efficiently. The newly developed concepts also open up the possibility of new research directions and to spawn further ideas to advance chemoinformatics or even completely different research domains.

A.2 Deutsche Zusammenfassung

Mit dem in letzter Zeit schnell wachsenden Interesse an makrocyclischen Verbindungen und den neuartigen Syntheseverfahren, die ihre Herstellung ermöglichen, werden neue computergestützte Arzneimitteldesignverfahren benötigt, um den Designprozess zu verbessern. Aus diesem Grund müssen bestehende Methoden neu bewertet und verbessert werden, um sowohl größere als auch komplexere Moleküle handhaben zu können. In dieser Dissertation wurde der aktuelle Stand der Technik in drei Anwendungsdomänen untersucht, nämlich virtuelles Screening, Generierung von Konformeren für kleine Moleküle und Approximation von Graph Edit Distance.

Für das virtuelle Screening wurden grundlegende Mängel mit bestehenden Methoden identifiziert, die sowohl zu mangelhafter Laufzeit-Leistung als auch zu geringer Genauigkeit beim auffinden von Treffern führen. Mit einer fundamentalen Neuausrichtung des Optimierungsziels wurde ein neuartiger Ausrichtungsalgorithmus entwickelt, Greedy 3-Point Search, der beim Auffinden von Treffern sowohl schneller als auch genauer ist als frühere Methoden. Die durch den Einsatz dieser neuen Methode gewonnene Genauigkeit sollte den wissenschaftlichen Wert des pharmakophorbasierten virtuellen Screenings nochmals erheblich steigern.

Methoden zur Generierung von Konformeren sind weit verbreitet, wobei die offen zugänglichen Methoden schlechtere Ergebnisse liefern als kommerzielle Methoden, während sie rechnerisch weniger effizient sind. Die neu entwickelte CONFORGE-Methode zielt darauf ab, dieses Problem zu lösen, indem sie sowohl die Leistung als auch die Genauigkeit öffentlich verfügbarer Methoden verbessert und damit näher an kommerzielle Methoden herankommt oder sie sogar übertrifft. Die entwickelte Software ist für jeden über das CDPKit verfügbar, welches aus dem öffentlichen GitHub-Repository [40] heruntergeladen werden kann.

Die Berechnung der Graph Edit Distance (GED) ist NP-hart, was eine Verwendung für größere Graphen/Moleküle unmöglich macht. Es gibt verschiedene Näherungsverfahren, dennoch wäre eine Erhöhung ihrer Genauigkeit für die Verwendung mit vielen Folgeverfahren von Vorteil. Aus diesem Grund wurde eine neuartige Darstellung der lokalen Knotenumgebung mit Nachbarschaftsbäumen entwickelt, die in Verbindung mit der Bipartite Graph Matching (BGM) Methodik verwendet werden kann. Dies ermöglicht eine genauere GED-Schätzung, während sie in Bezug auf die Laufzeit in der gleichen Größenordnung wie Standard-BGM bleibt. Andere Methoden, die eine bessere Genauigkeit als BGM bieten, benötigen mehrere Größenordnungen mehr Zeit. Das neue Verfahren füllt damit einen Genauigkeits-/Laufzeit-Kompromisspunkt aus, der zuvor nicht erreichbar war.

Mit den vorgestellten neuartigen Methoden ist zu erwarten, dass zukünftige Makrocyclen- und Chemoinformatik-Forschung im Allgemeinen genauer und effizienter durchgeführt werden kann. Die neu entwickelten Konzepte bieten sich auch dafür an neue Forschungsrichtungen zu eröffnen und weitere Ideen hervorzubringen, die Chemoinformatik oder auch andere Forschungsgebiete weiter voranbringen werden.

Bibliography

[1] Acharya, C., Coop, A., Polli, J. E., and Mackerell, Jr, A. D. Recent advances in ligand-based drug design: relevance and utility of the conformationally sampled pharmacophore approach. *Curr Comput Aided Drug Des 7*, 1 (Mar. 2011), 10–22.

[2] Aksu, B., and Yeğen, G. Chapter 2 - global regulatory perspectives on quality by design in pharma manufacturing. In *Pharmaceutical Quality by Design*, S. Beg and M. S. Hasnain, Eds. Academic Press, 2019, pp. 19–41.

[3] Anderson, A. C. The process of structure-based drug design. *Chemistry & Biology 10*, 9 (2003), 787–797.

[4] Aparoy, P., Reddy, K. K., and Reddanna, P. Structure and ligand based drug design strategies in the development of novel 5- LOX inhibitors. *Curr Med Chem 19*, 22 (2012), 3763–3778.

