universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Graph algorithms for alchemical transformations using the free energy package Transformato“

verfasst von / submitted by

Josef Hackl, MA BA BA BSc BSc

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

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066875
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Bioinformatik
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Stefan Boresch
Mitbetreut von / Co-Supervisor:	Dr. Marcus Wieder, MSc MSc

Abstract

Alchemical free energy calculations estimate free energies by using unphysical intermediates. In addition to the computation of absolute solvation and binding free energy differences, this method can be used to compute relative free differences, for instance, the energy difference of binding between two ligands. Usually, the number of atoms between the two end states, i.e., the two molecules of interest, is not the same. However, this is a necessary condition for the molecular dynamics simulations on which the computation of the free energy differences is based. To preserve the number of atoms so-called dummy atoms which act as placeholders have to be introduced. In this Master Thesis, new features for Transformato, a package which helps to set up relative alchemical free energy calculations, werde developed. In particular, additional functions for the employment of these dummy atoms were implemented. Transformato uses a common core scaffold which contains the atoms present in both molecules. Both initial states of the alchemical transformations initialized by Transformato do not contain any dummy atoms, but consist solely of the physical atoms of the respective molecules. However, dummy atoms are generated via two separate alchemical paths leading to the common core. Starting from the initial states, physical atoms are successively turned into dummy atoms until the common core structure is attained. The generation of the common core was optimized and the processing of hydrogens adjusted. Graph traversal algorithms for the determination of appropriate mutation routes were applied so that a flawless Transformato workflow without manual postprocessing is ensured. Finally, the effect of different common core generation and mutation algorithms on the results of free energy calculations was investigated.

Kurzfassung

Alchemische Freie-Energie-Berechnungen berechnen freie Energien mithilfe unphysikalischer Zwischenzustände. Zusätzlich zur Berechnung absoluter Differenzen in der freien Solvatations- bzw. Bindungsenergie kann diese Methode auch zur Berechnung relativer Freier-Energie-Differenzen verwendet werden, beispielsweise für die Berechnung der Bindungsenergiedifferenz zwischen zwei Liganden. Normalerweise ist die Anzahl der Atome zwischen den beiden Endzuständen, also den beiden interessierenden Molekülen, jedoch nicht gleich. Dies ist aber eine notwendige Voraussetzung für die Molekulardynamiksimulationen, auf denen die Berechnung der Freie-Energie-Differenzen basiert. Um die Anzahl der Atome beizubehalten, müssen Dummy-Atome, welche als Platzhalter fungieren, eingesetzt werden. Im Zuge dieser Masterarbeit wurden neue Funktionen für Transformato, ein Softwarepaket zum Aufsetzen relativer alchemischer Freie-Energie-Berechnungen, entwickelt. Insbesondere wurden zusätzliche Funktionen für den Einsatz der erwähnten Dummy-Atome implementiert. Transformato verwendet einen den beiden Molekülen ‚gemeinsamen Kern' (Common Core), der die in beiden Molekülen vorhandenen Atome enthält. Beide Ausgangszustände der durch Transformato initialisierten alchemischen Transformationen enthalten keine Dummy-Atome, sondern bestehen ausschließlich aus den physikalischen Atomen der jeweiligen Moleküle. Dummy-Atome werden jedoch über zwei separate alchemische Pfade erzeugt, die zum Common Core führen. Ausgehend von den Ausgangszuständen werden physikalische Atome sukzessive in Dummy-Atome umgewandelt, bis die Struktur des Common Cores erreicht ist. Insbesondere wurde die Erzeugung des Common Cores optimiert und die darin involvierte Verarbeitung von Wasserstoffatomen verbessert. Es wurden Graphalgorithmen zur Bestimmung geeigneter Mutationsrouten eingesetzt, sodass ein einwandfreier Transformato-Workflow ohne manuelle Nachbearbeitung der Moleküle gewährleistet ist. Abschließend wurde der Einfluss verschiedener Common-Core-Erzeugungs- und Mutationsalgorithmen auf die Ergebnisse von Freien-Energie-Berechnungen untersucht.

Contents

1 Introduction

The aim of this Master Thesis is to facilitate the preparation of alchemical free energy calculations. Such calculations estimate free energy differences by using nonphysical intermediates, i.e., structures which are not found in nature as existing chemical species. In addition to the computation of absolute solvation and binding free energy differences, the method can be used to compute relative free energy differences, e.g., the free energy difference of binding between two ligands. A problem occurring in the latter approach is the need for so-called 'dummy atoms'. Usually, the number of atoms between the two end states, i.e., the two molecules of interest, is not the same. However, this is a necessary condition for the molecular dynamics simulations on which the computation of the free energy differences is based. To preserve the number of atoms, these dummy atoms act as placeholders [1, 2].

This Master Thesis is concerned with specific methods of handling these nonphysical atoms. A central part is the implementation of new features for `transformato`, a package which helps to set up relative alchemical free energy calculations using an innovative common core (CC) approach [3, 4]. In particular, helper functions are developed which optimize the employment of the aforementioned dummy atoms. The implemented functions are collected in the Python package tf-routes available on GitHub (https://github.com/jalhackl/tf_routes/tree/master/tf_routes).

In the following chapter, the basic principles of alchemical free energy calculations are explained. The third chapter presents the workflow of `transformato`. Subsequently, the need for improving and extending some algorithms of the software package is described in more detail. Examples of alchemical mutations proposed by the new algorithms and corresponding CC constructions for `transformato` are given. Finally, the effect of different CC generation and mutation algorithms on the results of free energy calculations is discussed.

2 Free Energy Calculations

2.1 Basics

In the last few years, the accuracy and feasibility of free energy calculations improved significantly [5]. The main reasons are developments in the accuracy of force fields [6], the increase in computational resources and, in particular, the usage of graphics processing units to cope with the high computational demands. By now, most MD software packages, like AMBER [7], CHARMM [8] or GROMACS [9], offer functions for alchemical free energy calculations which facilitates the set-up of such simulations.

Possible applications can be found in rational drug design and drug discovery; e.g., during lead optimization, the binding free energy differences between compounds are of interest. [6] As the free energy difference provides information about the thermodynamic favorability of a specific process, it can help to find ligands that bind most tightly to a biomolecule of interest.

One can distinguish between absolute and relative free energy calculations: Absolute free energy differences are, for instance, solvation or binding free energy differences of one compound (these results can be compared with the free energy of an unrelated compound) [10, 11], whereas the latter approach computes the free energy difference between, e.g., two ligands, which usually are related to each other. For many practical problems, such knowledge is sufficient, for instance, when the comparison of properties like binding affinity of two ligands is sought. The relative free energy differences between two ligands can provide information to predict protein-ligand binding affinities and to select specific ligands, drugs etc. for optimizing binding affinity. Knowledge of binding affinities can be harnessed for tasks like protein engineering [5].

Relative free energy calculations harness the concept of a thermodynamic cycle [12]; see Fig. 2.1: The horizontal arrows indicate the paths from the unbound to the bound state of each of the two ligands, the vertical arrows indicate the transformation from one molecule to the other one. According to the 2nd law of thermodynamics, the free energy differences along both paths in the figure leading from the unbound state of ligand A to the bound state of ligand B must be identical. In other words, we have

$$\Delta G_A + \Delta G_2 = \Delta G_1 + \Delta G_B. \tag{2.1}$$

From this one sees that the relative binding free energy difference between the two ligands can be expressed as [6]:

$$\Delta\Delta G = \Delta G_2 - \Delta G_1 = \Delta G_B - \Delta G_A. \tag{2.2}$$

To obtain knowledge about $\Delta\Delta G = \Delta G_2 - \Delta G_1$, the evaluation of the alchemical transformations ΔG_A and ΔG_B suffices. In practice, the determination of ΔG_2 or ΔG_1 is

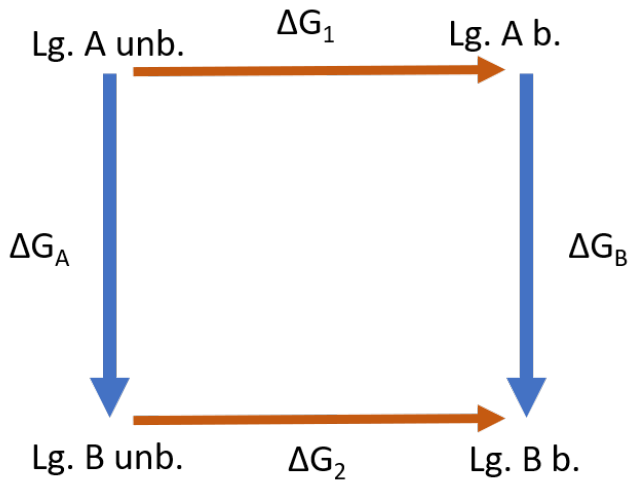


Figure 2.1: Thermodynamic cycle; red arrows indicate transitions between two states (unbound–bound) of each ligand, blue arrows indicate 'alchemical' transformations (Ligand A–Ligand B)

usually much more computationally expensive because the states are vastly different (e.g., in the case of ligand binding, water environment vs. protein, and under the assumption that the ligands exhibit a similar structure) and often requires an experimental set-up; thus, the alchemical calculation can substitute this step or at least indicate if, e.g., a certain ligand is a promising candidate.

The vertical part of the depicted thermodynamic cycle is easier to compute because the change between both states is much smaller (depending on the molecules of interest, only some atoms change and hence there are fewer annihilation or creation steps) and, thus, in general, fewer intermediate steps are necessary; however it involves 'alchemical' transformations, i.e., nonphysical intermediates have to be used.

In general, the free energy is given by $F = -k_B T \ln Q$, where Q denotes the partition function $Q = \int dr \exp(-\beta U)$ with $\beta = \frac{1}{k_B T}$. Hence, the free energy difference between states i and j can be described as [13]:

$$\triangle F_{ij} = -\frac{1}{\beta} \ln \frac{Z_j}{Z_i}. \tag{2.3}$$

To compute the free energy differences between two states, various methods exist. Usually, it is not possible to simply compute the difference between the two end states; hence, intermediate states have to be taken into account. The difference between the final states can be expressed as the sum of the difference between these intermediate states: $\Delta F = F_1 - F_0 = \sum_n \Delta F_n$ [14].

2.2 Methods for evaluating free energy differences

Various approaches for calculating free energy differences exist, e.g., thermodynamic integration and perturbation. Implementations of the latter approach use the Zwanzig formula or Bennett's acceptance ratio method (which is used in the `transformato` package described below). In the following, these three approaches will be briefly outlined. (For a more comprehensive comparison and estimation of performance differences, see [15] and [16]).

2.2.1 Thermodynamic Integration (TI)

In Thermodynamic Integration [17], the free energy difference between two states is computed by evaluating the integral over the derivative of the free energy between the initial and the final state of the transformation/process studied. TI computes intermediate states depending on the coupling parameter λ. $\lambda = 0$ and $\lambda = 1$ represent the physical, initial / final states of the system. Scaling between those two values, i.e., using values $0 \leq \lambda \leq 1$, gives rise to the nonphysical, 'alchemical' intermediate states.

Taking the derivative of the free energy, one gets $\frac{dF}{d\lambda} = \frac{d}{d\lambda} \int e^{-\beta U} dr = \left\langle \frac{dU(r,\lambda)}{d\lambda} \right\rangle_\lambda$ [13], where the angular brackets indicate the ensemble average. The free energy difference is then given as the integral from $\lambda = 0$ to $\lambda = 1$:

$$\Delta F = \int_{\lambda=0}^{\lambda=1} \frac{dF(\lambda)}{d\lambda} d\lambda = \int_{\lambda=0}^{\lambda=1} \left\langle \frac{\partial U(\lambda)}{\partial \lambda} \right\rangle_\lambda d\lambda. \tag{2.4}$$

This integral has to be approximated using numerical integration, i.e., $\Delta F \approx \sum_i w_i \left\langle \frac{dU(r,\lambda)}{d\lambda} \right\rangle_{\lambda_i}$. The weights w_i depend on the choice of the numerical quadrature scheme. A popular and simple choice is the trapezoidal rule, which in its most basic form uses equal spacing between states and approximates the integral by $\int_{\lambda=j}^{\lambda=i} \frac{dF(\lambda)}{d\lambda} d\lambda \approx \frac{i-j}{2} (f_i + f_j)$. More efficient schemes can lower the number of necessary λ-states, e.g., Simpson's rule, Gauss-Legendre Quadrature, cubic spline interpolation or Clenshaw-Curtis integration. Furthermore, usage of non-equidistant spacing, which can be easily be applied for the trapezoidal rule, but also for Simpson's rule, can improve efficiency. For alchemical free energy simulations, it seems advisable to include the endpoints since those are the only physical states of the simulations (e.g., in Gauss-Legendre quadrature the endpoints are not included). [15, 18]

In any case, the integration scheme and the amount of intermediate steps has to be chosen in such a way that the introduced bias is below the statistical noise [13]. (For a more detailed comparison of various numerical quadrature schemes, e.g., the trapezoidal rule and Simpson's rule, see [18]).

2.2.2 Free Energy Perturbation / Zwanzig Relation

Free energy perturbation relies on the Zwanzig relation (sometimes called exponential formula). For each configuration of state A, the energy difference between this state and

the corresponding state B is calculated, which is then used to compute the free energy difference between A and B according to [19, 20]

$$\Delta F_{A\to B} = F_B - F_A = -\frac{1}{\beta}\ln\frac{Q_B}{Q_A} = -k_b T \ln\left\langle \exp\left(\frac{-\Delta U_{A\to B}}{k_B T}\right)\right\rangle_A . \tag{2.5}$$

Several variants of the formula exist: For instance, one can either calculate forward or backward perturbations (either $\Delta F\left(A\to B\right)$ or $-\Delta F\left(B\to A\right)$), or, as it is usually the case, use double-wide sampling where energy differences from both directions are processed [15].

As for thermodynamic integration, usually it is necessary to introduce intermediate states. The free energy between the final states 0 and 1 is calculated as the sum of the differences between all adjacent intermediate states; in the case of n states: $\Delta F_{0\to 1} = \sum_{i=0}^{n-1}\left(F\left(i+1\right) - F\left(i\right)\right)$. There has to be a sufficient number of intermediate states — there must be significant overlap between the two states — otherwise convergence is poor. However, there are still use cases when other methods are not feasible [21], and there exists an extension to non-equilibrium work (Jarzynski's equation) [21]. In general, however, it should be only used if the difference between the two states are tiny [13].

2.2.3 Bennett Acceptance Ratio (BAR)

An extension of the perturbation approach for computing the free energy difference between two states is Bennett's Acceptance Ratio [22].

Expansion of the denominator and numerator of the ratio of the canonical partition functions of both states leads to:

$$\frac{Q_0}{Q_1} = \frac{Q_0\int W\exp\left(-U_0 - U_1\right)dq^N}{Q_1\int W\exp\left(-U_0 - U_1\right)dq^N} = \frac{\langle W\exp\left(-U_0\right)\rangle_1}{\langle W\exp\left(-U_1\right)\rangle_0} \tag{2.6}$$

where W denotes a (for now arbitrary) weighting function. Assuming a Gaussian distribution of the estimation error in the limit of large samples, the optimal W can be determined as $W\left(q_1 ... q_n\right) = c\left(\frac{Q_0}{n_0}\exp\left(-U_1\right) + \frac{Q_1}{n_1}\exp\left(-U_0\right)\right)^{-1}$ [22].

To use Bennett's acceptance ratio method, two simulations have to be carried out. One starts at $\lambda = 0$, the other one at $\lambda = 1$. Forward and backward simulations are processed simultaneously.

The free energy difference between two λ-states can be expressed as

$$\triangle F\left(\lambda_i \to \lambda_j\right) = \beta^{-1}\left(\ln\frac{\left\langle f\left(U_{\lambda_i} - U_{\lambda_j} + C\right)\right\rangle_{\lambda_j}}{\left\langle f\left(U_{\lambda_j} - U_{\lambda_i} + C\right)\right\rangle_{\lambda_i}}\right) + C \tag{2.7}$$

with the Fermi function $f\left(x\right) = \frac{1}{1+\exp(\beta x)}$ [15, 20].

To obtain the value of the optimum shift constant,

$$C = \beta^{-1}\ln\frac{Q_i N_j}{Q_j N_i}, \tag{2.8}$$

a self-consistency problem has to be solved iteratively [20].

The free energy between two λ-states is then given by $\triangle F\left(\lambda_i \rightarrow \lambda_j\right) = \beta^{-1} \ln \frac{N_j}{N_i} + C$ [15]. N_j and $N_i$denote the number of sampled configurations from state j and i, respectively.

BAR gives a minimal variance free energy estimate. There is also an alternative derivation for the formula. It can be shown that the Bennett acceptance ratio method works as a maximum likelihood estimator. Using BAR, one obtains the asymptotically unbiased estimation (i.e., for an infinite number of measurements it yields an unbiased estimation) with the lowest variance [23].

Using variational calculus to minimize the variance, ΔF can be expressed implicitly as: $\frac{\partial \ln L(\Delta F)}{\partial \Delta F} = \sum \frac{1}{1+\exp(\beta(M+W_i-\Delta F))} - \sum \frac{1}{1+\exp(\beta(M-W_j-\Delta F))} = 0$, where M denotes $M = \beta^{-1} \ln \frac{N_j}{N_i}$ [23].

BAR also depends on the overlap between the states; however, it is more robust and reliable in cases of rather poor overlap [16] (especially when compared to FEP). In any case, phase space overlap is a necessary condition, without overlap the method will fail. (Whether BAR outperforms thermodynamic integration crucially depends on the smoothness of the integrand. For more pronounced changes in molecular properties between the computed states, BAR seems to be superior [13].)

MBAR, which is used in the `transformato` package to estimate the free energy differences, is a further improvement of the BAR method. BAR combines information from one forward and one backward simulation. In contrast to BAR, MBAR does not only consider samples of two states, but data from all states is used. Weights have to be determined for all state combinations. Again, the emerging equations give rise to a self-consistency problem which can be solved, e.g., via iteration methods or a Newton-Raphson solver. For state i, an estimation of free energy can be computed as

$$F_i = -\beta^{-1} \ln \sum_{j=1}^{K} \sum_{n=1}^{N_j} \frac{\exp\left(-\beta u_i\right)}{\sum_{k=1}^{K} N_k \left(\beta F_k - \beta u_k\right)}. \tag{2.9}$$

The estimate depends on all samples K, in the case of only two states, MBAR reduces to BAR. Although this formula gives the estimation for state i, it is determined only up to an additive constant. Thus, only free energy differences, $\Delta F_{ij} = F_j - F_i$ are meaningful. [24]

MBAR is also related to another technique for evaluating free energy differences, the weighted histogram analysis method (WHAM) [25]. As in MBAR, data from all samples is used simultaneously, but the data is partitioned into bins which are used to generate histograms of the probabilities. Here, MBAR equals the limiting case for a bin width of zero. [24, 26]

2.3 Soft-core potentials

Alchemical transformations rely on a coupling parameter λ which is used to gradually modify interactions. For example, by scaling λ, one can weaken the interaction with one part of the system and remove them completely at the final state. (In the simplest case, there are only two states corresponding, e.g., to an atom present in one of the

two molecules but not in the other one. If the two molecules have more considerable differences, this annihilation process has to be carried out for each atom, which has to be transformed into a dummy atom.) The term 'van der Waals endpoint problem' (or even 'catastrophe') denotes several problems which can occur when a particle is removed (i.e., turned into a dummy atom by turning off its intermolecular interactions). [27]

We illustrate the types of problems, which can arise, in the case of a linear dependence of the coupling parameter, i.e.

$$U(\lambda) = U_o + \lambda \sum_{1 \leq i \leq N-1} u_{i,N}. \tag{2.10}$$

Eq. 2.10 represents a system of N interacting particles. $N-1$ interact normally throughout (this is represented by the constant term U_0), but the interaction between the N^{th} particle and its siblings is scaled by λ. For $\lambda = 0$ and $\lambda \approx 0$ various issues emerge. If $\lambda = 0$, there are no interactions (and hence no repulsion) and the non-interacting dummy particle can be located at the exact position of another particle. This could give rise to errors resulting from divisions by zero. (In contrast to the next two scenarios, this seems to be a minor problem avoidable by efficient coding, i.e., implementing an additional clause for this condition to avoid the division by zero. In fact, it appears that all common MD simulations packages manage this case automatically [27]).

