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 Anna Leopold Hackl, BA BA BSc BSc MA

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

Master of Science (MSc)

Wien, 2022 / Vienna, 2022

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 Bionformatik
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Stefan Boresch
Mitbetreut von / Co-Supervisor:	Dr. Marcus Wieder, MSc MSc

Contents

1 Introduction **1**

2 Free Energy Calculations **3**
2.1 Basics . . . 3
2.2 Methods for evaluating free energy differences . . . 4
2.2.1 Thermodynamic Integration (TI) . . . 5
2.2.2 Free Energy Perturbation / Zwanzig Relation . . . 5
2.2.3 Bennett Acceptance Ratio (BAR) . . . 6
2.3 Soft-core potentials . . . 7
2.4 Dummy atoms, Single/Dual topology . . . 8
2.5 Serial atom insertion . . . 8

3 Transformato **11**
3.1 Common core approach . . . 11

4 Problem description **13**
4.0.1 Used algorithms and software packages . . . 13
4.1 Assessment of common core settings . . . 14
4.2 Graph algorithms . . . 17
4.3 Added functionality . . . 18
4.3.1 Functions . . . 18
4.3.2 Processing of hydrogen atoms . . . 20
4.3.3 Examples for processing molecules . . . 22

5 Results **25**
5.1 Visualizations . . . 25
5.2 Scoring schemes . . . 26
5.3 Results for selected molecule pairs . . . 31

6 Conclusion **37**

7 Appendix **39**
7.1 First set of molecules (set-up failed) . . . 39
7.2 Second set of molecules . . . 40
7.3 Routes for molecules from Transformato paper . . . 44
7.4 Results for individual runs . . . 47

List of Figures **61**

Bibliography **65**

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 unphysical 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].

This Master Thesis is concerned with specific methods of handling these unphysical 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 approach[2][3]. 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 of the algorithms of the software package is described in more detail. Examples for alchemical mutations proposed by the new algorithms and corresponding common core constructions for Transformato are given. Finally, the effect of different mutation algorithms on the results of free energy calculations is discussed.

2 Free Energy Calculations

2.1 Basics

In the last years, the accuracy and feasibility of free energy calculations improved significantly [4]. The main reasons are developments in the accuracy of force fields[5], 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[6], CHARMM[7] or GROMACS[8], 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.[5]As free energy provides information about the thermodynamic favorability of a specific process, it can help to find ligands that bind 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)[9][10], 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 bind 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 [4].

Relative free energy calculations harness the concept of a thermodynamic cycle [11](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, both paths in the figure from the unbound state of ligand A to the bound state of ligand B must exhibit the same free energy difference. These paths are closed and so the energy difference has to be zero, i.e.,

$$\Delta G_A + \Delta G_2 = \Delta G_1 + \Delta G_B.$$

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

$$\Delta\Delta G = \Delta G_2 - \Delta G_1 = \Delta G_B - \Delta G_A$$

[5]. To obtain knowledge about $\Delta\Delta G = \Delta G_B - \Delta G_A$, the evaluation of the alchemical transformations $\Delta G_1$and ΔG_2 suffices. (In practice, the determination of ΔG_A or ΔG_B

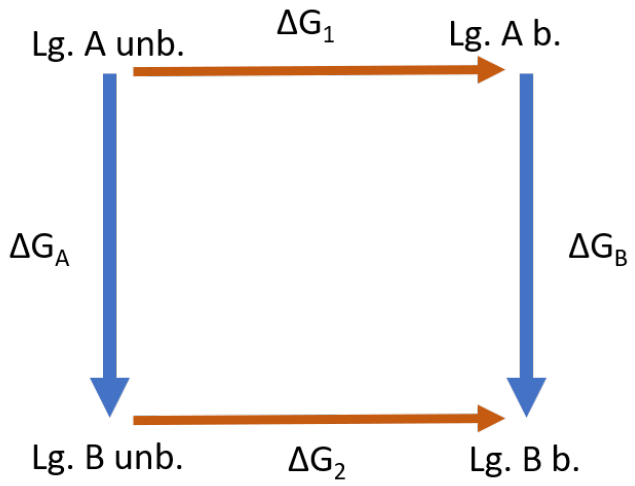


Figure 2.1: Thermodynamic cycle; red arrows indicate transitions between two states (unbound - bound) of each ligand, blue arrows indicate 'alchemical' transformations (Ligand A - Ligand B)

usually 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 less annihilation or creation steps are) and thus, in general, fewer intermediate steps are necessary; however it involves 'alchemical' transformations, i.e., non-physical 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:

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

[12].