[5] Blumenthal, D. B., Boria, N., Gamper, J., Bougleux, S., and Brun, L. Comparing heuristics for graph edit distance computation. *The VLDB Journal 29*, 1 (Jan 2020), 419–458.

[6] Carletti, V., Gaüzère, B., Brun, L., and Vento, M. Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance. In *Graph-Based Representations in Pattern Recognition* (2015), pp. 188–197.

[7] Durrant, J. D., and McCammon, J. A. Molecular dynamics simulations and drug discovery. *BMC Biology 9*, 1 (Oct. 2011), 71.

[8] Ekins, S., Mestres, J., and Testa, B. In silico pharmacology for drug discovery: methods for virtual ligand screening and profiling. *Br J Pharmacol 152*, 1 (June 2007), 9–20.

[9] Fischer, A., Smieško, M., Sellner, M., and Lill, M. A. Decision making in structure-based drug discovery: Visual inspection of docking results. *Journal of Medicinal Chemistry 64*, 5 (2021), 2489–2500. PMID: 33617246.

[10] Friedrich, N.-O., de Bruyn Kops, C., Flachsenberg, F., Sommer, K., Rarey, M., and Kirchmair, J. Benchmarking commercial conformer ensemble generators. *Journal of Chemical Information and Modeling 57*, 11 (2017), 2719–2728. PMID: 28967749.

[11] Friedrich, N.-O., Meyder, A., de Bruyn Kops, C., Sommer, K., Flachsenberg, F., Rarey, M., and Kirchmair, J. High-quality dataset of protein-bound

ligand conformations and its application to benchmarking conformer ensemble generators. *Journal of Chemical Information and Modeling 57*, 3 (2017), 529–539. PMID: 28206754.

[12] Gao, X., Xiao, B., Tao, D., and Li, X. A survey of graph edit distance. *Pattern Anal. Appl. 13* (02 2010), 113–129.

[13] Garcia-Hernandez, C., Fernández, A., and Serratosa, F. Ligand-based virtual screening using graph edit distance as molecular similarity measure. *Journal of Chemical Information and Modeling 59*, 4 (2019), 1410–1421. PMID: 30920214.

[14] Gaüzère, B., Bougleux, S., Riesen, K., and Brun, L. Approximate graph edit distance guided by bipartite matching of bags of walks. pp. 73–82.

[15] Gini, G. QSAR methods. *Methods Mol Biol 2425* (2022), 1–26.

[16] Good, A. 4.19 - virtual screening. In *Comprehensive Medicinal Chemistry II*, J. B. Taylor and D. J. Triggle, Eds. Elsevier, Oxford, 2007, pp. 459–494.

[17] Hawkins, P. C. D. Conformation generation: The state of the art. *Journal of Chemical Information and Modeling 57*, 8 (2017), 1747–1756. PMID: 28682617.

[18] Horvath, D. Pharmacophore-based virtual screening. *Methods Mol Biol 672* (2011), 261–298.

[19] Huang, N. T., and Villar, S. A short tutorial on the weisfeiler-lehman test and its variants. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (jun 2021), IEEE.

[20] Joshi, T., Sharma, P., Joshi, T., Mathpal, S., Pandey, S. C., Pandey, A., and Chandra, S. Chapter 4 - recent advances on computational approach towards potential drug discovery against leishmaniasis. In *Pathogenesis, Treatment and Prevention of Leishmaniasis*, M. Samant and S. Chandra Pandey, Eds. Academic Press, 2021, pp. 63–84.

[21] Karplus, M., and McCammon, J. A. Molecular dynamics simulations of biomolecules. *Nature Structural Biology 9*, 9 (Sep 2002), 646–652.

[22] Kwitkowski, V. E., Prowell, T. M., Ibrahim, A., Farrell, A. T., Justice, R., Mitchell, S. S., Sridhara, R., and Pazdur, R. Fda approval summary: Temsirolimus as treatment for advanced renal cell carcinoma. *The Oncologist 15*, 4 (2010), 428–435.

[23] Langer, T., and Hoffmann, R. D. *Pharmacophores and Pharmacophore Searches*. John Wiley & Sons, Dec. 2006.

[24] Langer, T., and Wolber, G. Pharmacophore definition and 3D searches. *Drug Discov. Today Technol. 1*, 3 (Dec. 2004), 203–207.

[25] Leach, A. R., Gillet, V. J., Lewis, R. A., and Taylor, R. Three-dimensional pharmacophore methods in drug discovery. *J. Med. Chem. 53*, 2 (Jan. 2010), 539–558.

[26] Li, T., Dong, H., Shi, Y., and Dehmer, M. A comparative analysis of new graph distance measures and graph edit distance. *Information Sciences 403-404* (2017), 15–21.