For $\lambda \to 0$, numerical instabilities can occur because even within a small time-step, the interactions abruptly can become highly repulsive. It should be noted that this problem emerges at values $\lambda \approx 0$, but not at $\lambda = 0$ (i.e., when the particles are completely decoupled).

If thermodynamic integration is used, a related problem occurs because $\left\langle \frac{\partial U}{\partial \lambda} \right\rangle_\lambda$ can become singular. This is obvious for the example using a linear pathway: $\frac{\partial U(\lambda)}{\partial \lambda} = \sum_{1 \leq i \leq N-1} u_{i,N}$. As for the problem leading to a division by zero, the fact that one of the still interacting particles can be located at the same place in the simulation box as the dummy atom causes this quantity to change unpredictably and, in the worst case, become singular [27].

The established way to avoid such problems is the usage of soft-core potentials [28–30]. An additional term is added to the particle-particle distance in the Lennard-Jones potential so that no division by 0 occurs and the corresponding derivative does not become singular. One possibility is that the usual Lennard-Jones potential is replaced by a slightly modified potential which ensures that at $r = 0, \lambda > 0$ the divisor is never 0 and the division by zero is avoided:

$$U_{LJ}(r, \lambda) = (1 - \lambda) \left(\frac{A}{(r^2 + \lambda\delta)^6} - \frac{B}{(r^2 + \lambda\delta)^3} \right) \tag{2.11}$$

However, the usage of soft-core potentials has some limitations. In particular, the availability of soft-core potentials depends on the used software package. Free energy calculations have to be explicitly supported.

2.4 Dummy atoms, Single/Dual topology

Usually, the number of atoms of both molecules of interest, i.e., of the two end states of an alchemical free energy calculation, is not the same. However, the atom number has to stay constant (as the simulation takes place in the canonical ensemble). Because of that constraint, so-called dummy atoms are necessary [1]. These dummy atoms do not participate in any non-bonded interactions, but they have to be connected via bonded interactions to the rest of the molecule they belong to so that they do not get detached and float through the simulation box.

There are two main approaches for setting up alchemical mutations, both involving dummy atoms:

In single topology, during the mutation process, physical atoms, which belong only to the start state, are transformed into dummy atoms at the end state. Conversely, as the alchemical transformation proceeds, dummy atoms present at the initial state are re-transformed into physical atoms of the second molecule.

In dual topology, no direct mutation of real atoms into dummy atoms occurs. However, both physical molecules are augmented with all dummy atoms of the respective other state (hence, there is no state during the simulation without dummy atoms). Thus, the number of dummy atoms equals the number of atoms which have no direct correspondence in the other molecule. Usually, in total, even more dummy atoms are present in the system as in a single topology setup [1].

The implementation of such dummy atoms, however, is not without pitfalls. In single topology, errors can arise if too many bonded interactions are kept. The order in which interactions are turned off has to be constrained by specific rules to ensure that the contributions exactly cancel out each other, and attention has to be paid to under which conditions dummy atom contributions cancel out exactly. [1]

2.5 Serial atom insertion

As an alternative to modifying the coupling parameter, one can try to create new states by turning off atoms in one step without intermediate values. This approach, called serial atom insertion, was introduced In [27]. Atoms are turned off serially (one after one or in small batches). There is no gradual damping of the interactions; the LJ-interactions of a molecule are either present ($\lambda = 1$) or turned off ($\lambda = 0$).

A sufficient phase space overlap between neighboring states is crucial for any alchemical free energy calculation. So, the question arises whether turning off one atom in one step is in agreement with this prerequisite. The feasibility of this approach relies on Bennett's acceptance ratio, which was shown to work with neighboring states created by serial atomic insertion [27].

In particular, serial atom insertion has the advantage that it works even for MD programs which lack explicit support for free energy simulations and soft-core potentials are not needed (because the coupling parameter takes only the values 0 and 1). The free energy differences between states can be assembled from 'normal' simulations.

The most severe restriction due to the van der Waals endpoint problem is the instability of the integrator near the end state where an atom has almost vanished (i.e., $\lambda = 0$ or $\lambda = 1$). As such states are not used in serial atom insertion, this problem is automatically avoided.

The feasibility of this approach depends on the sufficient overlap between the states. Using BAR/MBAR, no intermediate steps are necessary and atoms can be turned off one by one. Contrariwise, thermodynamic integration cannot be used because it is precisely the scaling of the interaction parameter between 0 and 1 which is avoided. This also implies that the singularity risk of the derivative due to the van der Waals endpoint problem can be neglected (as this part of the problem only concerns thermodynamic integration).

3 transformato

`transformato` uses a common core scaffold which contains the subset of atoms which are present in both molecules (i.e., a one-to-one correspondence between atoms of both molecules, which in the test cases shown below is always based on atom identity). Both initial states of the alchemical transformations initialized by `transformato` do not contain any dummy atoms, but consist solely of the physical atoms of the respective molecules. However, dummy atoms are generated via two separate alchemical paths leading to the common core. Starting from the initial states, physical atoms are successively turned into dummy atoms until the common core structure is reached.

In each transformation step, one physical atom (or, if phase space overlap is sufficient, a batch of adjacent atoms) is changed into a dummy atom (until the common core is attained).

The common core architecture circumvents some potential problems associated with the single and double topology approaches for dummy atoms. The physical end states of the molecules are mutated until the common core structure is attained. This implies that both starting states do not contain any dummy atoms (these states are identical to the physical molecules of interest). Therefore, it is ambiguous if the alchemical transformations implemented in `transformato` rather belong to the single or the dual topology paradigm: As in the latter, different dummy atoms are generated for each molecule along the path to the common core, but — in contrast to the usual dual topology approach — the final states are free of any dummy atoms.

The 'removal' of atoms, i.e., the mutation into dummy atoms, is performed using the serial atom insertion approach described above; hence, `transformato` does not rely on the use of soft core potentials. The computation of the resulting energy differences is based on MBAR.

Therefore, by setting up the mutation path between two molecules across the connecting common core and using serial atom insertion, the `transformato` workflow is independent of the underlying molecular dynamics package and of the availability of explicit free energy calculation code. In principle, `transformato` can work on top of every molecular dynamics simulation package.

Input can be created via CHARMM-GUI [31]. The solution builder of the website generates appropriate files to run MD simulations for the systems for which free energy differences shall be determined (e.g., for relative solvation free energy differences, input files for both molecules in a water box and in vacuum). [2, 32].

3.1 Common core approach

The main condition for the common core of two molecules is the existence of a one-to-one correspondence of atoms, i.e., the existence of a graph isomorphism. The `transformato` workflow imposes some further conditions on the properties of the common core:

The junction between the common core and dummy region has to be unique; only one dummy region is allowed to be connected via one bond to the common core. In particular, the maximum common substructure must not encompass partial rings (which would imply that dummy regions are connected via multiple bonds). (In general, such transformations can pose intricate problems because ring breakage can lead to fast changes in free energy and cause significant estimation error [33].) To meet these requirements, adjusting of the parameters and algorithms for finding the common core is important. For facilitating the construction of appropriate common cores, it was also necessary to improve the processing of the hydrogen atoms. As hydrogens are turned off beforehand in one step and not via serial atom insertion like the heavy atoms, it would be detrimental to consider them in the common core generation (see section 'Processing of hydrogen atoms' in the next chapter). In previous versions of `transformato`, hydrogens were included in the common core generation steps, which led to small or often even completely infeasible common cores. Before the common core construction is carried out, hydrogens have to be removed from the molecule representation. This is an important step because the presence of hydrogen atoms can lead to a different common core (which is created, using default settings, by maximizing the number of corresponding atoms) and subsequently a suboptimal mutation route.

3.2 `transformato` workflow

During the mutation process, the contributions of the non-CC atoms are turned off gradually; five stages can be discerned [2]: In the first step, the electrostatic interactions of dummy atoms are gradually turned off. Next, Lennard-Jones interactions of all hydrogen atoms outside the common core are removed. This can be carried out in a single step. During the third stage, the LJ-interactions of the non-CC non-hydrogen atoms are processed. One atom per step is turned off following the serial atomic insertion approach. (Possibly, a small group of atoms could be turned off in one step, but it is not recommended to process more than two atoms at once to ensure sufficient phase space overlap between the states.) The atom which connects the common core and the dummy region is called the junction atom and indicated as 'X' [2] (In the plots in the next chapters, for simplicity, the indication of the junction atom is sometimes omitted. However, it should be stressed that, e.g., a transformation to a methane common core yields, in fact, a CH_3X common core.). This dummy atom which is directly connected to the common core needs special treatment. In contrast to the other atoms of a dummy region, its LJ interactions are not turned off completely, though its partial charge has to be zero. [2] This junction atom ensures that the double free energy differences are not influenced by contributions of dummy atoms. [1] In the last step, it has to be ensured

that both common cores are identical, e.g., differences in charge distribution have to be adjusted and the bond between junction atom and the common core atom has to be identical.

4 Problem description

The main objective of this work was the assessment and improvement of some routines used in the free energy package `transformato`. A further goal was to minimize the necessity of manual adjustments by the user; i.e., reasonable mutation routes should be generated automatically. The proposed route should be directly usable for the further `transformato` workflow.

In a first step, the reliability of the current `transformato` workflow had to be assessed, for instance the quality of the proposed common cores. Different settings for the construction of the maximum common substructure were compared.

The order of the transformation steps was optimized, especially for the case of more complex mutation routes which occur for connected dummy regions involving ring structures, multiple chains or different atom types.

Particularly intricate problems occur for ring structures. In contrast to a chain, which usually has only one possible mutation order without leading to disconnected components, multiple possibilities exist for rings. Because of this, the evaluation of the mutation algorithms in the following sections will focus on such transformations. The mutation of atoms should neither generate vacancies in the inner part of the molecule nor should rings remain opened longer than necessary, and a sufficiently systematic and — especially concerning rings — symmetric processing of the nodes has to be performed. These rules have to be implemented by maintaining the crucial constraint that no atoms are detached from the main part encompassing the CC, i.e., no disconnected components must emerge under any circumstances.

Using a graph representation of the involved molecules, the construction of the intermediates between the final states was optimized. New algorithms were written in Python and subsequently integrated into the existing `transformato` package. Finally, the effect of different algorithms on the efficiency of the free energy calculations was validated through molecular dynamics simulations.

To obtain a reliable test set of molecules, SDF (Structure Data File)-files containing positional information about suitable ligands were downloaded from the PDBbind-CN [34] database. These test molecules cover a broad range in size, complexity and potentially intricate compounds, like polycyclic structures and highly branched chains.

4.1 Overview of algorithms and software packages used

The `transformato` package is written in Python. Therefore, Python packages were also used for molecule processing and graph representations. The creation of molecule objects and the determination of the maximum common substructure is done via RDKit [35].

NetworkX [36] provides functions for graph visualization and analysis. It is easy to convert molecules created using RDKit into NetworkX graph-objects and hence utilize the functions of NetworkX for the molecules and CCs constructed. Particularly, graph traversal algorithms like breadth-first and depth-first search can be easily implemented (see below).

4.2 Assessment of CC settings

RDKit allows the search for a maximal common substructure (which can serve as the CC for `transformato`) via the `rdFMCS.FindMCS`-function. Per default, the objective is to maximize the number of atoms, albeit different settings, like maximizing the number of bonds or ignoring or equalizing specific atom types are available. Currently, for `transformato` maximization of atoms and atom identity is used. As stated above, the presence of hydrogens can influence the maximum common substructure heavily because the number of atoms is modified, and this quantity is maximized for the maximal common substructure.

Settings concerning the allowed involvement of ring structures in the CC are of crucial importance. Firstly, these parameters can influence the CC construction drastically and, secondly, they can even be decisive whether the generated CC is valid for the `transformato` workflow. Especially for the generation of CCs for polycyclic molecules, e.g., sterols, these parameters are of utmost importance. Important ring-related settings are `ringMatchesRingOnly`, `completeRingsOnly` and the `ringCompare-parameter`.

To obtain valid CCs of molecules involving cyclic structures for the processing of `transformato`, `ringMatchesRingOnly` and `completeRingsOnly` must be set to True: The former argument indicates that ring atoms of one molecule are only matched against ring atoms of the other molecule, the latter ensures that no partial rings are involved in the CC. Especially, the latter constraint is a necessary condition for a valid CC. Otherwise, if partial rings take part of the CC, dummy regions will be inevitably connected to the CC via multiple bonds.

The `ringCompare`-parameter parameter accepts the `StrictRingFusion`-argument. It imposes that in the case of multiple rings, aromaticity is properly considered. Fig. 4.1 illustrates the effect of the parameters on the CC of two cyclic molecules, cholesterol, and cortisol. However, as shown in the bottom row of fig. 4.1, enforcing `StrictRingFusion` can still lead to maximum common substructures that are not valid `transformato` CCs in the case of a dummy region which is connected via multiple ring atoms of the same ring with the common core.

These problems occur because the requirements for a valid `transformato` CC are stricter than the constraints imposed by RDKit (even when all ring-related parameters are applied, i.e., `ringMatchesRingOnly`, `completeringsonly` and `strictringfusion`): For `transformato`, it is not only indispensable that the CC solely comprises complete rings, but also the dummy regions consisting of the non-CC atoms must not contain parts of a ring structure (which implies that there has to be a unique connection between dummy region and CC). This is due to the aforementioned constraints on the junction atom X:

When it is connected to the CC via one single atom (and, as the CC consists of a single connected component, it, therefore, is not part of a cyclic structure), it is guaranteed that the contributions from dummy atoms cancel out and do not influence the calculation of the double free energy differences [2].

When an invalid CC is generated, it appears to be advisable to warn the user that the generated CC does not fit the expectations of the `transformato` workflow. The new mutation algorithms presented below can deal with such invalid CCs. A helper function arbitrarily chooses one of the connections between the CC and the dummy region, and the other ones are ignored for the mutation path. However, this only ensures that a mutation route is returned; the current `transformato` workflow is not adapted to deal with such CCs properly. Therefore, it only makes sense for test purposes and may not provide a reasonable input for the regular workflow.

Alternatively, one can check after the creation if the CC is valid. If it is not, one could for instance search for a new CC encompassing fewer atoms until a valid CC is found (see below). A further option would be to prohibit the involvement of ring atoms for the CC of this particular pair of molecules.

Furthermore, an efficient and straightforward solution for avoiding the generation of invalid CCs, which always yields the best — i.e., largest — valid CC if one exists, has been implemented: The basic observation is that if CC atoms within a cyclic region give rise to an invalid CC, it is impossible that any atoms within this region could participate in a valid one (if this would be the case, the whole cyclic region would be already in the generated CC). After generating a CC with standard parameters, it is checked if the additional requirements concerning the ring structure hold, i.e., if the non-CC atoms (as well as the CC atoms) of both molecules do not participate in a partial cycle. (Alternatively, one could check if a path between the non-CC atoms adjacent to the CC, i.e., the X-atoms, exists. If this is the case, the CC obviously is not valid, since no unique atom which connects the CC and a specific dummy region is present.) If this is not the case, the rings that contain atoms which participate in the partial ring causing the invalid CC are removed from the representation used for creating the maximum common substructure and afterward a new search for a valid CC is started. This procedure is repeated iteratively until a valid CC is found.

Of course, in the worst scenario, i.e., if both (non-identical) molecules only consist of ring-participating atoms, no valid CC for `transformato` may exist. Similarly, the size of the valid CC may be tiny compared to that of the physical end states, which may make the CC approach as implemented in `transformato` very inefficient. In general, if at least one of the molecules solely consists of a polycyclic compound without any functional groups etc., and the other one does not have the same compound, no valid CC can exist. A — maybe rather contrived — example of such a pair of molecules and the construction of a valid CC via the iterative approach is given in fig. 4.2. One sees that even the activation of the ring-related parameters of RDKit is not sufficient to yield a valid CC, whereas the iterative approach gives the desired result. However, the resulting valid CC (which has the maximally possible size) is small compared to the original size of the molecules, despite the apparent similarity between both structures.

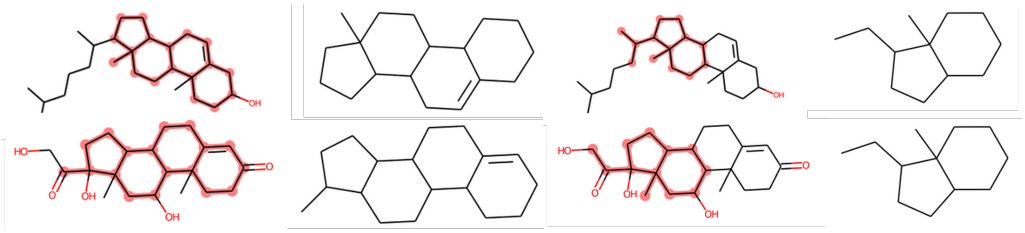


Figure 4.1: 'Strict fusion' can affect the construction of the CC drastically, illustrated for cholesterol (top row) and cortisol (bottom row). First and second columns: CC of the two molecules without strict fusion; third and fourth column: common core of the two molecules with strict fusion. In the images of the full molecule graph (first and third column), CCs are marked in red. In the example shown, none of the settings yield a valid CC for `transformato`; the iterative approach explained in the main text would be necessary to obtain one.

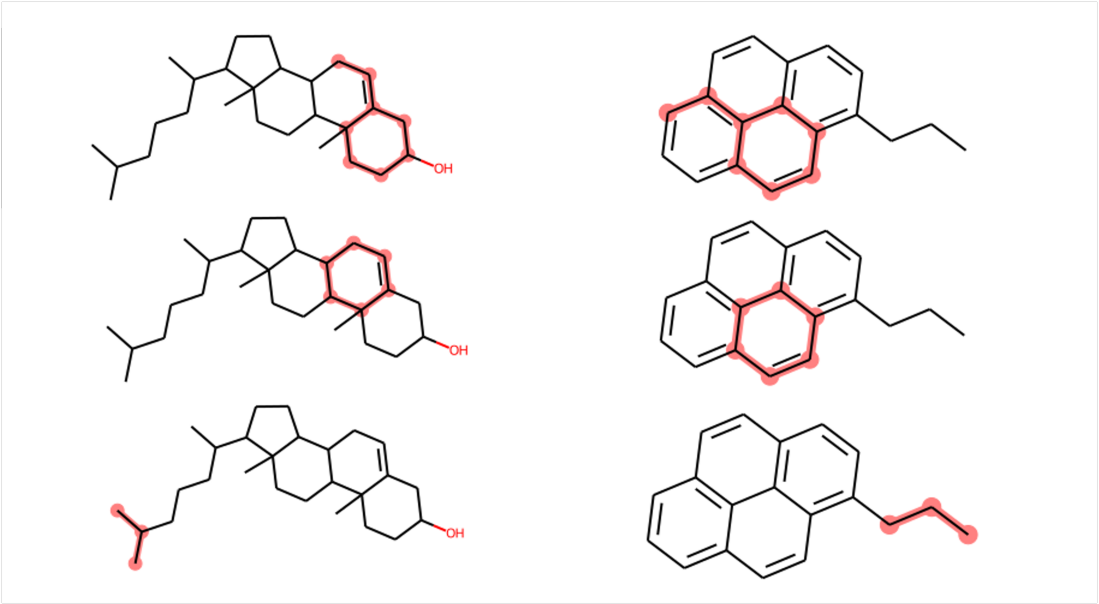


Figure 4.2: CCs of cholesterol (left) and 1-propylpyrene (right); top row: CompleteRingsOnly = False; middle row: CompleteRingsOnly = True; bottom row: iterative approach to obtain a valid `transformato` CC

4.3 Graph algorithms

Using NetworkX and RDKit, the molecules and their CC are represented as graphs (in which nodes indicate atoms and edges bonds between them). The selection of the optimal mutation route can be understood as a graph traversal problem, in which the constraints mentioned above are either implemented via the weights of the edges or by sorting. Hence, the main task was to find suitable algorithms and graph initializations to ensure an optimal processing of the mutation path.