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 endstates; 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: $\Delta F = F_1 - F_0 = \sum_n \Delta F_n$ [13].

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 [14] and [15]).

2.2.1 Thermodynamic Integration (TI)

In Thermodynamic Integration, the free energy difference between two states is computed by evaluating the integral over the free energy between the final states. 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 $0 > \lambda < 1$, gives rise to the unphysical, 'alchemical' intermediate states.

Taking the derivative of the free energy, one gets $\frac{dF}{d\lambda} = \frac{d}{d\lambda} \int e^{-\beta U dr} = \left\langle \frac{dU(r,\lambda)}{d\lambda} \right\rangle_\lambda$[12], , where the angular brackets indicate the ensemble average . The free energy difference is then given as the integral from $\lambda = 0$ to $\lambda = 1$:

$$\Delta F = \int_{\lambda=0}^{\lambda=1} \frac{dF(\lambda)}{d\lambda} d\lambda.$$

This integral can be approximated using numerical integration, $\Delta F = \sum_i w_i \left\langle \frac{dU(r,\lambda)}{d\lambda} \right\rangle_{\lambda_i}$. The weights w_i depend on the numerical scheme chosen for approximation. A popular and simple choice is the trapezoidal rule which uses equal spacings between states and approximates the integral by $\int_{\lambda=j}^{\lambda=i} \frac{dF(\lambda)}{d\lambda} d\lambda \approx \frac{i-j}{2} (f_i + f_j)$. 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[12]. (For a comparison of various numerical quadrature schemes, e.g. trapezoidal rule and Simpson's rule, see [16]).

2.2.2 Free Energy Perturbation / Zwanzig Relation

Free energy perturbation relies on the Zwanzig relation (or exponential formula). For each configuration of state A the energy difference between this state and the corresponding state B is calculated. So the free energy difference is given by

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

[17].

Several variants of the formula exist: For instance, one can either calculate forward or backward perturbations (either $\Delta F(A \to B)$or $-\Delta F(B \to A)$ is computed), or, as it is usually the case, use double-wide sampling where energy differences from both directions are processed [14].

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 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 - i.e. overlap must be quite big -, otherwise convergence is extremely bad. However, there are still use cases when other methods are not feasible[18] and there exists an extension to non-equilibrium work (Jarzynski's equation)[18], but usually it should be only used if one knows that the difference between the samples is very small[12].

2.2.3 Bennett Acceptance Ratio (BAR)

An extension of perturbation for computing the free energy difference between two states is the usage of Bennett's Acceptance Ratio[19].

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

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

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 $W\left(q_1 ... q_n\right) = c\left(\frac{Q_0}{n_0}\exp\left(-U_1\right) + \frac{Q_1}{n_1}\exp\left(-U_0\right)\right)^{-1}$[19].

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 simulation are processed simultaneously.

For two λ-states, the free energy difference can be expressed

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

with the Fermi function $f\left(x\right) = \frac{1}{1+\exp(\beta x)}$ [14][17].

To obtain the value of the optimum shift constant

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

, a self-consistency problem has to be solved iteratively[17].

The free energy between two λ-states is then given by $\triangle F\left(\lambda_i \to \lambda_j\right) = \beta^{-1} \ln \frac{N_j}{N_i} + C$[14]. 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 (i.e. for an infinite number of measurements it yields an unbiased estimation) estimation with lowest variance[20].

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 $M = \beta^{-1} \ln \frac{N_j}{N_i}$ [20].

Also BAR depends on the overlap between the states, however it is more robust and reliable in cases of rather poor overlap [15] (especially when compared to FEP). (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[12].)

2.3 Soft-core potentials

Usually, alchemical transformations rely on a coupling parameter which is used to gradually modify interactions. For example, the scaling of the coupling parameter can lead to a weakening of the interaction and complete removal for 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 greater differences, for each atom which has to be transformed into a dummy atom this annihilation process has to be carried out.) 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 the intermolecular interactions). [21]

The occurrence of this problem can be easily illustrated for the case of a linear dependence of the coupling parameter, i.e.

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

.

(Of course, the equal spacing of the coupling parameter does not imply equal phase-space overlap between the states [12]; thus for many simulations this simple set-up is certainly not the optimal solution.) 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 pertaining division 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 do not carry out the division by zero. In practice, it seems that all common MD simulations packages manage this case automatically[21]).

For $\lambda \to 0$, numerical instabilities can occur because even within a small time-step the interactions 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 exactly the same place in the simulation box as the dummy atom causes this quantity to change in an unpredictable way and, in the worst case, become singular [21].