[27] Mallinson, J., and Collins, I. Macrocycles in new drug discovery. *Future Medicinal Chemistry 4*, 11 (2012), 1409–1438. PMID: 22857532.

[28] Marsault, E., and Peterson, M. Macrocycles are great cycles: Applications, opportunities, and challenges of synthetic macrocycles in drug discovery. *Journal of medicinal chemistry 54* (03 2011), 1961–2004.

[29] Meng, X.-Y., Zhang, H.-X., Mezei, M., and Cui, M. Molecular docking: a powerful approach for structure-based drug discovery. *Curr Comput Aided Drug Des 7*, 2 (June 2011), 146–157.

[30] Morris, C., Fey, M., and Kriege, N. M. The power of the weisfeiler-leman algorithm for machine learning with graphs. In *IJCAI 2021* (2021), pp. 4543–4550.

[31] Morris, G. M., and Lim-Wilby, M. Molecular docking. *Methods Mol Biol 443* (2008), 365–382.

[32] National Center for Biotechnology Information. Pubchem: Compound summary of temsirolimus. `https://pubchem.ncbi.nlm.nih.gov/compound/Temsirolimus`. Accessed: 2023-02-07.

[33] Newman, D. J., and Cragg, G. M. Chapter 1 bioactive macrocycles from nature. In *Macrocycles in Drug Discovery*. The Royal Society of Chemistry, 2015, pp. 1–36.

[34] Pirhadi, S., Shiri, F., and Ghasemi, J. B. Methods and applications of structure based pharmacophores in drug discovery. *Curr Top Med Chem 13*, 9 (2013), 1036–1047.

[35] Poli, G., Seidel, T., and Langer, T. Conformational sampling of small molecules with icon: Performance assessment in comparison with omega. *Frontiers in Chemistry 6* (2018), 229.

[36] Qing, X.-Y., Lee, X. Y., De Raeymaecker, J., Tame, J., Zhang, K., De Maeyer, M., and Voet, A. Pharmacophore modeling: Advances, limitations, and current utility in drug discovery. *Journal of Receptor, Ligand and Channel Research 7* (11 2014), 81–92.

[37] Rahman, M., Santra, S., Kovalev, I., Kopchuk, D., Zyryanov, G., Majee, A., and Chupakhin, O. Green synthetic approaches for practically relevant (hetero)macrocycles: An overview.

[38] Riesen, K., and Bunke, H. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput. 27*, 7 (2009), 950–959.

[39] Roy, K., and Das, R. N. A review on principles, theory and practices of 2D-QSAR. *Curr Drug Metab 15*, 4 (2014), 346–379.

[40] Seidel, T. Chemical data processing toolkit. `https://github.com/molinfo-vienna/CDPKit`. Accessed: 2023-02-07.

[41] Seidel, T., Ibis, G., Bendix, F., and Wolber, G. Strategies for 3d pharmacophore-based virtual screening. *Drug Discovery Today: Technologies 7*, 4 (2010), e221–e228. 3D Pharmacophore Elucidation and Virtual Screening.

[42] Sengupta, S., and Mehta, G. Macrocyclization via c–h functionalization: a new paradigm in macrocycle synthesis. *Org. Biomol. Chem. 18* (2020), 1851–1876.

[43] Simoens, S., and Huys, I. R&d costs of new medicines: A landscape analysis. *Frontiers in Medicine 8* (2021).

[44] Thurakkal, L., Nanjan, P., and Porel, M. Design, synthesis and bioactive properties of a class of macrocycles with tunable functional groups and ring size. *Scientific Reports 12*, 1 (Mar 2022), 4815.

[45] Tomar, V., Mazumder, M., Chandra, R., Yang, J., and Sakharkar, M. K. Small molecule drug design. In *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, Eds. Academic Press, Oxford, 2019, pp. 741–760.

[46] Verma, J., Khedkar, V. M., and Coutinho, E. C. 3D-QSAR in drug design–a review. *Curr Top Med Chem 10*, 1 (2010), 95–115.

[47] Wolber, G., and Langer, T. LigandScout: 3-D pharmacophores derived from protein-bound ligands and their use as virtual screening filters. *J Chem Inf Model 45*, 1 (Jan. 2005), 160–169.

[48] Xu, M., Luo, S., Bengio, Y., Peng, J., and Tang, J. Learning neural generative dynamics for molecular conformation generation, 2021.

[49] Yu, W., and MacKerell, Jr, A. D. Computer-Aided drug design methods. *Methods Mol Biol 1520* (2017), 85–106.

[50] Yudin, A. K. Macrocycles: lessons from the distant past, recent developments, and future directions. *Chem Sci 6*, 1 (Nov. 2014), 30–49.