Depth First Search (DFS) follows each chain of the graph as long as possible, i.e., until a leaf node is reached. In contrast, Breadth First Search (BFS) explores all chains simultaneously [37]. Problems and differences of both algorithms are illustrated using several examples below. As will be discussed in more detail, the main problem of DFS is that it produces undesired outcomes. Often, heavy atoms near or next to the CC atoms are processed during the first steps of the graph algorithm, which may lead to an early opening of rings. Processing is completed only at a (much) later stage. Phrased differently, atoms are turned into dummy atoms in a highly 'asymmetric' manner.

In each of the algorithms implemented, the root of the graph traversal is the node which connects the dummy region and the CC, i.e., the junction atom 'X'. The shortest paths to all nodes of the dummy region are determined.

The longest of these shortest path identifies the node which has the greatest distance from the root (i.e., the atom with the greatest distance from the CC). Therefore, the last element of the list returned by the search algorithm indicates the atom which has to be removed first; hence, the list of mutations orders has to be reversed.

If weighted graphs are used, the Dijkstra algorithm can be applied. It finds the shortest path between two nodes or between a root node and all other nodes of the weighted graph (the weights indicate the edge length from one node to the other one). For unweighted graphs (or, equivalently, graphs with uniform weights), the Dijkstra algorithm reduces to BFS. Fig. 4.3 shows the different routes for modified weights. In the test cases presented below, all graphs are initialized with uniform weights.

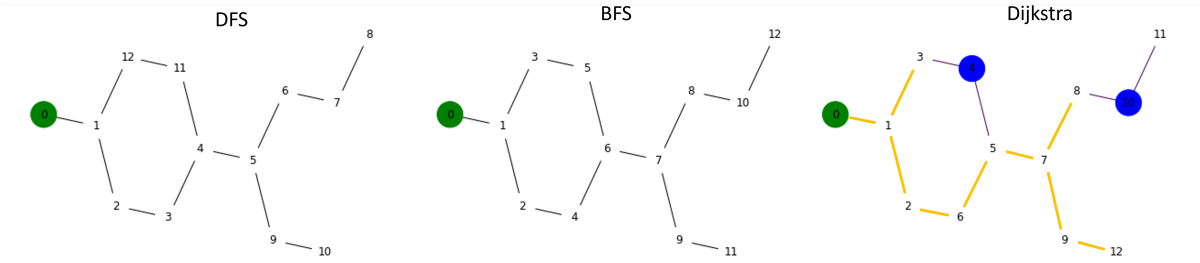


Figure 4.3: Comparison of different graph traversal algorithms. Green nodes represent the root atom; left: depth first search (DFS); middle: breadth first search (BFS); right: Dijkstra algorithm. In the Dijkstra algorithm, blue atoms have increased weights; the edges connecting blue colored nodes have increased weights leading to a mutation route differing from BFS; in the `transformato` workflow, the final processing of the nodes happens in reversed order; i.e., the atom first visited is removed last

4.4 New functionality added

4.4.1 Functions for creating mutation paths

To use the newly implemented mutation algorithms, the graph object is initially created with weights according to the atom type stored in a dictionary. The simulations shown below use uniform weights; hence, all atom types are treated equally. However, it is possible to modify these to enforce a specific mutation route (e.g., accelerating or postponing the processing of heteroatoms).

The core functionality is given by the mutation processing functions. Currently, three new functions are implemented, in addition to the simple, existing DFS-approach. These four main functions for computing mutation routes are:

1. `_calculate_order_of_LJ_mutations:` this function performs the DFS algorithm, and is the only one previously available in `transformato`. It may lead to defective mutation routes (various examples are shown below and compared to the routes created by the improved algorithms) and, hence, should only be used for test purposes. The other three algorithms are new, and all of them resolve most of the problems of this existing earlier algorithm (especially issues related to the isolated removal of ring atoms).

2. `_calculate_order_of_LJ_mutations_new:` the BFS/Dijkstra-algorithm is applied once for creating a mutation route.

3. `_calculate_order_of_LJ_mutations_new_iter:` the BFS/Dijkstra-algorithm is applied iteratively, i.e., after each removal of an atom, until all atoms of the dummy regions are processed.

4. `_calculate_order_of_LJ_mutations_new_iter_change:` similarly to the last function, this approach works iteratively. In contrast to `_calculate_order_of_LJ_-mutations_new_iter`, after each removal of an atom, the algorithm/routine for the next step is chosen depending on the current state: If the last processed atom belongs to a cycle or a chain, the algorithm prioritizes the pruning of this cycle or chain before other parts of the molecule are processed.

These functions can be further modified by passing arguments which activate some helper functions. Auxiliary functions carry out tasks to ensure the desired mutation route; e.g., the function `cycle_checks` counts the number of cycles an atom participates in. Further features of all algorithms are 'preferential removal'; i.e., if two atoms have the same priority (given by the current weight) for the next mutation step, the weight of the atom which is next to an already removed atom is updated so that this atom is processed next. In the following, the most important of these functions are shortly described:

- `cycle_checks(G)`: this function takes a NetworkX-graph object as input, checks which atoms participate in how many cycles/rings and finally returns a dictionary with the atoms as key and the number of rings the atom is participating in as value and a dictionary with the degree (i.e., number of edges) of each atom node. It is currently used in `_calculate_order_of_LJ_mutations_new` (via the `change_-route_cycles`-function).

- `change_route_cycles(route, cycledict, degreedict, weightdict,G)`: this function is used in `_calculate_order_of_LJ_mutations_new` and sorts nodes according to degree, cycle participation and information about the nodes which have been removed immediately before. The preliminary mutation path is sorted using a cycle and degree dictionary. If nodes have the same weight (i.e., distance from root), the node participating in more cycles is removed later. If nodes have the same weight (i.e., distance from root) and same cycle participation number, the node which has more neighbors already processed is removed earlier.

- `cycle_checks_nx(G)`: This function modifies the weight of the graph, nodes participating in many cycles get lower weight. It is currently used in `_calculate_-order_of_LJ_mutations_new_iter` and `..._new_iter_change`. It returns a nx-graph-object with updated weights (according to cycle participation of the atom).

- `order_checks_nx(G, removearray, G_total)`: This function performs the 'preferential removal'. If a node is connected to the node removed in the previous step, its weight gets a small increase so that the removal of this node is prioritized. It is currently used in `_calculate_order_of_LJ_mutations_new_iter` and returns a nx-graph-object with updated weights.

If the `cyclecheck`-argument of the new mutation algorithms is set to True, updates are updated according to cycle participation (as the systematic processing of ring structures is one of the central goals, this functionality should always be enabled, except for comparison

and test purposes). If in the case of `_calculate_order_of_LJ_mutations_new and _calculate_order_of_LJ_mutations_new_iter` the options `ordercycles` or `ordercheck`, respectively, are also set to True, weight updating according to preferential removal decides that the node in which neighborhood nodes have already been turned off is removed next if there is no reason to prioritize one of the nodes (i.e., without preferential removal, the weight of the nodes would be the same).

In each algorithm, all nodes of the graph (i.e., atoms) are usually initialized with the same weight (e.g., 5). Alternatively, the user could also pass an individual dictionary with different weights for each atom type. For the graph algorithms, NetworkX is used. For example, the BFS- / Dijkstra-algorithm starting from the node connecting common core and dummy region is implemented via the NetworkX-function `single_source_dijkstra` which determines the path length of all dummy nodes to the root.

The main difference between these algorithms is that in `_calculate_order_of_LJ_mutations_new_iter` and `..._iter_change` the graph traversal part performed using the Dijkstra algorithm is applied after each exclusion step; i.e., $n!$ atoms are visited instead of n. Even for relatively large molecules, the additional computational cost is negligible, in particular in comparison to the computational time needed for the search of the CC. The advantage of the iterated version is that after each mutation step, weights can be updated or even the search algorithm can be modified. This feature allows for different mutation strategies depending on the current state. Such an approach is demonstrated in `_calculate_order_of_LJ_mutations_new_iter_change`. This algorithm takes into account whether the last removed node was part of a chain or a cycle. Depending on the state, the chain, or cycle, is processed fully before the algorithm moves on to other parts of the molecule. Fig. 4.4 demonstrates the difference between the two versions of the iterated algorithm. An additional, more practical difference should be kept in mind. Whereas in the non-iterated algorithm the mutation route has to be reversed (since the atom node with the highest distance from the root is the first which has to be turned off), in the iterated versions, at each iteration the node with the highest distance is added to an array which determines the final mutation route.

The mutation routes computed by the various algorithms can be visualized directly via RDKit. The route is represented by a color gradient used for the atoms involved in the mutation process (`_show_common_core_gradient`). By default, the color spectrum ranges from red to green; the first atom to be removed is colored red, the last in green. Furthermore, an animated 3D-visualization of the mutation process is implemented using py3Dmol (`animated_visualization_3d_v2`) [38].

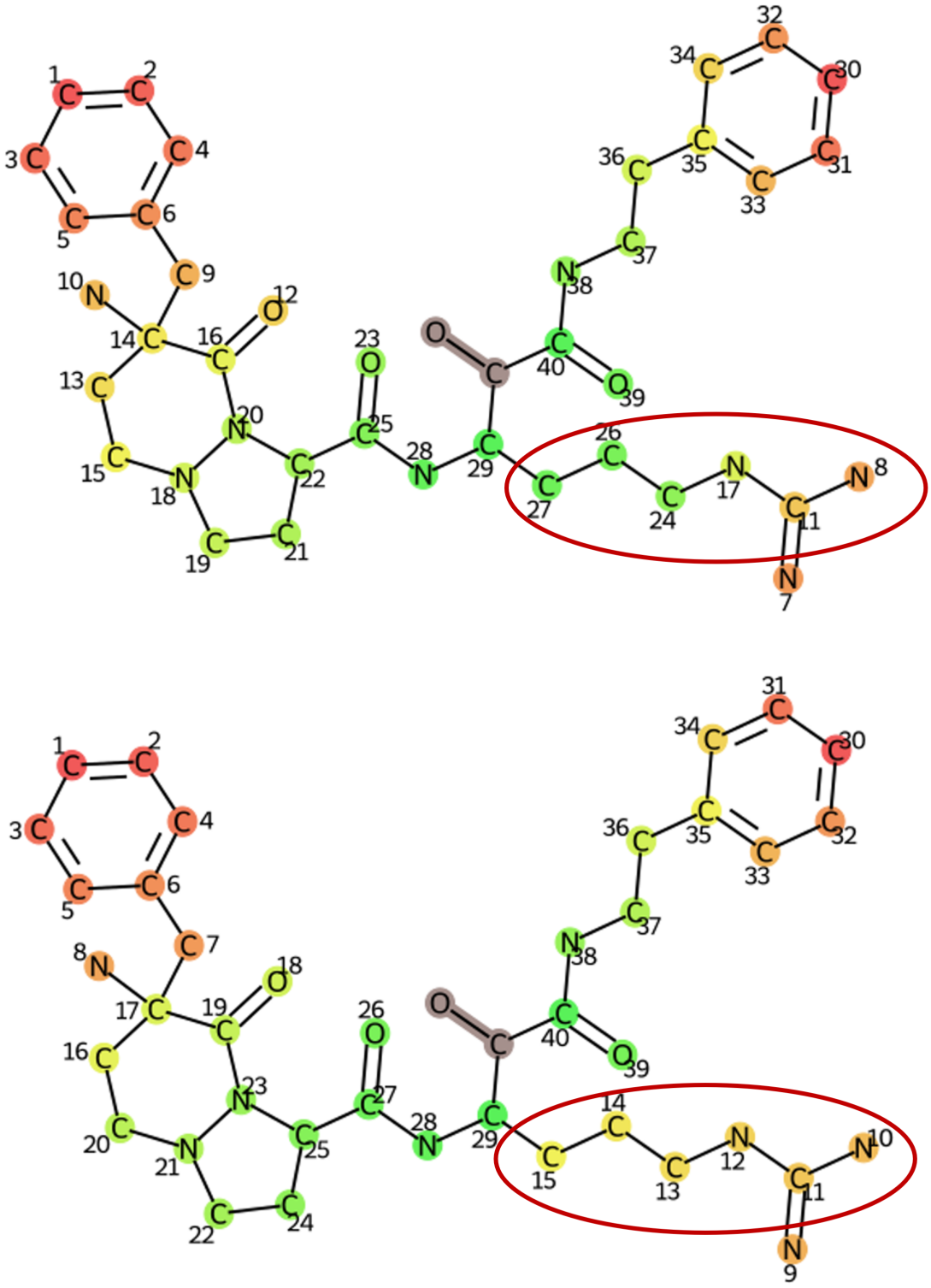


Figure 4.4: Example of the differences between the `iter-` and `iter_change`-algorithms; Top: `iter`; Bottom: `iter-change`; `iter-change` processes all atoms within a chain or a ring at once (if possible) before switching to other parts of the molecule

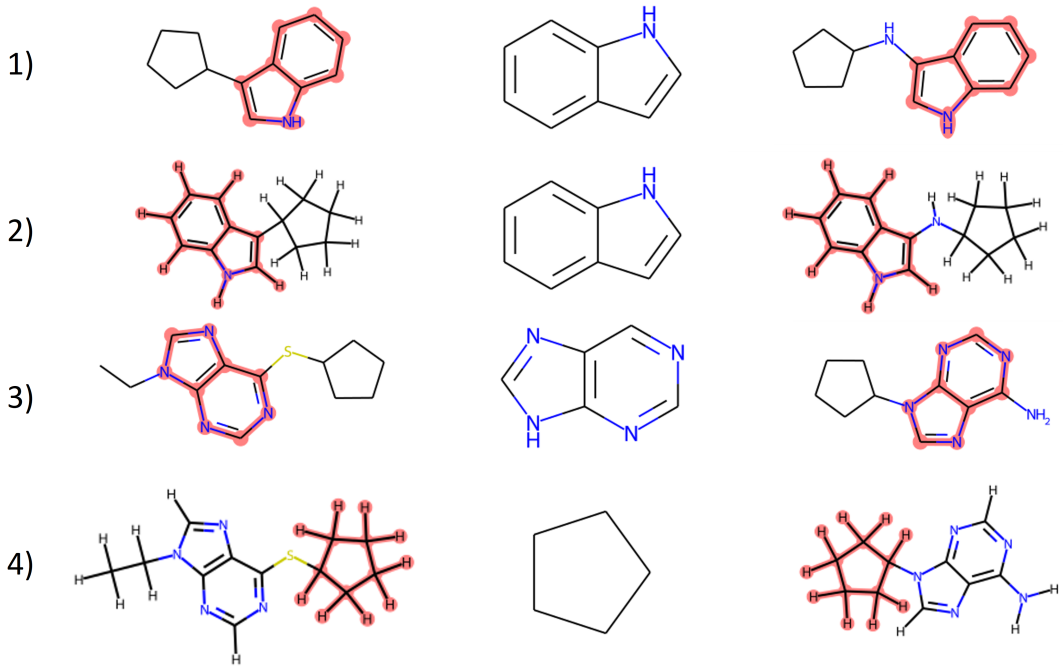


Figure 4.5: Examples demonstrating that the addition of hydrogens can influence the construction of the CC; left: initial state; middle: CC; right: final state. Rows 1)–2) and rows 3)–4) each show the same pair of molecules, in the respective upper row without, in the lower with hydrogens. In the representations of the end states, the CCs are marked in red. While in the first example (rows 1)–2)), the inclusion of hydrogens does not affect the CC generation, the CCs generated in rows 3) and 4) change when hydrogens are added to the RDKit-molecule representation

4.4.2 Processing of hydrogen atoms

Fig. 4.5 illustrates that the presence of hydrogens can influence the CC generation dramatically. In particular, the maximum common substructure for a molecule representation with hydrogens might contain fewer heavy, i.e., non-hydrogen, atoms than the substructure for a representation with hydrogens removed (although, including the hydrogens, the total number of atoms is higher in the first case). Since hydrogen atoms are turned off in a single step en bloc during the `transformato` workflow, the mutation route algorithm should not take hydrogens into account. Therefore, the CC should be generated for a molecule representation without considering hydrogens. Nonetheless, it is necessary that the molecule representations processed by `transformato` contain all hydrogens in explicit form because the indices of these atoms in the underlying data structures are used in some steps, e.g., for scaling the van der Waals interactions of the hydrogen atoms to zero.

This problem was solved by removing and adding hydrogens appropriately. In a first

step, the `transformato` function determining the CC (`_find_mcs` in `mutate.py`, which is the Python file containing most `transformato`-functions responsible for generating the CC and the mutation paths) had to be modified. To obtain the desired CC but retain the `transformato` workflow, the following scheme was implemented: A deep copy of both molecules is created. The hydrogens of these copies are removed; then their CC is computed. For both molecules, the indices of the atoms corresponding to the CC are determined. Finally, it is checked for each CC atom of both molecules whether hydrogen atoms are in its neighborhood. If such hydrogens are found, their indices are added to the lists of CC atoms. These lists of atoms, including hydrogens, are stored for both molecules and the function returns the maximum common substructure (determined for the molecules without hydrogens). Thus, the procedure yields the necessary output for further processing in `transformato`: The molecule representations and lists of CC atoms for both of the molecules include hydrogens, but the maximum common substructure giving rise to the CC is computed only for the heavy atoms.

Similarly, the functions for computing the mutation routes had to be adapted for input molecules containing hydrogens. By default, a helper function removes the hydrogens from the graph representation as well as the corresponding indices from the list with the atoms of the dummy regions before the mutation algorithms are applied. Afterward, the hydrogen atoms adjacent to CC heavy atoms are added to the CC.

A special problem occurs for molecule pairs with switching 'X'-atom; fig. 4.6 shows 2-/7-pyrrolidinindole as an example. The red highlighting shows a correct CC for both molecules; in each of them one should note the hydrogen atom, which is directly bound to one of the CC-carbon atoms, but itself is not part of the CC. If the hydrogens were included, the resulting CC would not be valid anymore: It is necessary that a further dummy region emerges which consists solely of one hydrogen atom because otherwise the CCs would have a different number of atoms and would not be isomorphic anymore. If, however, the CC is generated without hydrogens, the information which of the hydrogens turns into a dummy region is lost and the construction of a valid CC cannot be assured. It has to be stressed again that the presence of these hydrogen dummy regions does not affect the mutation steps, but is nonetheless crucial for the internal `transformato` workflow.

There is a straightforward way to circumvent this problem: From the substructure matches, the mapping between the CC atoms of each molecule can be read off. For each (non-hydrogen) CC atom, one counts how many hydrogens are connected to it. Finally, the minimum number of hydrogens is added to the CC. (In the case of 2-/7-pyrrolidinindole, this means that for each CC atom one hydrogen is added, except for the 2- and 7- position because for these positions in one molecule no hydrogen is connected). In this context, also the difficulty that there are possibly many substructure matches has to be mentioned.

It is even possible that substructure matches differ solely regarding hydrogens. A parameter was added to the function which searches for the maximum common substructure to handle this case. If `iterate_over_matches` is set to true, one of the substructure matches with most hydrogens, i.e., the biggest possible substructure, is selected. Fig. 4.7 shows two CCs, only differing in the number of hydrogens.