The common way to avoid such problems is the usage of soft-core potentials [22]. 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. The usual Lennard-Jones potential is replaced by a slightly modified potential which ensures that at $r = 0, \lambda > 0$ the divisor is greater than 0 and hence the infinity is smoothed out:

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

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. 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 are transformed into dummy atoms which belong only to the start state. As the alchemical transformation proceeds, dummy atoms are re-transformed in atoms of the second molecule.

In dual topology, no direct mutation of real atoms into dummy atoms occur. However, both physical molecules are extended with all dummy atoms of the other one (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 single topology. [1]

The implementation of such dummy atoms, however, is not without pitfalls. Using 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 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. In [21], this approach is called Serial atom insertion. 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 overlap of neighboring states in phase space is crucial for any alchemical free energy calculation. So the question arises if turning off of 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 [21].

In particular, Serial atom insertion has the advantage that it works even for MD algorithms 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 endstate 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, no intermediate steps are necessary and atoms can be turned off one by one. Contrariwise, thermodynamic integration cannot be used because it is exactly 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 of the potential problems associated with the single and double topology approaches for dummy atoms. The physical endstates 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 molecules 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 according to the serial atom insertion described above; hence, Transformato does not rely on the use of softcore potentials. The computation of the resulting energy differences is based on Bennett's acceptance ratio described above.

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[23][24].

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 common core and dummy region has to be unique; only one dummy region is only 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 [25].)

During the mutation process, the atoms are turned off gradually: In a first step, all hydrogen atoms outside the common core are excluded. The electrostatic interactions of dummy atoms are turned off. In a next step, the van-der-Waals-interactions are processed (in one step per atom in line with serial atomic insertion).

Before the common core construction is carried out, hydrogens are 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 flawed mutation route. For the workflow of Transformato, inclusion of hydrogens in the mutation algorithms would be detrimental because these atoms are already turned off beforehand.

4 Problem description

The main objective was the assessment and improvement of some routines used in the free energy package Transformato. A further goal was to minimize the necessity of additional adjustments by the user, i.e. reasonable mutation routes have to 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 possible orders exist. (Because of this fact, the evaluation of the mutation algorithms in the following sections will focus on such transformations.) Neither should the mutation of atoms generate vacancies in the inner part of the molecule nor should opened rings remain 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 common core, i.e. no disconnected components 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 have been 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 by means of molecular dynamics simulations.

To obtain a reliable test set of molecules, sdf-files of ligands were downloaded from the PDBbind-CN database. These test molecules should cover a broad range in size, complexity and potentially intricate compounds, like polycyclic structures and highly branched chains.

4.0.1 Used algorithms and software packages

The Transformato package is written in Python. Therefore, Python packages also were used for molecule processing and graph representations. The creation of molecule objects and the determination of the maximum common substructure is done via Rdkit[26]. NetworkX[27] 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 common cores constructed. Particularly, graph traversal algorithms like breadth-first and depth-first search can be easily implemented (see below).

4.1 Assessment of common core settings

Rdkit allows the search for a maximal common substructure (which can serve as common core for Transformato) via the `rdFMCS.FindMCS`-function. Per default, the objective is to maximize the number of atoms, albeit different settings, like maximize the number of bonds or ignoring or equalizing specific atom types are available. At the moment, for Transformato maximization of atoms and atom identity is used. If the data comprises hydrogens, these are excluded before the common core is calculated. As stated above, the presence of hydrogens can influence the maximum common substructure heavily (fig. 4.1) 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 common core are of crucial importance. Firstly, these parameters can influence the common core construction drastically and, secondly, they can even be decisive whether the generated common core is valid for the Transformato workflow. Especially for the generation of common cores 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 common cores 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 common core. Especially the latter constraint is a necessary condition for a valid common core. (Otherwise, if partial rings take part of the common core, dummy regions will be inevitably connected to the common core via multiple bonds.)

The `ringCompare`-parameter parameter accepts the `StrictRingFusion`-argument. It imposes that in case of multiple rings aromaticity is properly taken into account. Fig. 4.2 illustrates the effect of the parameters on the common core of two cyclic example molecules. However, enforcing of `StrictRingFusion` can still lead to maximum common substructures that are not valid Transformato common cores in case of a dummy region which is connected via multiple ring atoms of the same ring with the common core. When an invalid common core is generated, it seems to be advisable to warn the user in this case that the common core does not fit the expectations of the Transformato workflow. The mutation algorithms presented below can deal with such invalid common cores. A helper function arbitrarily chooses one of the connections between common core and 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 common cores in a proper way. Therefore it makes only sense for test purposes and is not a reasonable input for the regular workflow. Alternatively, one could