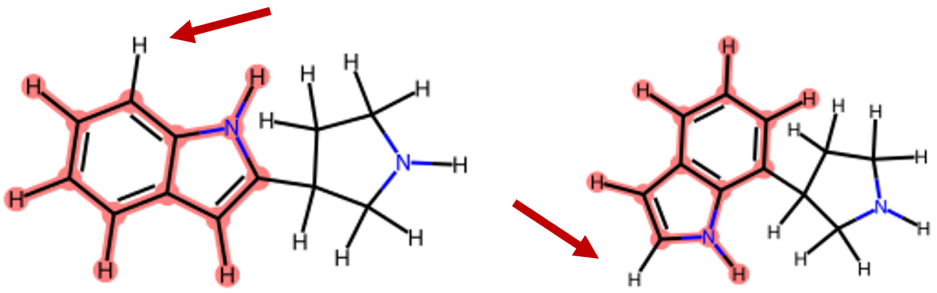


Figure 4.6: left: 2-pyrrolidinindole; right: 7-pyrrolidinindole. In each structure, one of the hydrogens indicated by the arrows — exactly the hydrogen atom which is placed at the position of the junction 'X'-atom at the other end state — must not be part of the CC (otherwise, the CC would be invalid)

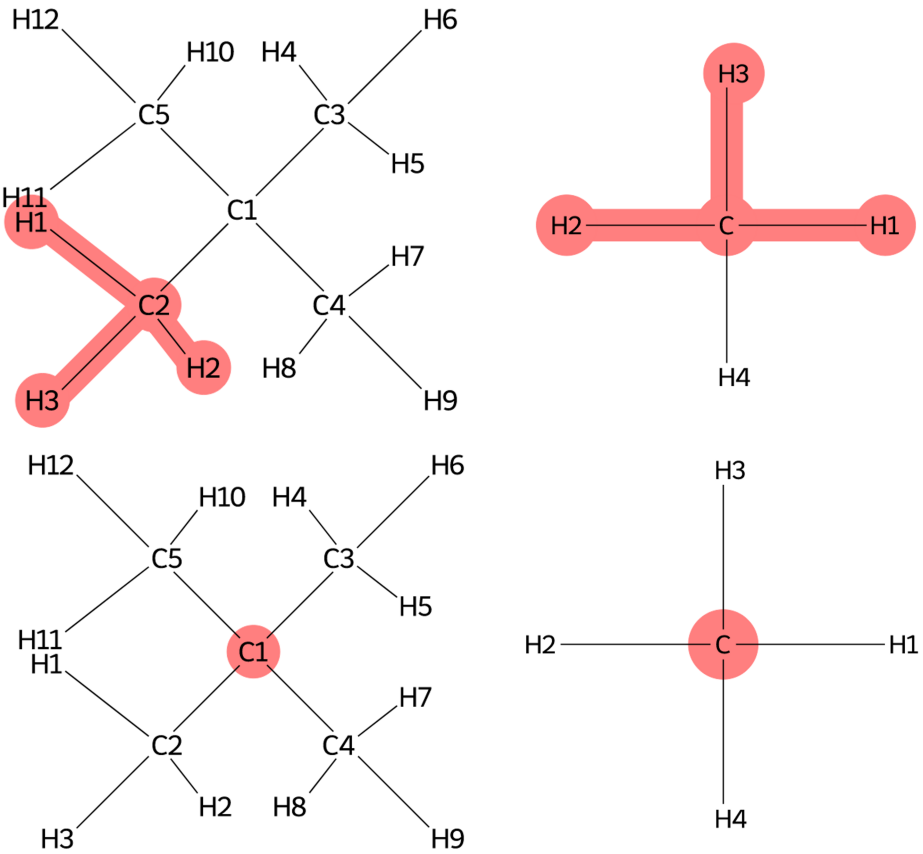


Figure 4.7: neopentane/methane CCs; upper row: CCs maximizing the number of atoms including hydrogens; lower row: CCs without considering hydrogens

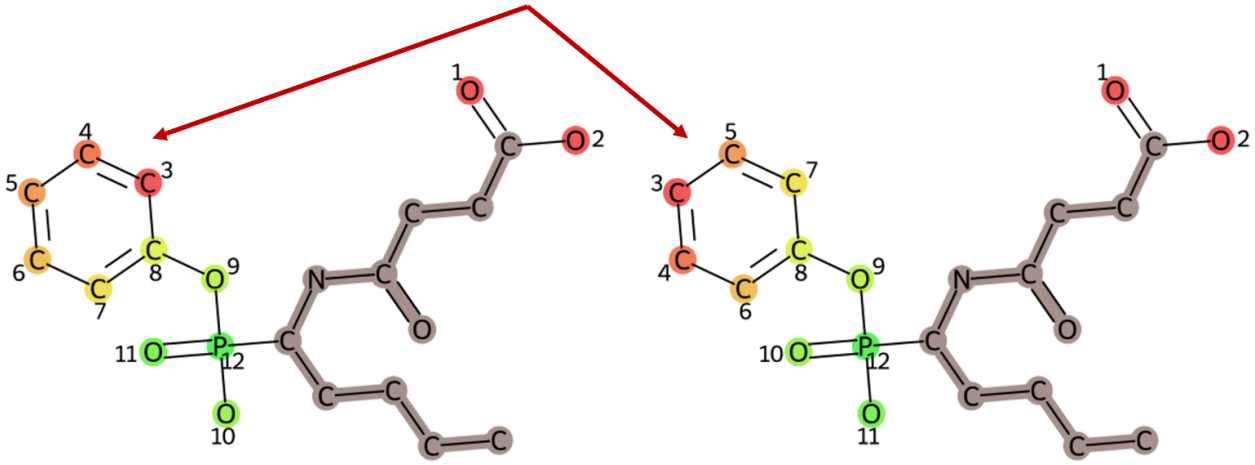


Figure 4.8: Comparison of typical DFS- (left) and BFS-mutation route (right) for a single ring structure. In all depictions of mutation routes, the CC is shown in dark, the color gradient starts at red, indicating the atom removed first, and ends at green. DFS starts at the ring atom adjacent to the carbon atom bonded to the oxygen, and, thus, ring breakage gives rise to one long chain which is processed subsequently. By contrast, BFS starts at the ring position most distant from this carbon and the two emerging chains are processed in a symmetric fashion.

4.4.3 Examples of processing molecules

As described above, the current version of `transformato` implements one suboptimal route finding algorithm using DFS and several new versions based on BFS. For single rings, BFS (or, in the case of different weights for various atom types, the Dijkstra algorithm) automatically processes the atoms in the most symmetric way, starting at the atom with the highest distance from the root), whereas DFS gives rise to a long chain. Fig. 4.8 illustrates this difference.

In more complex molecules, the systematic exploration of chains in depth first search inevitably leads to big local gaps in processing of the molecules. Fig. 4.9 illustrates this problem for a benzene ring which is directly attached to the CC. As DFS goes along one path until the end; i.e., until a leaf node is reached, four atoms of the ring are visited first, whereas the remaining two are explored last. Therefore, these two atoms are turned off first, but then the algorithm continues at a wholly different location and the remaining ring atoms persist in the system until the end of the mutation process. By contrast, BFS automatically produces the desired result for this system.

A similar scenario is shown in fig. 4.10. One ring atom (indicated by the red circle), which is adjacent to the X-junction atom, is processed last by DFS. Hence, it is processed first in the resulting `transformato` workflow, opening the ring, whereas the other atoms of

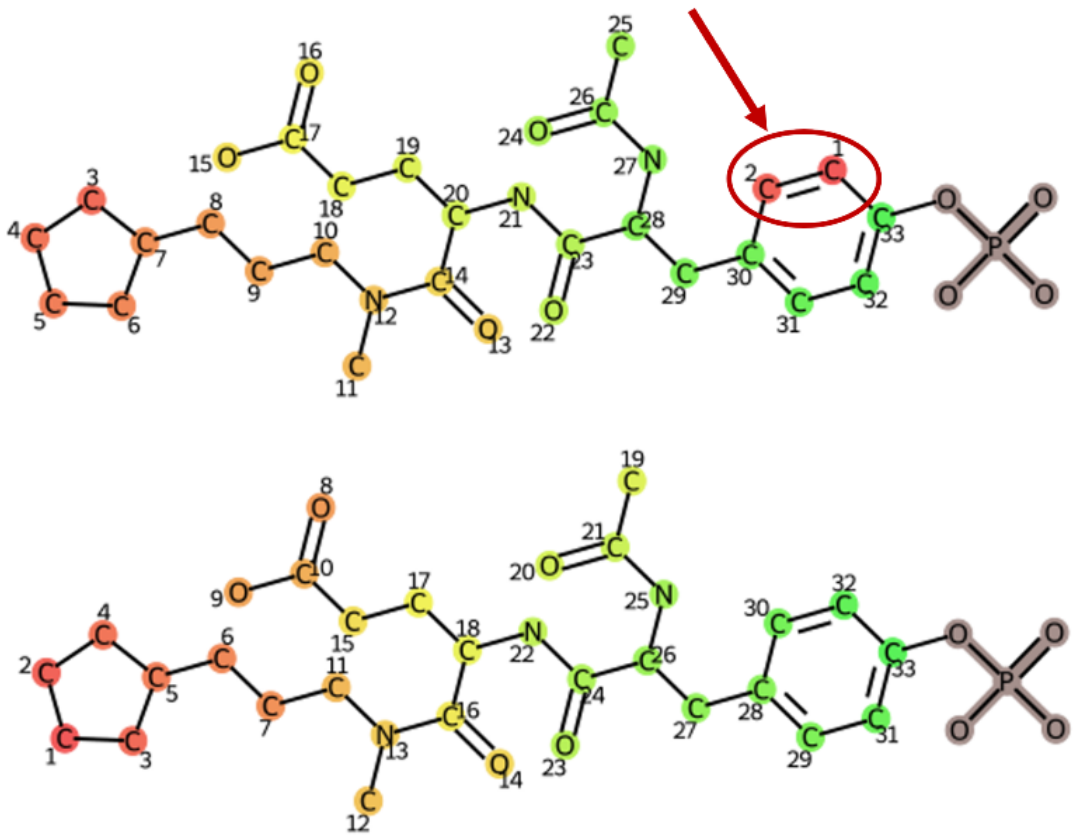


Figure 4.9: Top: DFS-algorithm; Bottom: BFS-algorithm; CC in dark; the red arrow indicates the undesired processing of the ring atoms in the DFS case.

the ring are removed much later. More complex ring structures exacerbate these problems. Fig. 4.11 shows an example where also the processing of the substituents is affected.

Multiple rings pose special difficulties for the mutation algorithms because the processing of one of the rings can easily lead to gaps in the adjacent ones. Figs. 4.12 and 4.13 illustrate that severe problems may occur when using DFS. As in the case of one ring, the exploration route implies that one of the rings is opened in such a way that it gives rise to a lengthy chain. Furthermore, an atom belonging to two or more rings is visited early, whereas other atoms of the ring are explored much later, so that both ring structures are opened and torn apart (fig. 4.12). Likewise, one half of each of the rings is turned off many steps before the remaining ring atoms are processed (fig. 4.13).

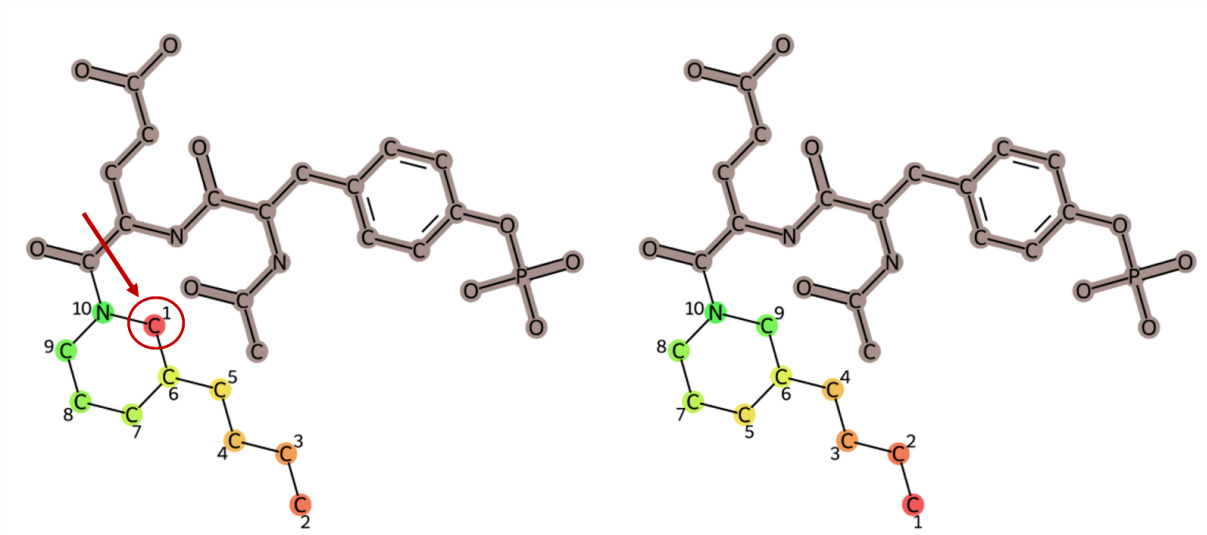


Figure 4.10: Left: DFS-algorithm; Right: BFS-algorithm; CC in dark; the red arrow and circle indicates the undesired processing of the ring atoms in the DFS case

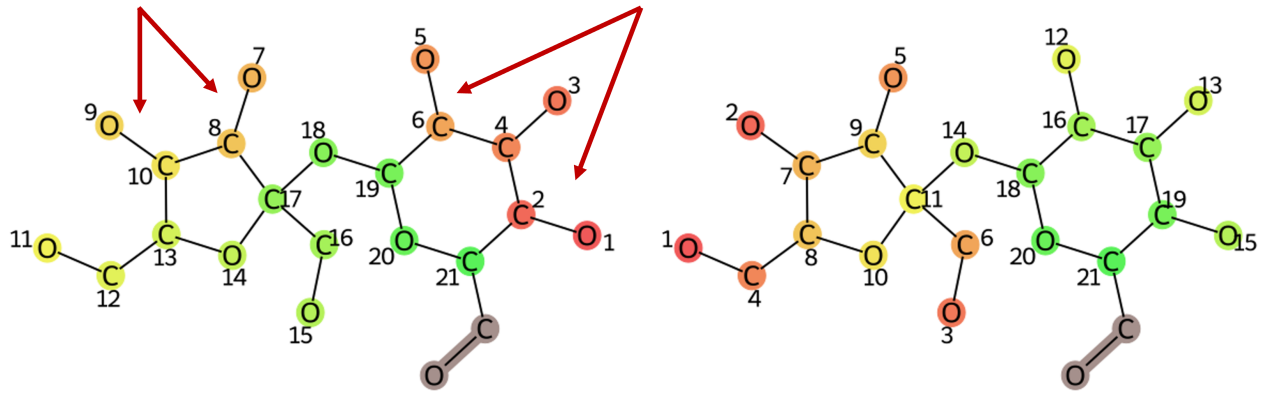


Figure 4.11: Left: DFS-algorithm; Right: BFS-algorithm; CC in dark; the red arrow indicates the undesired processing of the ring atoms in the DFS case

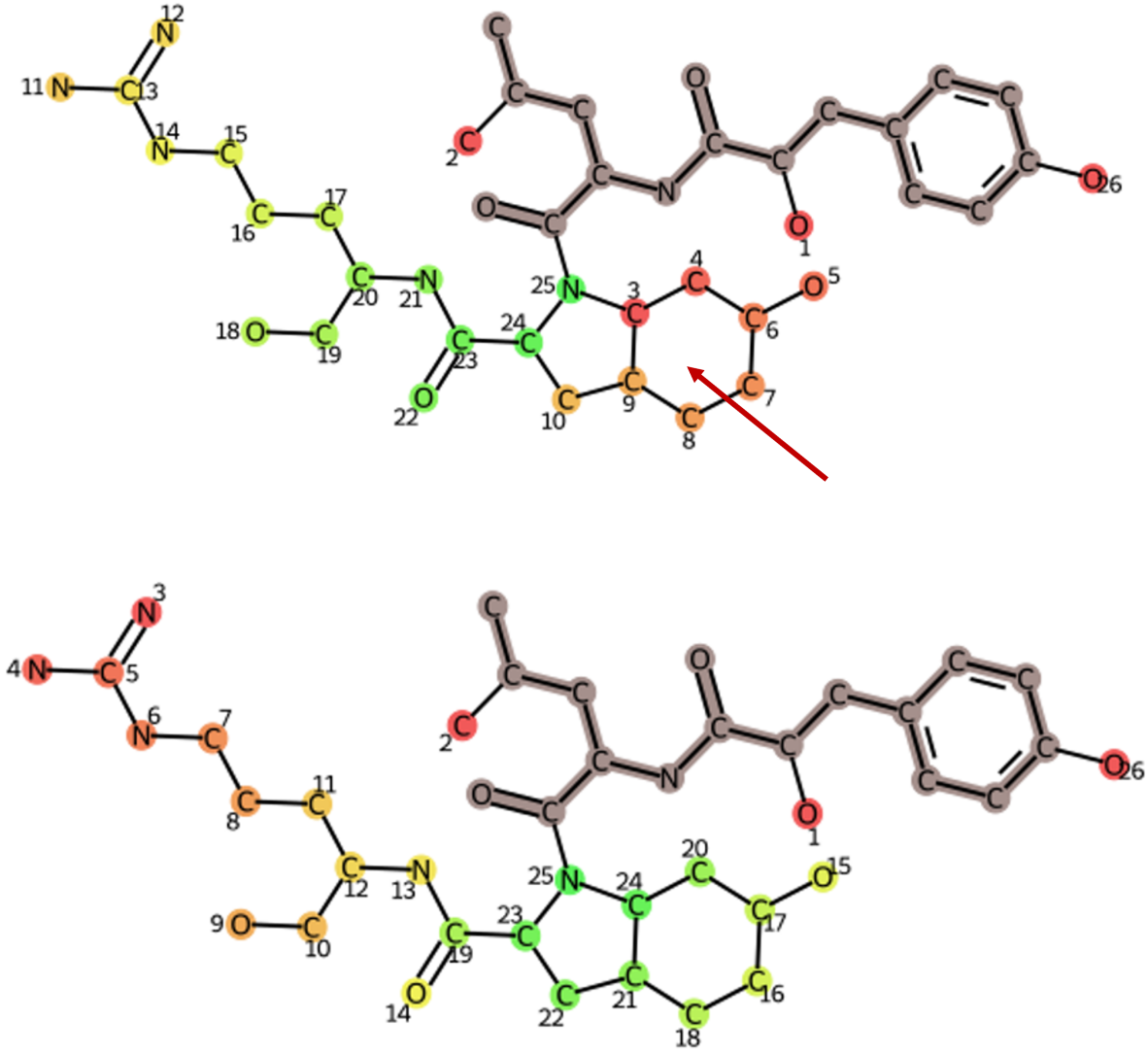


Figure 4.12: Top: DFS-algorithm; Bottom: BFS-algorithm; CC in dark; the red arrow indicates the undesired processing of the ring atoms in the DFS case

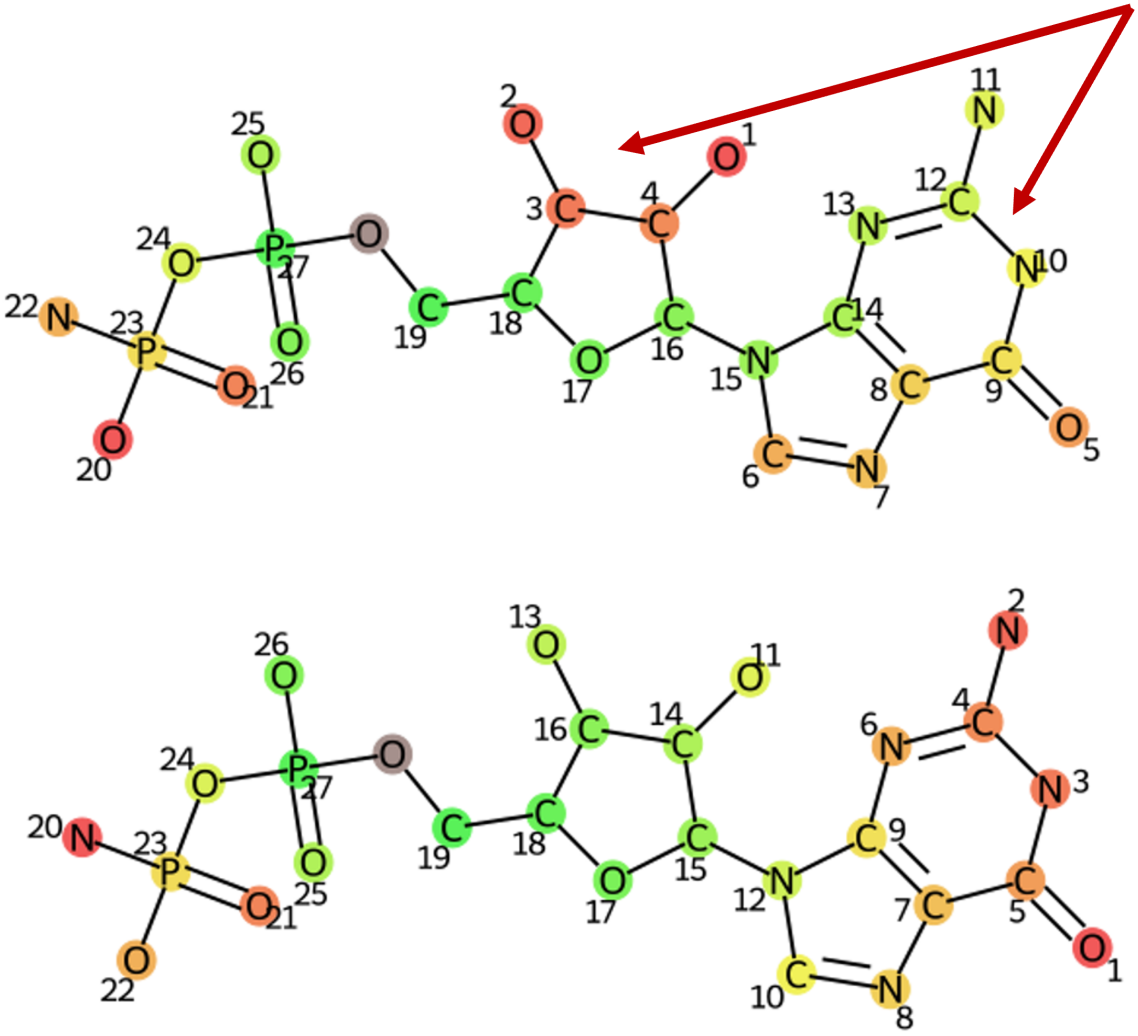


Figure 4.13: Top: DFS-algorithm; Bottom: BFS-algorithm; CC in dark; the red arrow indicates the undesired processing of the ring atoms in the DFS case

number of processed routes	dummy atoms (mean)	atoms/dummy region (mean)	number of cycles (mean)	polycyclic [%]
378	26.97	16.30	1.66	30.16

Table 5.1: Some statistics about the twenty ligands selected from the PDBbind-CN database and the calculations they were used for. Number of processed routes: total number of computed routes for a specific combination of two molecules; dummy atoms (mean): average number of total dummy atoms required for the computed mutation routes; atoms/dummy region (mean): number of total dummy atoms divided by the number of dummy regions; number of cycles (mean): average number of cyclic structural elements in all mutation routes; polycyclic: percentage of mutation routes which involve polycyclic structures, i.e., there are atoms present that participate in multiple cyclic elements

5 Results

A set of twenty ligands from the PDBbind-CN database were downloaded and used for testing CC construction, as well as mutation routes. General information about the ligands can be found in table 5.1. It should be noted, however, that the CCs for some of these ligand pairs violate the rules of `transformato` concerning a valid CC; i.e., dummy regions are connected by more than one atom to the CC (which basically implies that the atom is part of a ring structure).

In the current implementation of the mutation algorithm, this problem is solved by a helper function which chooses one of the possible connections between one of the atoms to the CC. For the following processing of the mutation algorithm, this connection is arbitrarily distinguished and the other ones removed.

5.1 Visualizations

The mutation route is visualized using a color gradient, in addition to numbering, see the earlier figures. Py3dMol [38] is used for a 3D-animation of the mutation process. Fig. 5.1 shows two molecules and their shared CC.

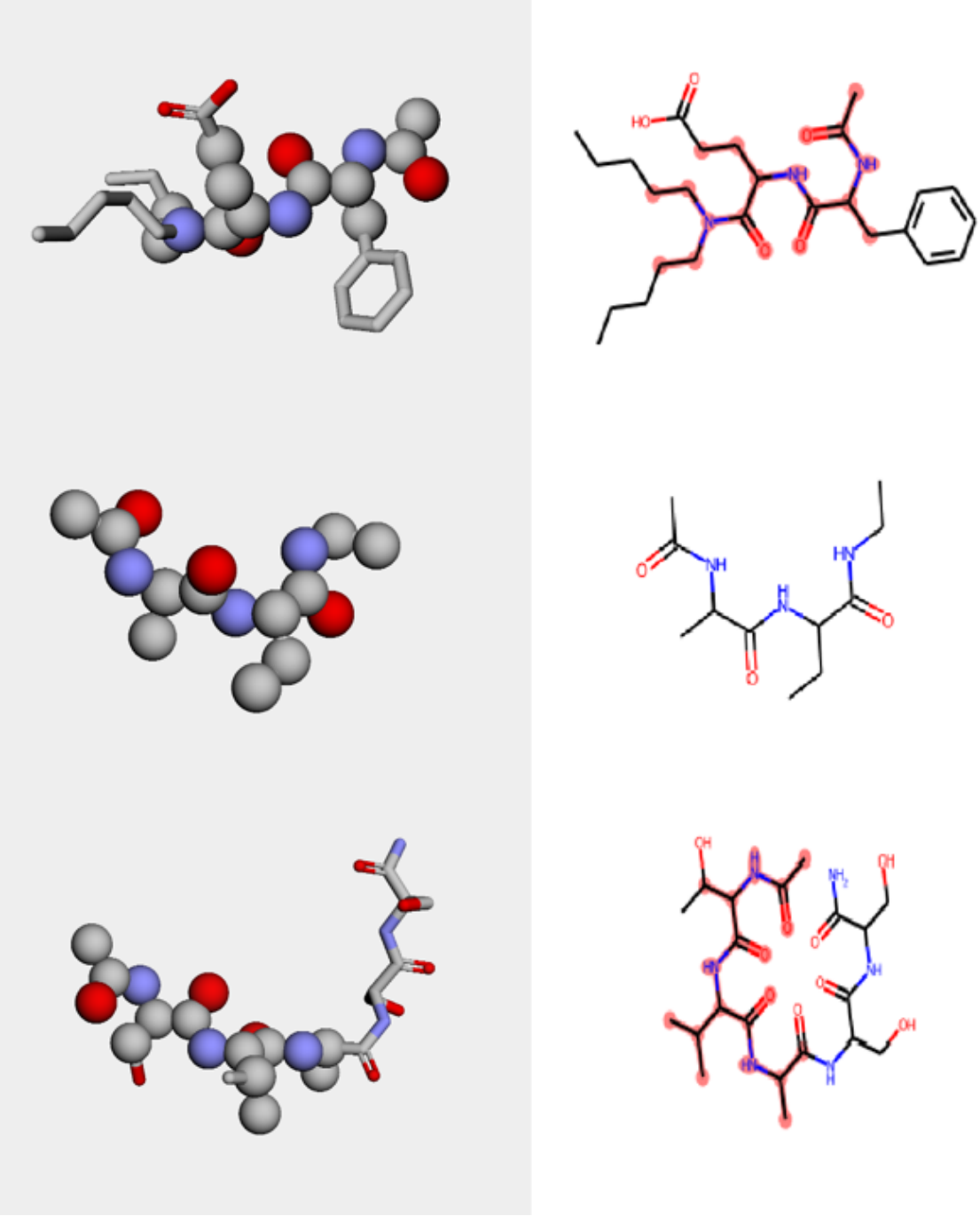


Figure 5.1: Two Visualizations of the mutation route with Py3Dmol (left column) and RDKit (right column). First and third row: Representation of the physical end states, left: spheres represent CC atoms, whereas non-CC atoms are shown in licorice representation; right: CC atoms are highlighted in red. Middle row: CC of the two end states.

5.2 Scoring schemes

Several scoring schemes have been implemented to assess and compare the mutation routes proposed by the new algorithms.
1) Betweenness centrality: Betweenness centrality measures the number of shortest paths going through a specific node [39]. More central atoms will have a higher, atoms remote from the CC will have a lower centrality coefficient. In particular, the last atom of a chain has a coefficient of 0 because no path between two other atoms visits the representing node. After each step, the node is removed from the graph. For avoiding undesired mutations, the maximum betweenness centrality of all mutation steps is more decisive; hence, the average of the mean as well as the maximum betweenness centrality of all removed nodes is shown below.

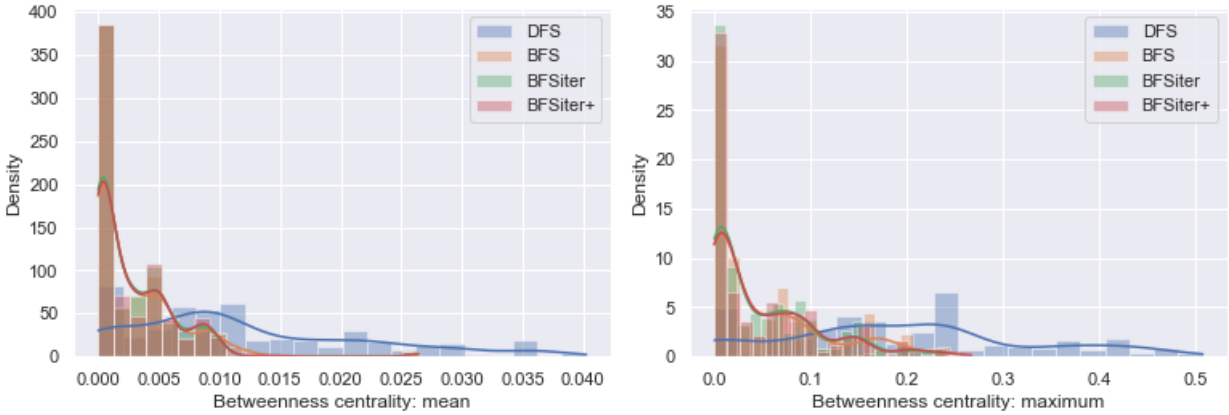


Figure 5.2: Betweenness centrality for the test ligands from the PDBbind-CN data set. Maximum and mean betweenness centrality are much higher for DFS than for all BFS-based mutation route algorithms. Curves are density estimates of the histograms.

2) Closeness centrality: Closeness centrality of a specific node is given by the inverted distances between this node and all other nodes of the graph [39]. Similar to betweenness centrality, more 'important', central atoms have a higher closeness centrality, whereas atoms more distant from the CC have a lower closeness centrality. For using this centrality measure as a scoring function for the mutation algorithms, the dummy region with the highest number of atoms (simply because this is probably the most 'interesting' one, it would also be possible to take the average of all dummy regions etc.) is selected. The closeness centrality of each node representing a dummy atom for the full graph representation, including all CC and dummy atoms, (i.e., all atoms, including already removed ones are used for calculating the statistics) is computed. For a 'good' mutation route, the atoms should be removed approximately in ascending order of their closeness centrality: The first atoms should have a high distance to most of the other atoms and consequently low closeness centrality, whereas the ones removed later should be the

more 'important' nodes with high closeness centrality. This is checked by computing Spearman's Rank correlation coefficients for each mutation route. The correlation between the order of mutation and closeness centrality is determined. A higher positive correlation coefficient shows that the closeness centrality of the atoms removed is increasing and, thus, indicates a 'better' route. The computed correlation coefficients for all mutation paths are visualized via histograms.

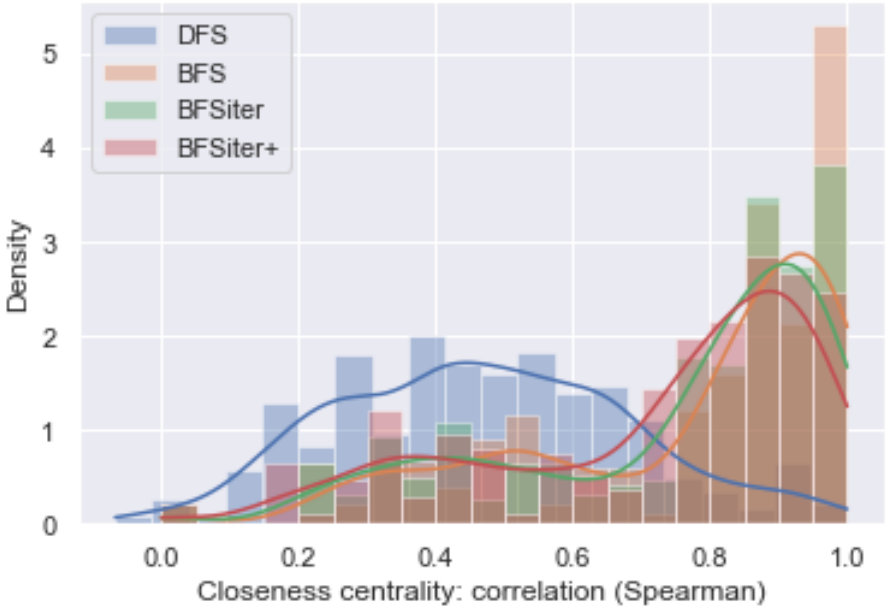


Figure 5.3: Histogram showing the distribution of the Spearman's Rank correlation coefficients of the closeness centrality values computed for the test ligands from the PDBbind-CN data set. It was checked if the closeness centrality of the removed atoms increases during mutation. A 'good' mutation route should show an increase in closeness centrality because at the beginning the atoms with a high distance from the other atoms (and hence the CC atoms) and thus a low closeness centrality are removed. When atoms are removed with ascending closeness centrality — which indicates a 'good' mutation route —, this leads to a higher positive correlation. Closeness centrality is computed for the full graph representation, including all CC and dummy atoms. Curves are density estimates of the histograms.

3) Ring-related scores: As stated above, the processing of ring structures is of crucial importance and pronounced differences between DFS and BFS occur. Four properties were calculated: The mean asymmetry at ring opening was measured: After the first atom of a ring structure is removed, usually two chains emerge. The length difference (i.e., the difference in atom number) of these two chains was measured. If both chains are equally long, the asymmetry is 0.

The 'asymmetry during ring disassembly'-score does not only evaluate the first atom removed from a ring, but checks at each mutation step involving a ring atom if asymmetric chains emerge.

The 'mean number of open rings' indicates how many ring structures are opened on average, and the 'mean processing time of rings' determines how many mutation steps it takes to process a ring completely (until only one atom of the former ring structure is present).

Even using the new algorithms, it is possible that a 'broken' ring exists for several mutation steps because atoms from other areas of the dummy region (e.g., a longer chain) are processed before the ring continues to be processed. However, in contrast to DFS, it should not happen that a ring near to the CC is opened, but some of its atoms are processed much later, and hence the mean and maximum time should be significantly shorter.

To compare the mutation algorithms, calculation of the scoring functions for the selection of ligands from the PDBbind-CN data set was performed.

In general, the computed statistics match the expectations. The range of betweenness centrality scores is much lower for BFS, suggesting that central nodes, i.e., atoms in a ring next to the common core, are processed last, whereas isolated ones, i.e., atoms at the final position of a chain, are processed preferably (fig. 5.2). Likewise, the correlation between the order of mutation of an atom and its closeness centrality scores is higher for BFS-based approaches because atoms distant from the common core are visited only at the final iterations of the algorithms and so there is a stronger positive correlation between closeness centrality and mutation order (fig. 5.3). For many mutation routes, the correlation of the BFS approaches is almost perfect, i.e., near to one, whereas for DFS it is much lower. All scoring functions show that the BFS-based algorithms tend to prune the molecule graph at positions more distant from the CC at the beginning of the processing, and rings are processed in a more systematic and 'symmetric' manner (fig. 5.4).

In the plots presenting the scoring-functions, all molecule combinations from the PDBbind-CN data set were used. Thus, the selection of molecule pairs gives rise to CCs and dummy regions with entirely different properties (e.g., number of atoms and presence of cyclic structures) and many 'trivial' structures are over-represented. It could be insightful to use only a subset (e.g., only molecules with dummy regions involving multiple ring structures or a minimum number of dummy atoms) or to try even a larger selection from the PDBbind-CN database.

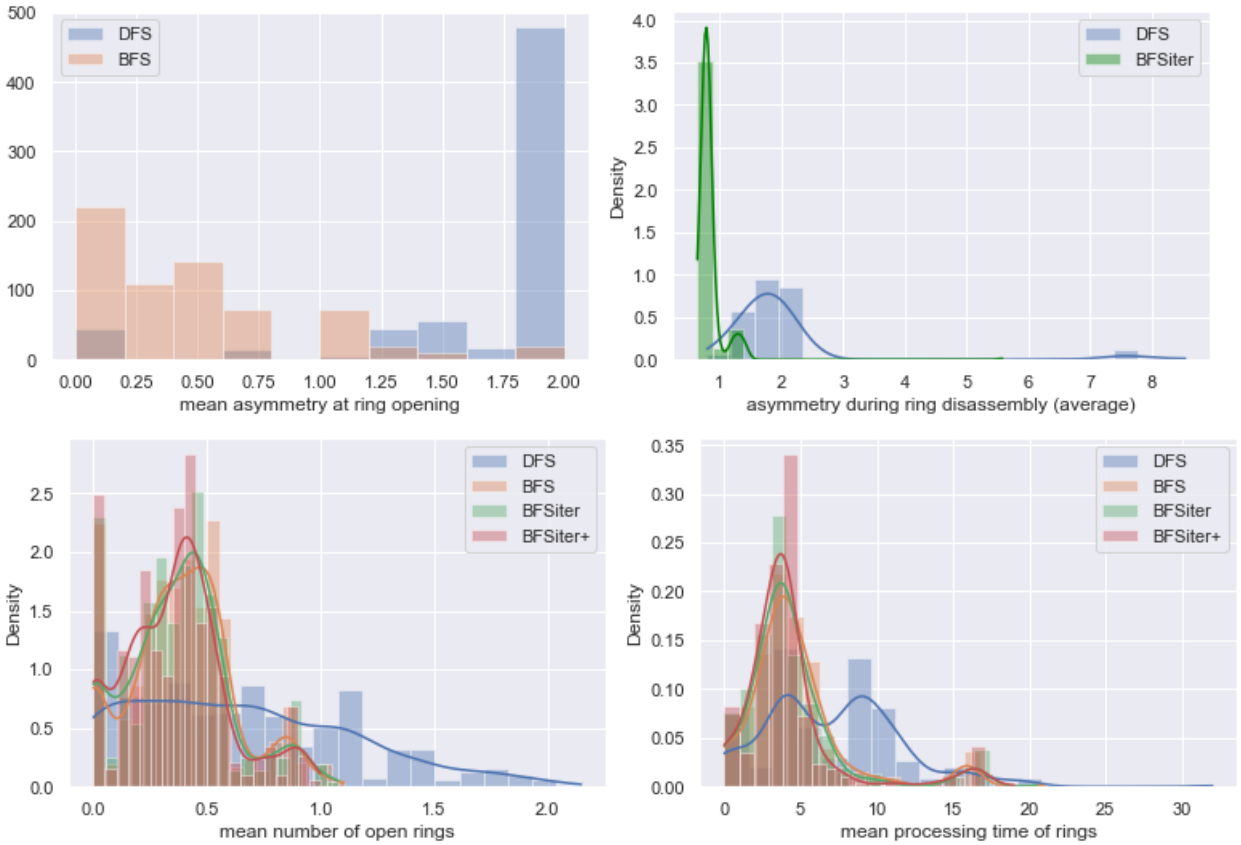


Figure 5.4: Ring-related scores for the PDBbind-CN test set. upper row: left: difference in number of atoms in the emerging chains after first removal of a ring atom, a score of 0 means symmetric processing; right: difference in number of atoms in the emerging chains after removal of a ring atom, lower score means more symmetric processing; lower row: left: mean number of open rings in the test molecules after each processing step, lower score means that rings are processed sequentially and not in parallel; right: mean number of mutation steps until a ring is totally processed. Curves are density estimations of the emerging distributions.

5.3 Results for selected molecule pairs

For a selection of small molecules taken from [4, 40], the relative solvation free energy differences were calculated using `transformato` with the old and the new mutation route algorithms. In particular, we studied the three pairs toluene/methane, 2-methylfuran/methane, and 2-methylindole/methane, as well as the solvation free energy difference between 2-cyclopentyl-indole and 7-cyclopentyl-indole (2-/7-CPI). For the last case, the free energy differences were recomputed only for the new algorithms; however, earlier results using the old algorithm were used for comparison. This is one of the most interesting examples because the old CC generation, which searched for the CC including hydrogens without the improvements reported in 'Processing of hydrogen atoms', generated a smaller CC (fig. 5.5). Thus, one would expect that for this example, differences should be especially pronounced.

Although these molecules are rather small and simple, they encompass some of the most interesting features, like rings. For instance, the mutation route for toluene is fundamentally different depending on the algorithm: the old algorithm starts next to the atom of the phenyl group that is connected to the methyl substituent — which serves as the CC — and processes the rest of the atoms in a chain-like manner. By contrast, the new one starts at the atom with maximum distance from the substituent and proceeds symmetrically until the CC is reached. An overview of the molecules and the corresponding mutation routes is shown in figs. 5.10 and 5.11. The simulation results are averaged over four runs (except 2-CPI/7-CPI, for which only three runs were performed). For all these examples, the standard deviation is smaller using the new route finding algorithm and adapted settings for the generation of the CC (fig. 5.6 and table 5.2).

For the 2-/7-CPI-transformation, a relative free energy difference of -1.55 ± 0.10 kcal/mol is computed using the CC and the route proposed by the new algorithms. In [1], for this transformation, $-1.43\pm$ 0.30 kcal/mol was determined with the smaller cyclopentane-X CC. It can be assumed that the differences between the old and new mutation route are even more pronounced because in [1] the calculations were repeated five times and averaged, in contrast to only three replicates for the run with the new mutation route. Of course, a direct comparison with the same number of replicates would be advantageous to quantify the improvement, but in any case, the change in standard deviation is remarkable. Calculation of the absolute free energy differences of both molecules yield -1.58 ± 0.30 kcal/mol. This indicates that the new route not only provides a smaller error, but also leads to a more accurate result.

However, probably the greatest advantage is that the mutation route for the new, bigger

mutation partners	old algorithm (DFS)	new algorithm (BFS)
toluene/methane	2.02 ± 0.21	2.05 ± 0.04
2-methylfuran/methane	1.47 ± 0.24	1.60 ± 0.16
2-methyl-1H-indole/methane	7.85 ± 0.23	8.20 ± 0.13

Table 5.2: results for a selection of mutation partners

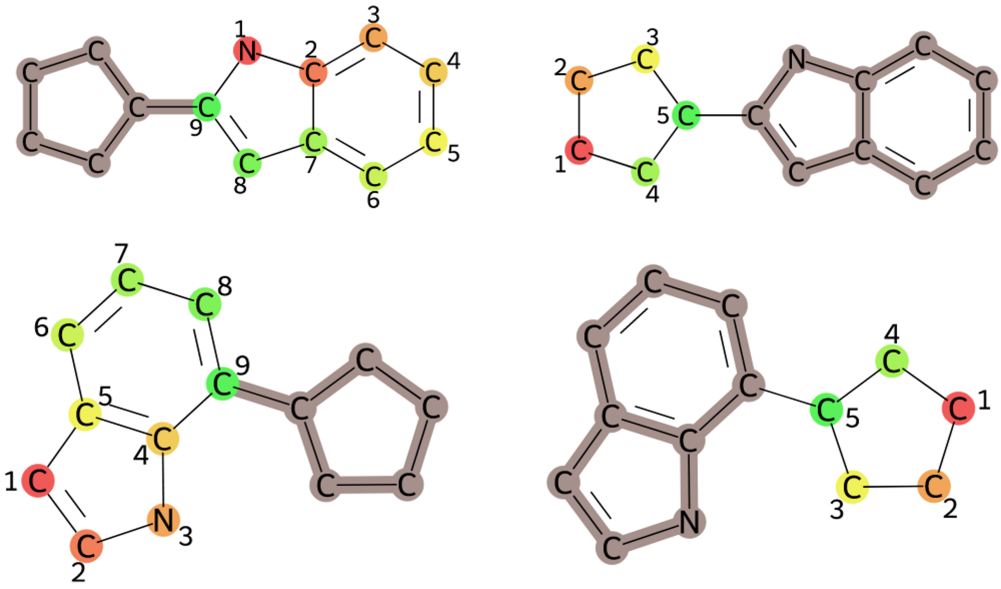


Figure 5.5: mutation routes for 2-cyclopentylindole (upper row)/7-cyclopentylindole (lower row); left: small CC with DFS-algorithm; right: bigger CC with BFS-algorithm; CC in dark; the smaller CC is obtained when hydrogens are not removed before the computation of the maximum common substructure (it should be noted that in this case even further manual post-processing is necessary because one atom of the indole is also attached to the CC)

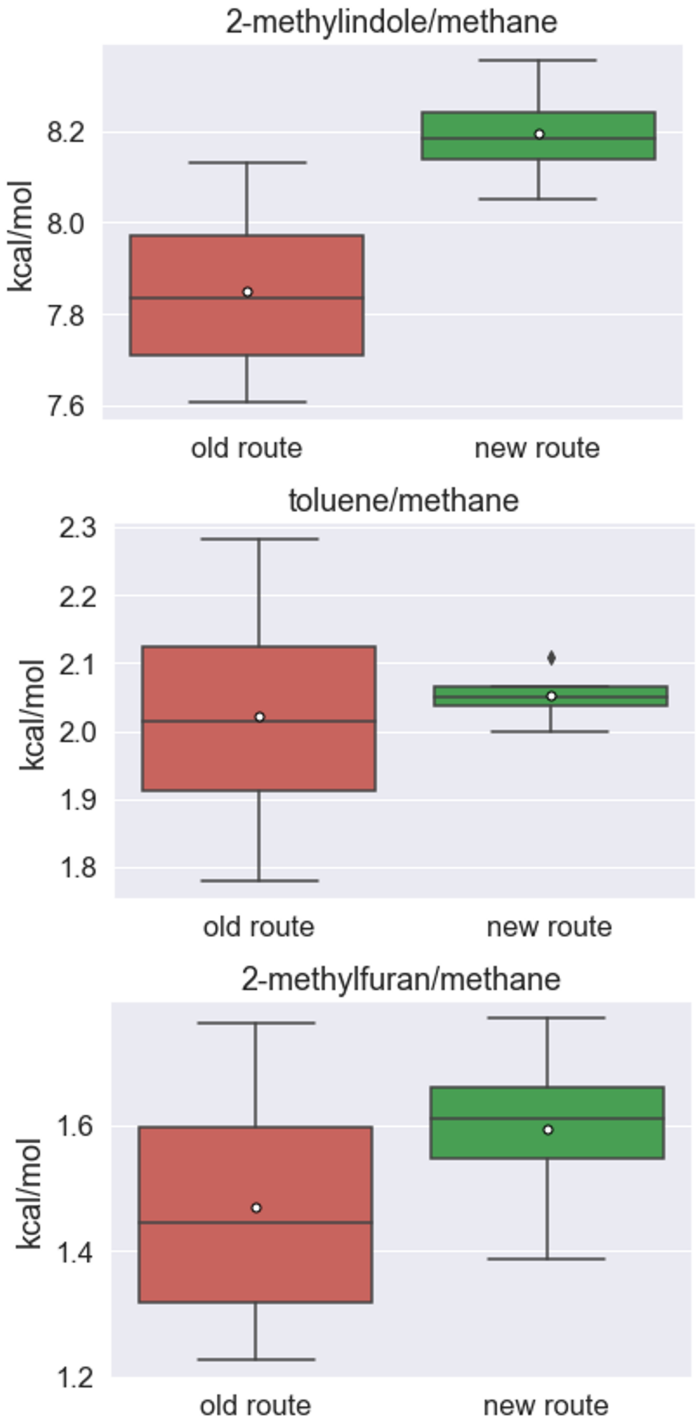
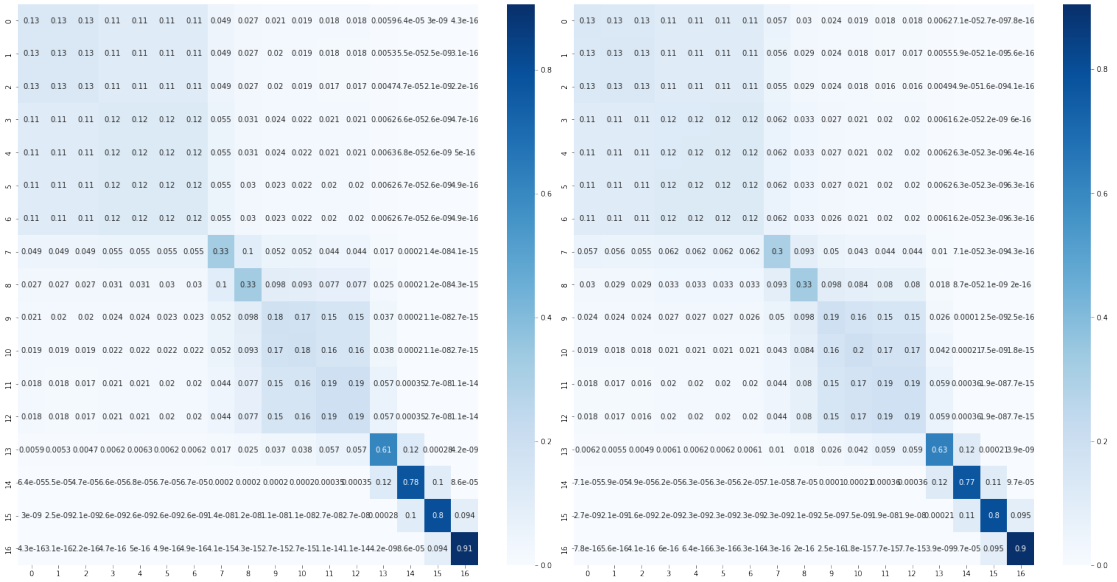


Figure 5.6: comparison of results for mutating 2-methylindole, toluene and 2-methylfuran to methane.

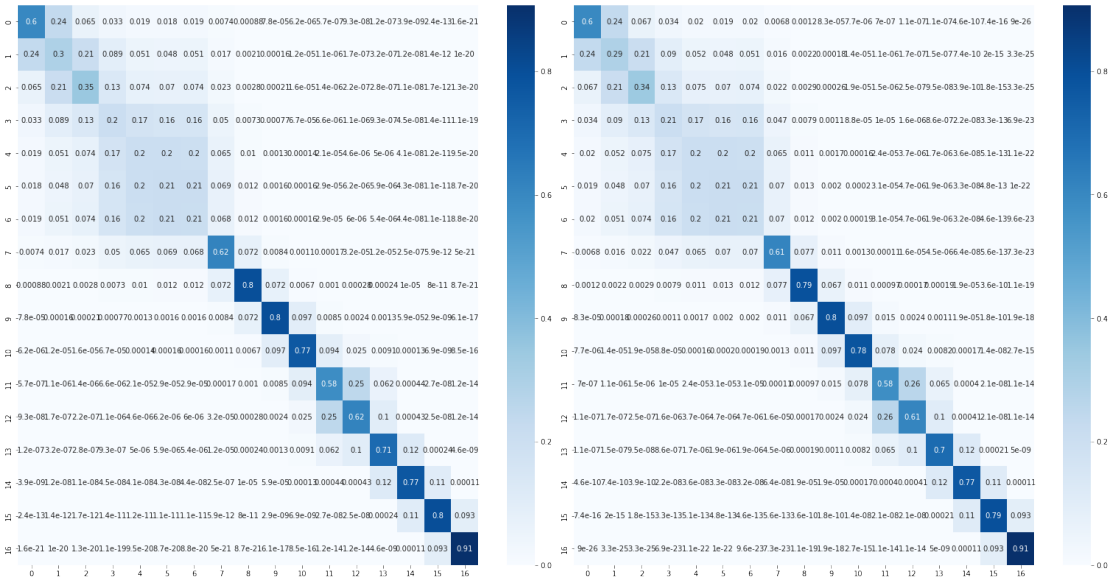
CC needs fewer states (only five in contrast to nine heavy atoms have to be mutated).

An additional means for detecting differences between the outcome of the mutation algorithms is to compare runs of different sampling length. The results of the MD runs set up with `transformato` can be evaluated using the functions of the MBAR class of `pymbar` [24]. Python scripts were written to evaluate the computed free energy differences for different simulation lengths. There are two crucial parameters: the reduced potential energy of an uncorrelated configuration n at a specific state k (`u_kn`) and the number of uncorrelated snapshots n (`N_k`). By removing the same number of configurations at each state k and adjusting (`N_k`) accordingly, shorter simulations were generated artificially. In 5.12, 5.13 and 5.14, a comparison between old and new route for molecule pairs consisting of toluene, 2-methylfuran, 2-methyl-1H-indole and methane is presented. The mean of the calculated free energy differences as well as the standard deviation is shown. In 5.15, free energy differences for the 2-CPI-mutations at the two conditions (water box and vacuum) are visualized. As expected, for longer simulation lengths the standard deviation decreases, whereas very short simulation lengths (i.e., a very low number of configuration snapshots as input for the MBAR computations using `pymbar`) lead to unreliable results. However, looking at the evolution of the free energy difference mean value and standard deviation, it is difficult to confirm the superiority of one of the mutation routes for these three transformations or to indicate a sufficient minimum simulation length. Furthermore, the `pymbar`-package [24] allows the computation of overlap plots. Figs. 5.7, 5.8, and 5.9 show overlap plots and the change of free energy difference of one run between the states for toluene $\rightarrow$ methane and overlap plots for 2-methyl-1H-indole. In the case of 2-methyl-1H-indole (the mutation involves processing of a double ring), significant differences for the water box between the old and new algorithm are discernible.



(a) toluene/vacuum old

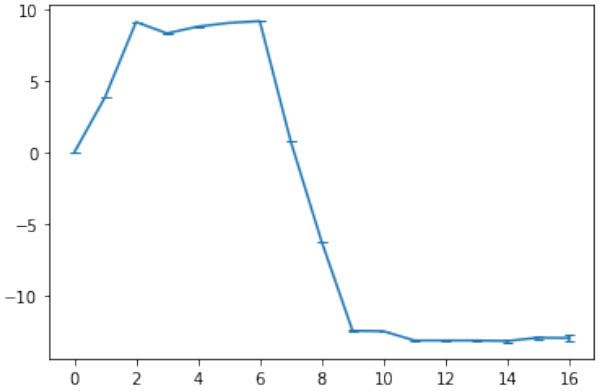
(b) toluene/vacuum new



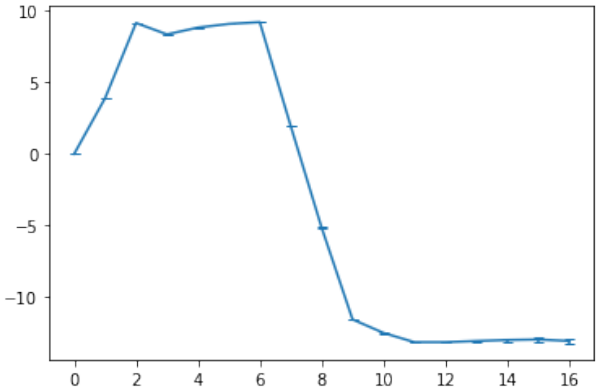
(c) toluene/waterbox old

(d) toluene/waterbox new

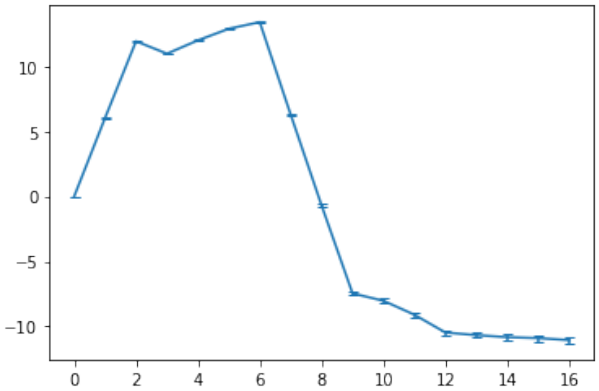
Figure 5.7: Overlap plots for toluene → methane: upper row: vacuum, lower row: water box; left: old mutation algorithm, right: new mutation algorithm



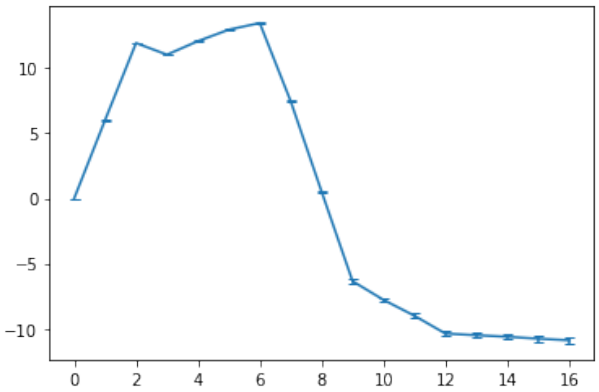
(a) toluene/vacuum old



(b) toluene/vacuum new

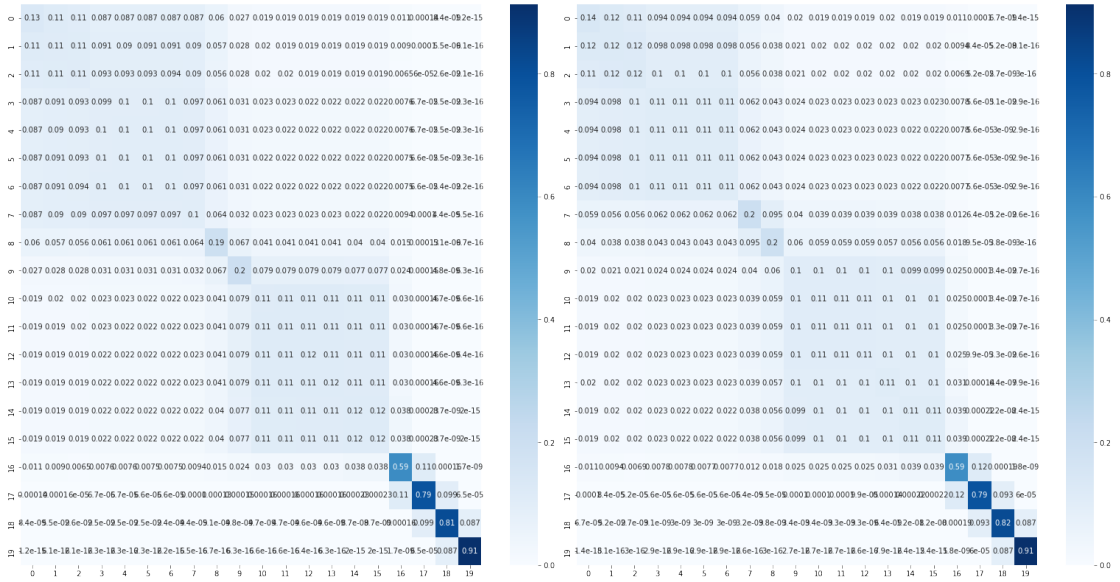


(c) toluene/waterbox old



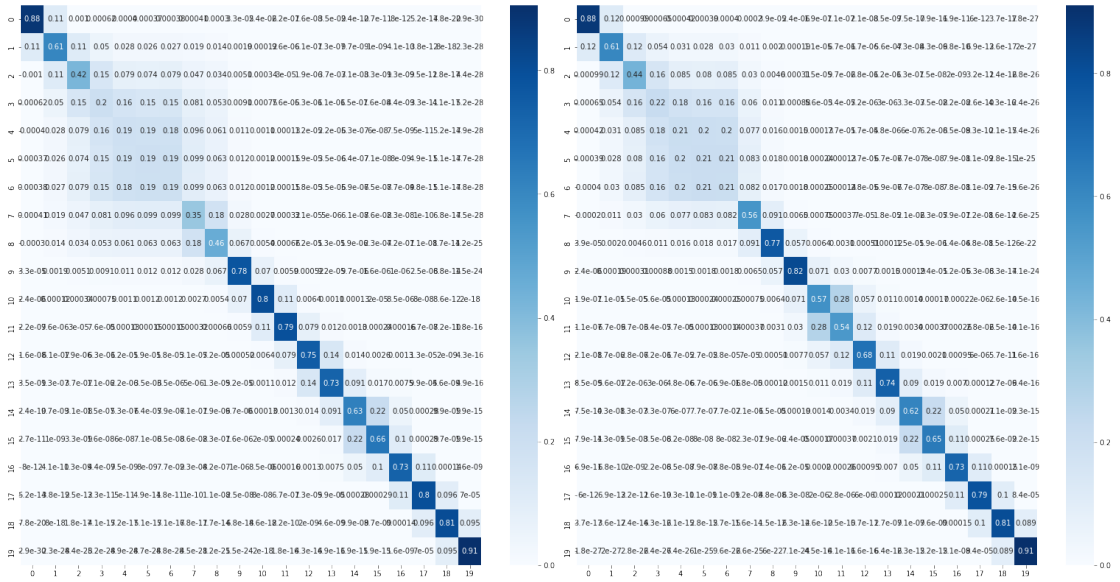
(d) toluene/waterbox new

Figure 5.8: free energy differences per state for toluene → methane



(a) toluene/vacuum old

(b) toluene/vacuum new



(c) toluene/waterbox old

(d) toluene/waterbox new

Figure 5.9: Overlap plots for 2-methyl-1H-indole → methane: upper row: vacuum, lower row: water box; left: old mutation algorithm, right: new mutation algorithm

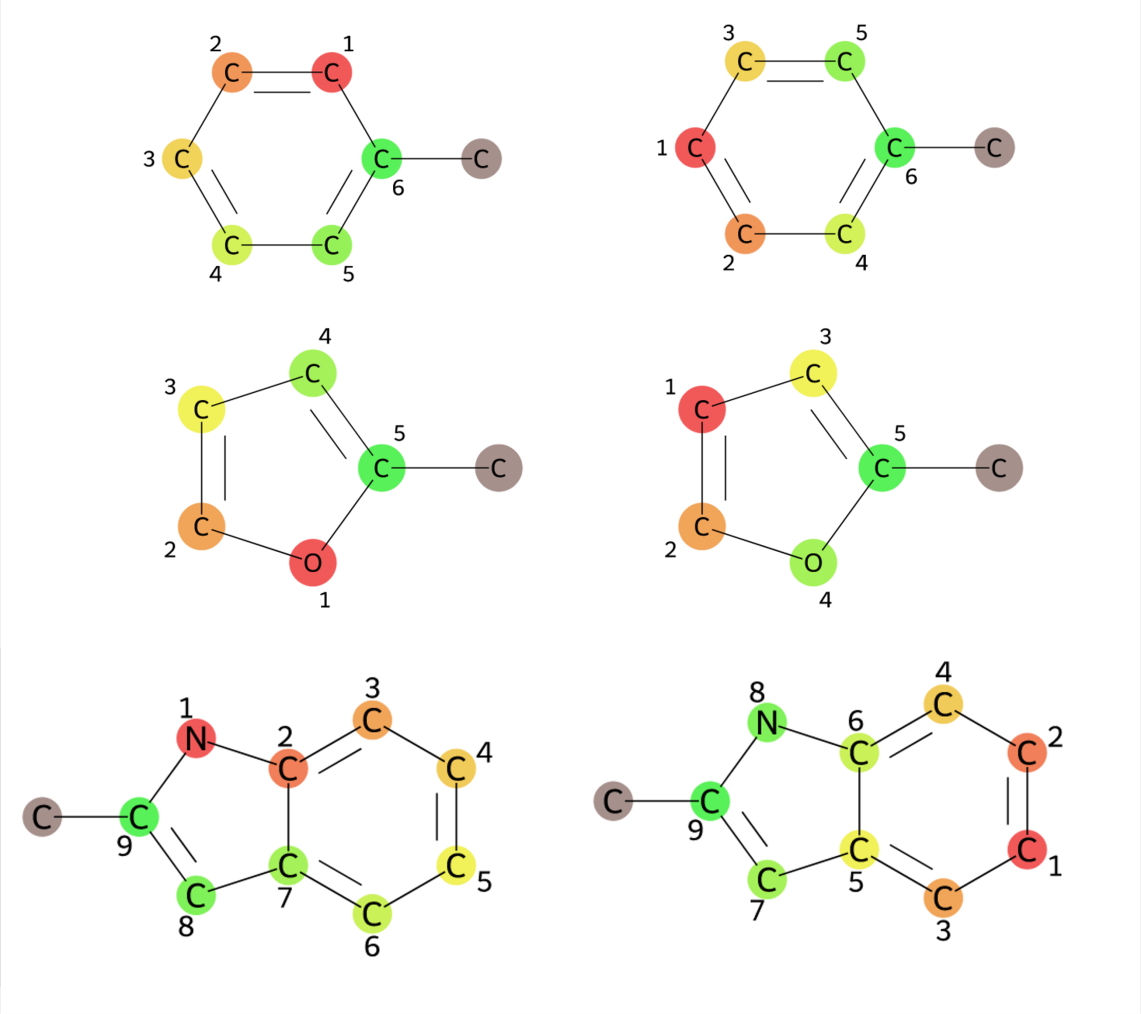


Figure 5.10: left: DFS-algorithm; right: BFS-algorithm; CC in dark; from top to bottom row: mutation routes for toluene/methane, 2-methylfuran/methane, 2-methylindole/methane

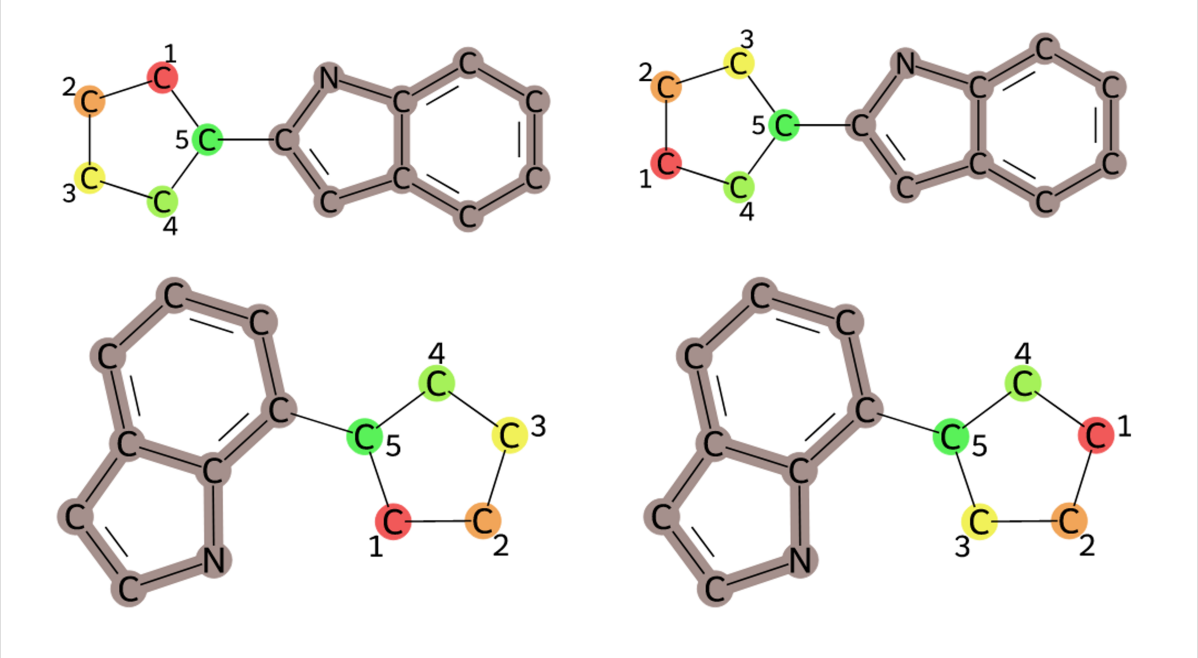


Figure 5.11: left: DFS-algorithm; right: BFS-algorithm; CC in dark; mutation routes for 2-cyclopentylindole/7-cyclopentylindole

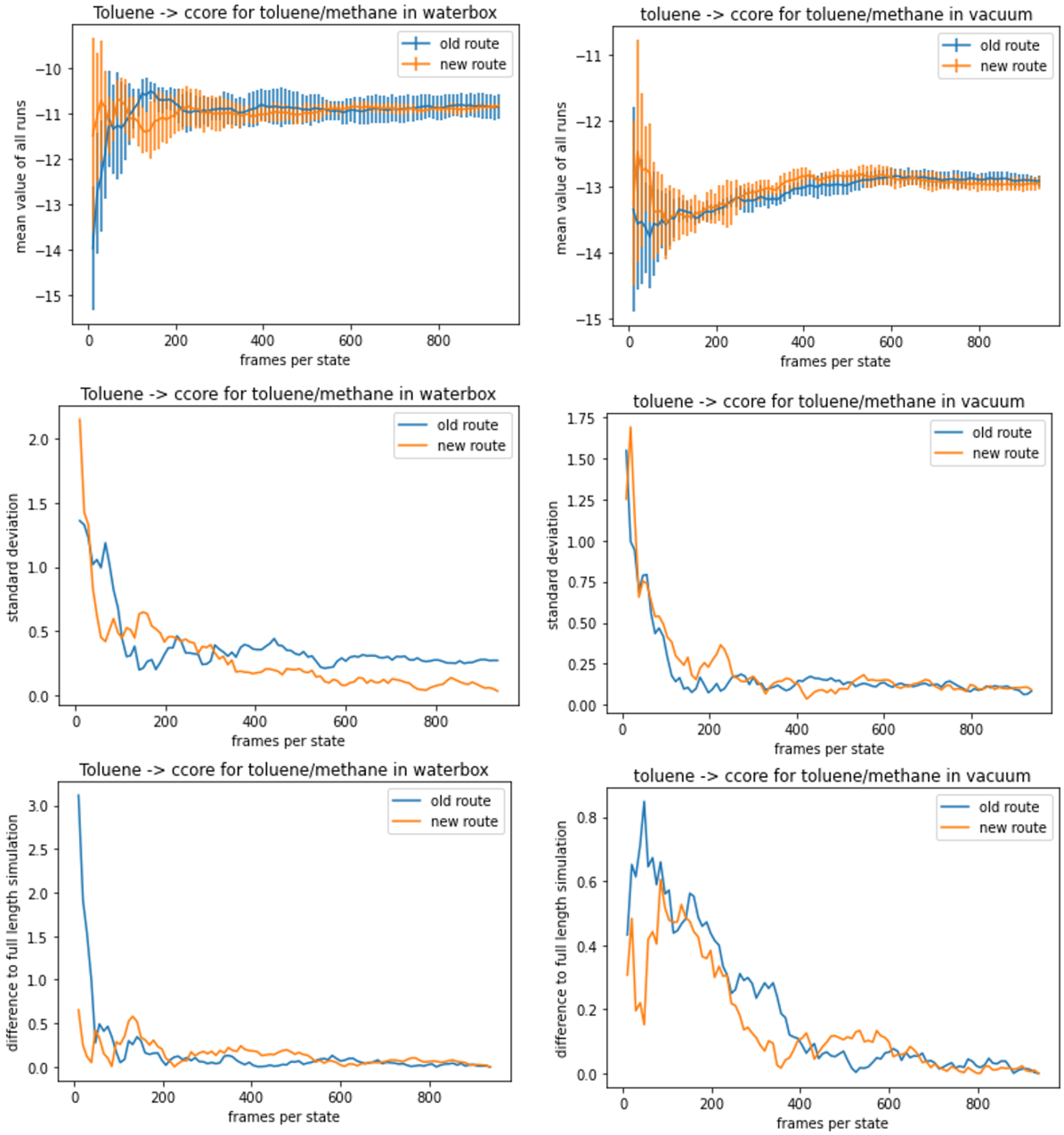


Figure 5.12: toluene → methane; left: water box; right: vacuum; mutation routes for toluene/methane; first row: mean value, bars indicate standard deviation; middle row: standard deviation; third row: difference to full-length simulation (i.e., the last value is zero)

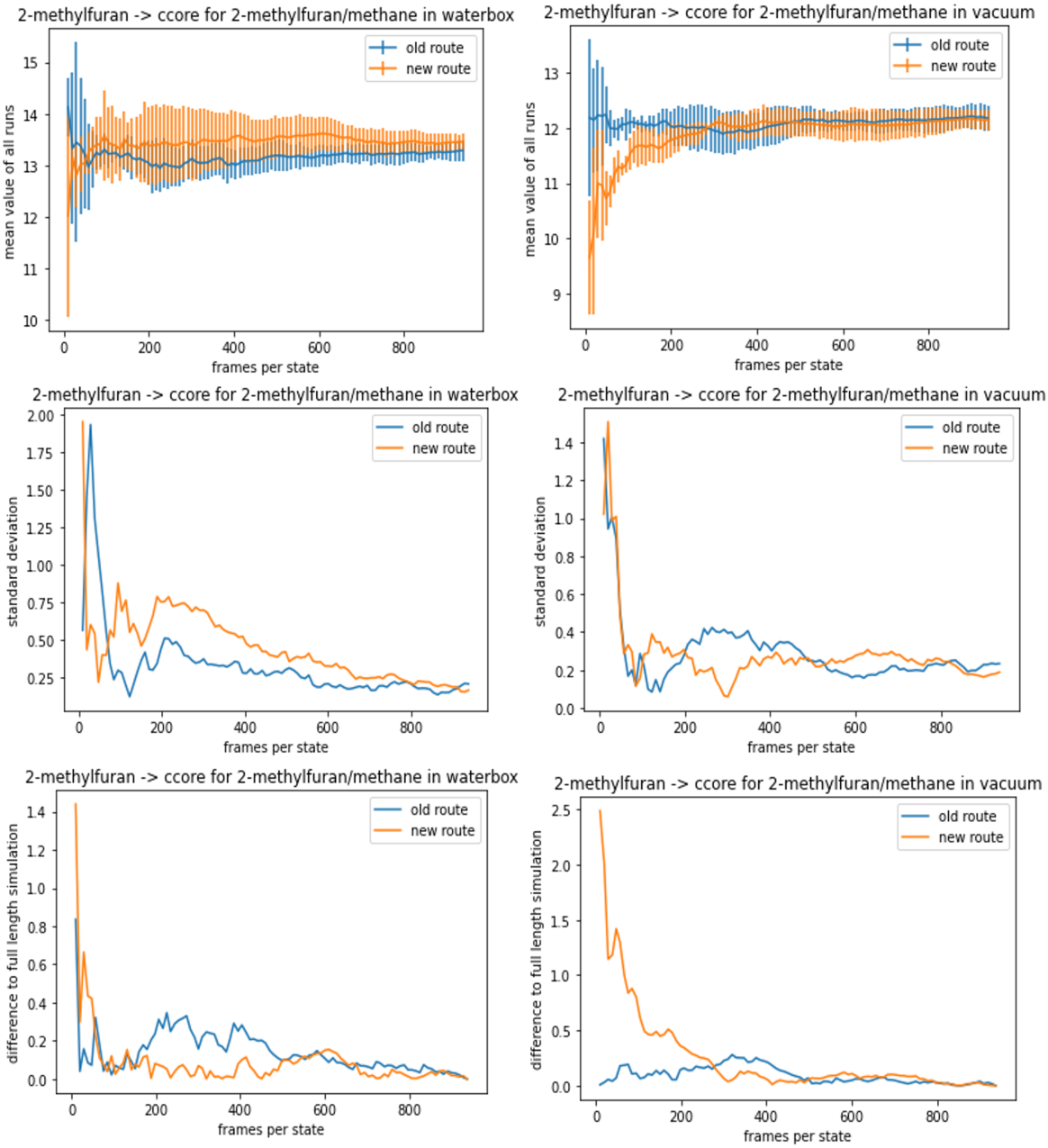


Figure 5.13: 2-methylfuran → methane; left: water box; right: vacuum; mutation routes for 2-methylfuran/methane; first row: mean value, bars indicate standard deviation; middle row: standard deviation; third row: difference to full-length simulation (i.e., the last value is zero)

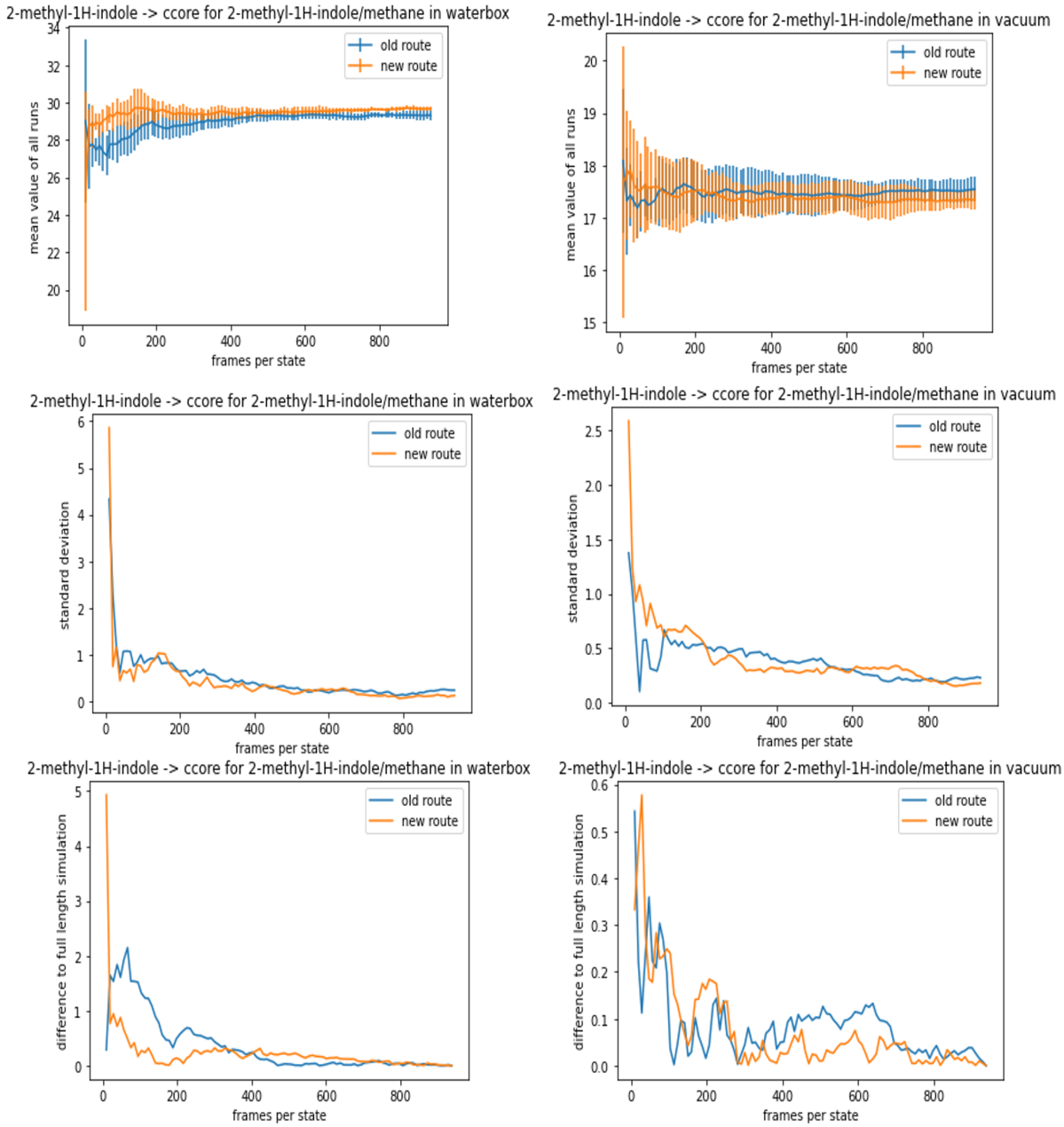


Figure 5.14: 2-methylindole → methane; left: water box; right: vacuum; mutation routes for 2-methylindole/methane; first row: mean value, bars indicate standard deviation; middle row: standard deviation; third row: difference to full-length simulation (i.e., the last value is zero)

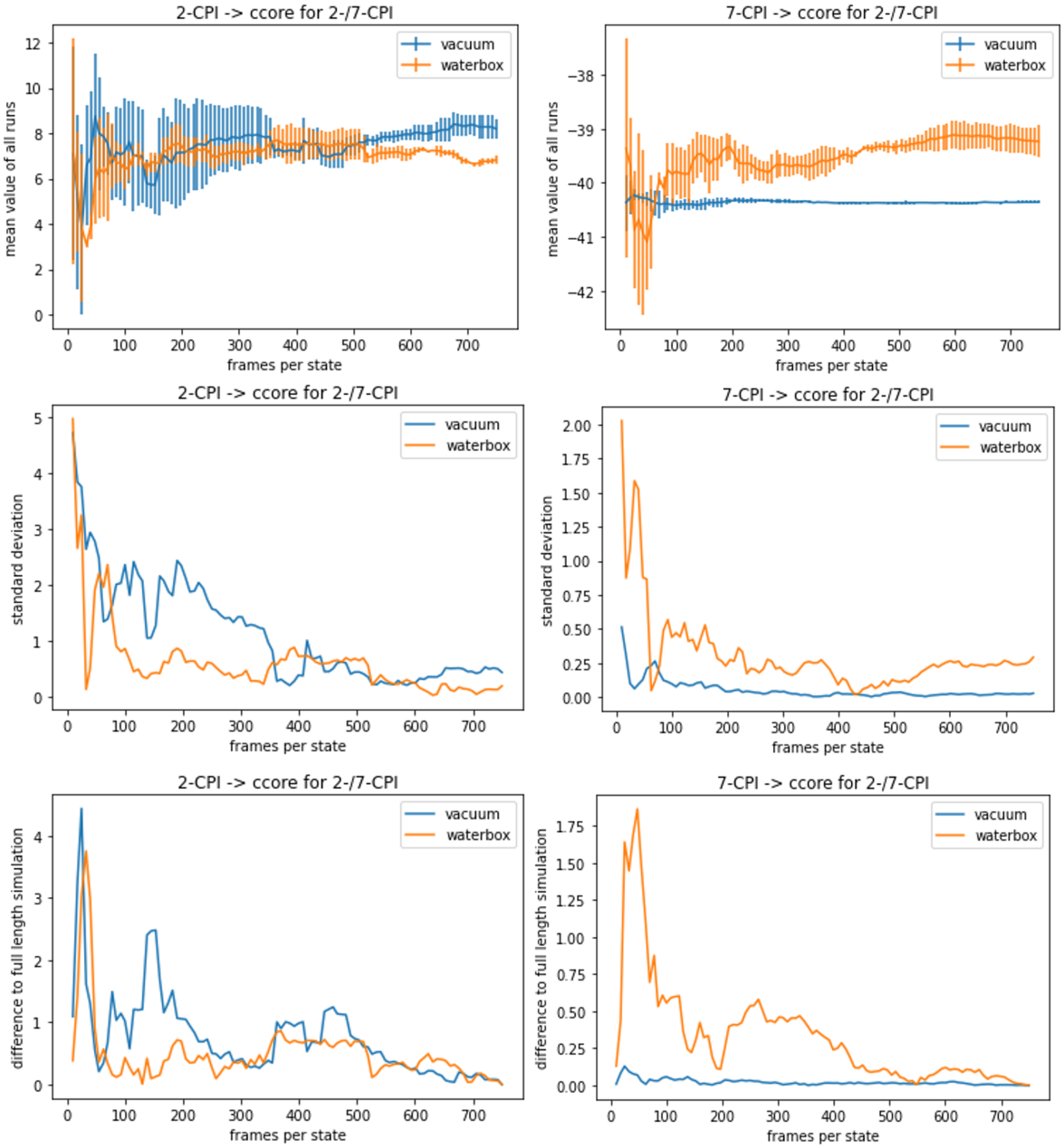


Figure 5.15: left: 2-CPI → CC for 2-/7-CPI; left: 7-CPI → CC for 2-/7-CPI; first row: mean value, bars indicate standard deviation; middle row: standard deviation; third row: difference to full-length simulation (i.e., the last value is zero)

6 Conclusion

Simulation results for the molecule pairs used in [4, 40] demonstrate the superiority of the new mutation algorithms. In each case, the standard deviation of the new algorithms is smaller and for 2-/7-CPI the free energy difference is closer to the result of absolute free energy calculations. In this report, only tests on rather small molecules and simple transformations have been carried out. Additional comparisons of mutation routes for more molecules and closer examination of the obtained results are necessary.

Especially for more complex transformations, an evaluation of the relation between sampling length and standard deviation for different mutation routes could be insightful. In such cases, differences between protocols might be more pronounced, and choosing an algorithm which allows for shorter sampling length by minimizing variance would lower the computational cost.

To investigate such correlations systematically, one could apply the mutation algorithms to a series of molecules with exactly defined, small modifications (for example, one added/removed atom or ring structure). This would allow one to probe if the size of the dummy regions or the CC and the combination of some elements, e.g., rings, have an effect. However, it should be stressed that in most cases the effects probably will be moderate and smaller than the estimated standard error of the simulations.

The tf-routes package presented in this work also allows processing heavy non-carbon atoms with specific priorities; i.e., atoms of different types can be represented as nodes with different weight, which impacts the mutation route. It would be interesting if there are more pronounced differences for molecules with such atoms.

However, in any case, more results for molecules with different structure (e.g., different size and number of rings and chains, possible different atoms systems) and different sampling length would be necessary. The molecule pairs assessed so far are rather small and do not cover the space of possible structures at all.

Elucidating the apparently more complex and specific role between mutation order and resulting free energy differences could allow further improvement of the mutation route creation (given that the overall differences suggest that further optimization is expedient).

The main contribution of this work is the adjustment of the CC generation and the implementation of new mutation algorithms. The main steps of the workflow are outlined in fig. 6.1. The red arrows indicate steps which are indispensable for the correct processing of hydrogens and are of particular importance for molecule pairs with terminal junction 'X'-atoms at different positions as in the shown 2-CPI/7-CPI transformation.

The greatest advantage of the new algorithms in comparison to the old versions is certainly that they always should yield a correct CC and a 'reasonable' route for every molecule pair (if a correct CC exists). In particular, the lack of special treatment of CC hydrogen atoms produced faulty common cores in previous versions of `transformato`.

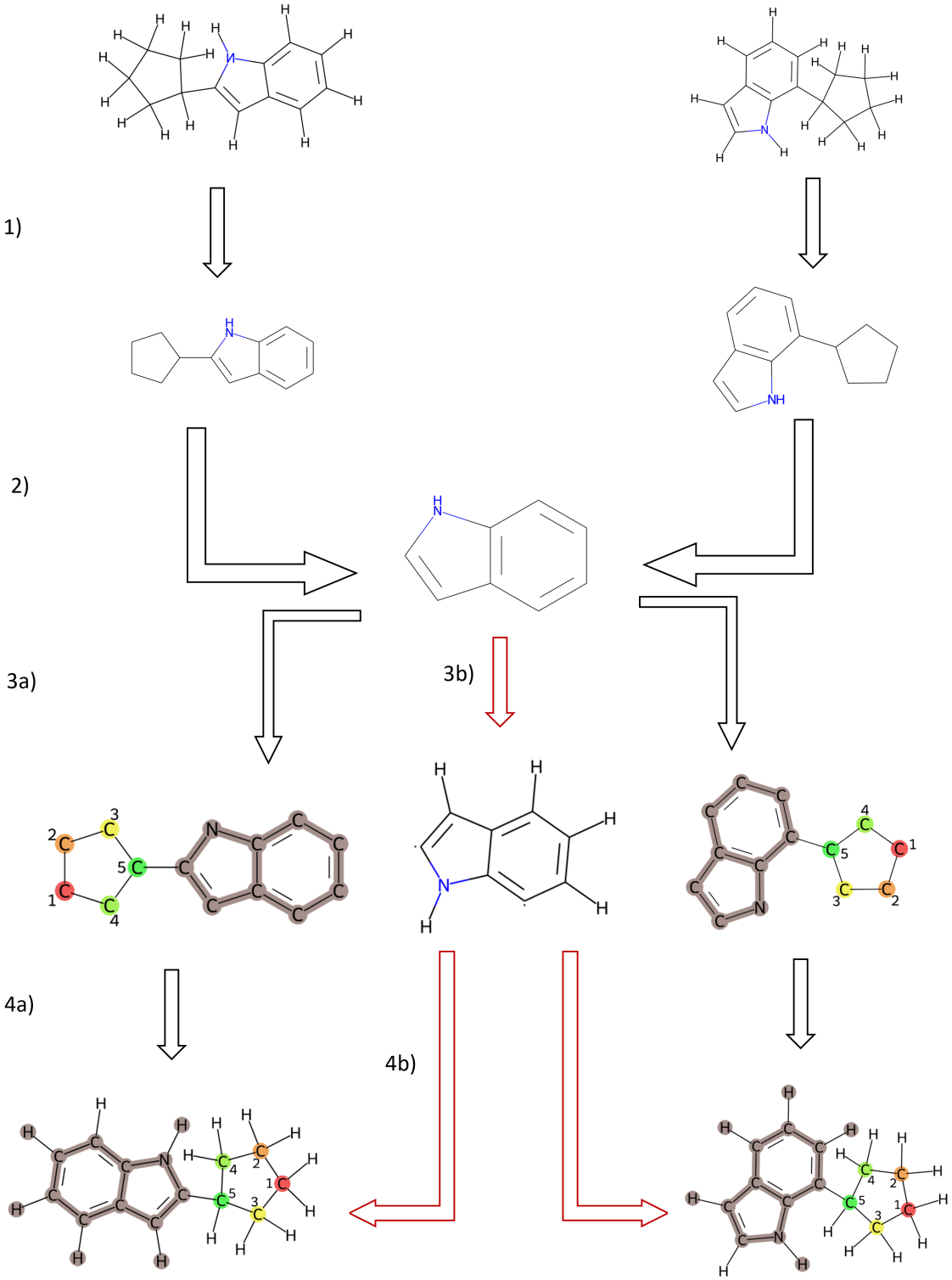


Figure 6.1: Main steps of processing a molecule pair: 1) remove hydrogens, but store a copy of the original molecule objects so that the information about the position and RDKit id of the hydrogens does not get lost 2) use representation without hydrogens to extract maximum common substructure according to `transformato` constraints 3) Create a mutation route for each molecule 4) add hydrogens again so that the molecules can be processed further in the `transformato` workflow 3b) the CC is endowed with hydrogens at the correct position using the information from the original molecule objects 4b) the CC with hydrogens at specific positions is mapped on both molecules to create the final representation

The new processing always generates a valid CC with the maximal number of heavy atoms. Furthermore, the old mutation route algorithm often led, especially in the case of bigger molecules, to atom removals in regions near the CC before chains etc. were systematically processed. Now both parts of the workflow, CC generation and mutation route determination, are adjusted and should always yield reasonable and efficient mutation routes.

List of Figures

Bibliography

[1] Markus Fleck, Marcus Wieder, and Stefan Boresch. Dummy atoms in alchemical free energy calculations. *Journal of chemical theory and computation*, 17(7):4403–4419, 2021.

[2] Johannes Karwounopoulos, Marcus Wieder, and Stefan Boresch. Relative binding free energy calculations with transformato: A molecular dynamics engine-independent tool. *Frontiers in molecular biosciences*, 9:954638, 2022.

[3] https://github.com/wiederm/transformato.

[4] Marcus Wieder, Markus Fleck, Benedict Braunsfeld, and Stefan Boresch. Alchemical free energy simulations without speed limits. a generic framework to calculate free energy differences independent of the underlying molecular dynamics program. *Journal of computational chemistry*, 43(17):1151–1160, 2022.

[5] Edward King, Erick Aitchison, Han Li, and Ray Luo. Recent developments in free energy calculations for drug discovery. *Frontiers in molecular biosciences*, 8:712085, 2021.

[6] Zoe Cournia, Bryce Allen, and Woody Sherman. Relative binding free energy calculations in drug discovery: Recent advances and practical considerations. *Journal of chemical information and modeling*, 57(12):2911–2937, 2017.

[7] David A. Case, Thomas E. Cheatham, Tom Darden, Holger Gohlke, Ray Luo, Kenneth M. Merz, Alexey Onufriev, Carlos Simmerling, Bing Wang, and Robert J. Woods. The amber biomolecular simulation programs. *Journal of computational chemistry*, 26(16):1668–1688, 2005.

[8] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. Charmm: the biomolecular simulation program. *Journal of computational chemistry*, 30(10):1545–1614, 2009.

[9] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C. Smith, Berk Hess, and Erik Lindahl. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1-2:19–25, 2015.

[10] Stefan Boresch, Franz Tettinger, Martin Leitgeb, and Martin Karplus. Absolute binding free energies: A quantitative approach for their calculation. *The Journal of Physical Chemistry B*, 107(35):9535–9551, 2003.

[11] William L. Jorgensen, J. Kathleen Buckner, Stephane Boudon, and Julian Tirado-Rives. Efficient computation of absolute free energies of binding by computer simulations. application to the methane dimer in water. *The Journal of Chemical Physics*, 89(6):3742–3746, 1988.

[12] Peter Kollman. Free energy calculations: Applications to chemical and biochemical phenomena. *Chem. Rev.*, (93):2395–2417, 1993.

[13] Michael R. Shirts and David L. Mobley. An introduction to best practices in free energy calculations. *Methods in molecular biology (Clifton, N.J.)*, 924:271–311, 2013.

[14] Antonia S. J. S. Mey, Bryce Allen, Hannah E. Bruce Macdonald, John D. Chodera, Maximilian Kuhn, Julien Michel, David L. Mobley, Levi N. Naden, Samarjeet Prasad, Andrea Rizzi, Jenke Scheen, Michael R. Shirts, Gary Tresadern, and Huafeng Xu. Best practices for alchemical free energy calculations. *Living Journal of Computational Molecular Science*, 2(1), 2020.

[15] Stefan Bruckner and Stefan Boresch. Efficiency of alchemical free energy simulations. i. a practical comparison of the exponential formula, thermodynamic integration, and bennett's acceptance ratio method. *Journal of computational chemistry*, 32(7):1303–1319, 2011.

[16] Anita de Ruiter, Stefan Boresch, and Chris Oostenbrink. Comparison of thermodynamic integration and bennett acceptance ratio for calculating relative protein-ligand binding free energies. *Journal of computational chemistry*, 34(12):1024–1034, 2013.

[17] John G. Kirkwood. Statistical mechanics of fluid mixtures. *The Journal of Chemical Physics*, 3(5):300–313, 1935.

[18] Stefan Bruckner and Stefan Boresch. Efficiency of alchemical free energy simulations. ii. improvements for thermodynamic integration. *Journal of computational chemistry*, 32(7):1320–1333, 2011.

[19] Robert W. Zwanzig. High–temperature equation of state by a perturbation method. i. nonpolar gases. *The Journal of Chemical Physics*, 22(8):1420–1426, 1954.

[20] Vytautas Gapsys, Servaas Michielssens, Jan Henning Peters, Bert L. de Groot, and Hadas Leonov. Calculation of binding free energies. *Methods in molecular biology (Clifton, N.J.)*, 1215:173–209, 2015.

[21] Stefan Boresch and H. Lee Woodcock. Convergence of single-step free energy perturbation. *Molecular Physics*, 115(9-12):1200–1213, 2017.

[22] Charles H. Bennett. Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, (22):245–268, 1976.

[23] Michael R. Shirts, Eric Bair, Giles Hooker, and Vijay S. Pande. Equilibrium free energies from nonequilibrium measurements using maximum-likelihood methods. *Physical review letters*, 91(14):140601, 2003.

[24] Michael R. Shirts and John D. Chodera. Statistically optimal analysis of samples from multiple equilibrium states. *The Journal of Chemical Physics*, 129(12):124105, 2008.

[25] Shankar Kumar, John M. Rosenberg, Djamal Bouzida, Robert H. Swendsen, and Peter A. Kollman. The weighted histogram analysis method for free-energy calculations on biomolecules. i. the method. *Journal of Computational Chemistry*, 13(8):1011–1021, 1992.

[26] Pavel V. Klimovich, Michael R. Shirts, and David L. Mobley. Guidelines for the analysis of free energy calculations. *Journal of computer-aided molecular design*, 29(5):397–411, 2015.

[27] Stefan Boresch and Stefan Bruckner. Avoiding the van der waals endpoint problem using serial atomic insertion. *Journal of computational chemistry*, 32(11):2449–2458, 2011.

[28] Thomas C. Beutler, Alan E. Mark, René C. van Schaik, Paul R. Gerber, and Wilfred F. van Gunsteren. Avoiding singularities and numerical instabilities in free energy calculations based on molecular simulations. *Chemical Physics Letters*, 222(6):529–539, 1994.

[29] M. Zacharias, T. P. Straatsma, and J. A. McCammon. Separation-shifted scaling, a new scaling method for Lennard-Jones interactions in thermodynamic integration. *The Journal of Chemical Physics*, 100(12):9025–9031, 06 1994.

[30] Thomas Steinbrecher, InSuk Joung, and David A. Case. Soft-core potentials in thermodynamic integration: comparing one- and two-step transformations. *Journal of computational chemistry*, 32(15):3253–3263, 2011.

[31] Sunhwan Jo, Taehoon Kim, Vidyashankara G. Iyer, and Wonpil Im. Charmm-gui: a web-based graphical user interface for charmm. *Journal of computational chemistry*, 29(11):1859–1865, 2008.

[32] Benedict Braunsfeld. *Implementation and Testing of CHARMM as Backend to the Free Energy package Transformato*. Master's thesis, University of Vienna, Vienna, 2021.

[33] Shuai Liu, Lingle Wang, and David L. Mobley. Is ring breaking feasible in relative binding free energy calculations? *Journal of chemical information and modeling*, 55(4):727–735, 2015.

[34] Renxiao Wang, Xueliang Fang, Yipin Lu, and Shaomeng Wang. The pdbbind database: collection of binding affinities for protein-ligand complexes with known three-dimensional structures. *Journal of medicinal chemistry*, 47(12):2977–2980, 2004.

[35] Rdkit: Open-source cheminformatics; http://www.rdkit.org.

[36] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA, 2008.

[37] Shimon Even and Guy Even. *Graph algorithms.* Cambridge University Press, Cambridge, second edition edition, 2012.

[38] https://pypi.org/project/py3dmol.

[39] Mark E. J. Newman. *Networks: An introduction.* Oxford University Press, Oxford, 2010.

[40] Hannes H. Loeffler, Stefano Bosisio, Guilherme Duarte Ramos Matos, Donghyuk Suh, Benoit Roux, David L. Mobley, and Julien Michel. Reproducibility of free energy calculations across different molecular simulation software packages. *Journal of chemical theory and computation*, 14(11):5567–5582, 2018.