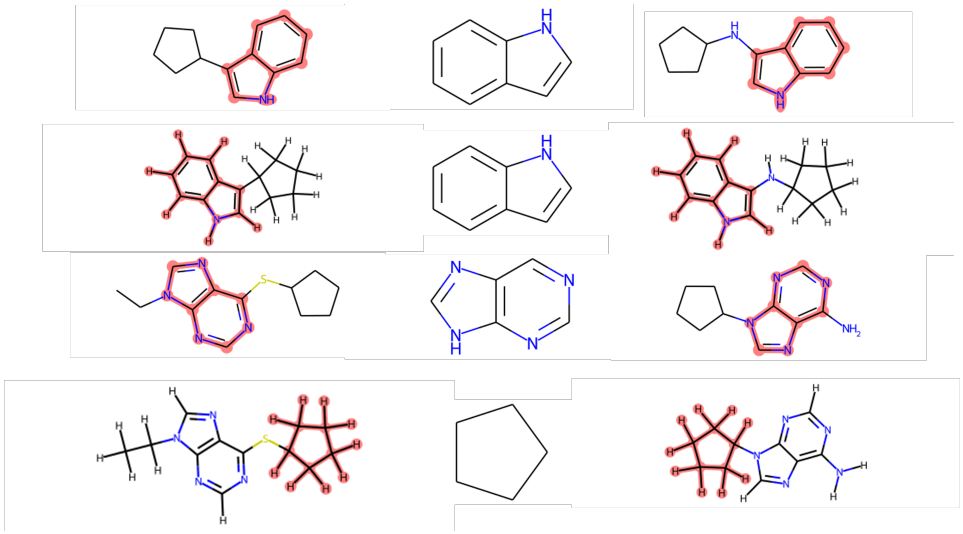


Figure 4.1: These examples demonstrate that the addition of hydrogens can influence the construction of the common core; left: molecule 1; middle: common core; right: molecule 2; in the representations of the full molecule, the common cores are marked in red; the first and the last two rows show the same molecules, in the upper row without, in the lower with hydrogens, in case of the lower molecule combination the common core changes when hydrogens are added to the Rdkit-molecule representation

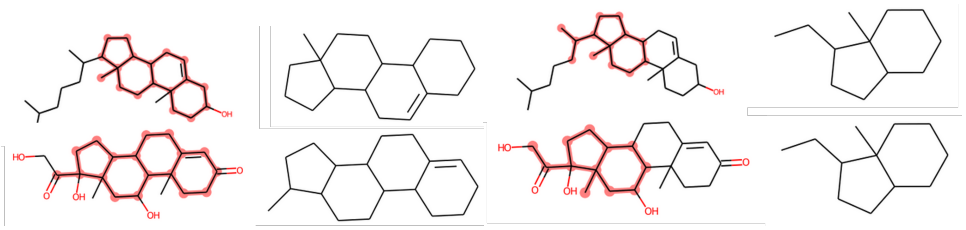


Figure 4.2: 'Strict fusion' can affect the construction of the common core drastically; first and second column: Common core of two molecules (cholesterol and cortisol) without strict fusion; third and fourth column: Common core of two molecules with strict fusion; in the images of the full molecule graph common cores are marked in red. In this case, none of the settings yield a valid common core, the iterative approach would be necessary to obtain one

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

These problems occur because the requirements for a valid Transformato common core are more strict than the constraints imposed by rdkit (even when all ring-related parameters are applied, i.e. `ringMatchesRingOnly`, `completeringsonly` and `stricringfusion`): For Transformato, is not only indispensable that the common core solely comprises complete rings, but also the molecule fragment consisting of the non-common core atoms (which implies that there is a unique connection between dummy region and common core). An efficient and straightforward solution for this problem, which always yields the best - i.e. largest - valid common core if one exists, has been implemented: The basic observation is that if common core atoms within a cyclic region give rise to a invalid common core, 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 common core). After generating a common core with standard parameters, it is checked if the additional requirements concerning the ring structure hold, i.e. if the non-common core atoms (as well as the common core atoms) of both molecules do not participate in a partial cycle. (Alternatively, one could check if a path between the non-common core atoms adjacent to the common core, i.e. the X-atoms, exists. If this is the case, the common core obviously is not valid, since no unique atom which connects common core 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 common core are removed from the representation used for creating the maximum common substructure and afterwards a new search for a valid common core is started. This procedure is repeated iteratively until a valid common core is found. Of course, in the worst scenario, i.e. if both (non-identical) molecules only consist of ring-participating atoms, no valid common core is conceivable (and, given the total size of the molecules, the valid size can be very small). In general, if at least one of the molecules

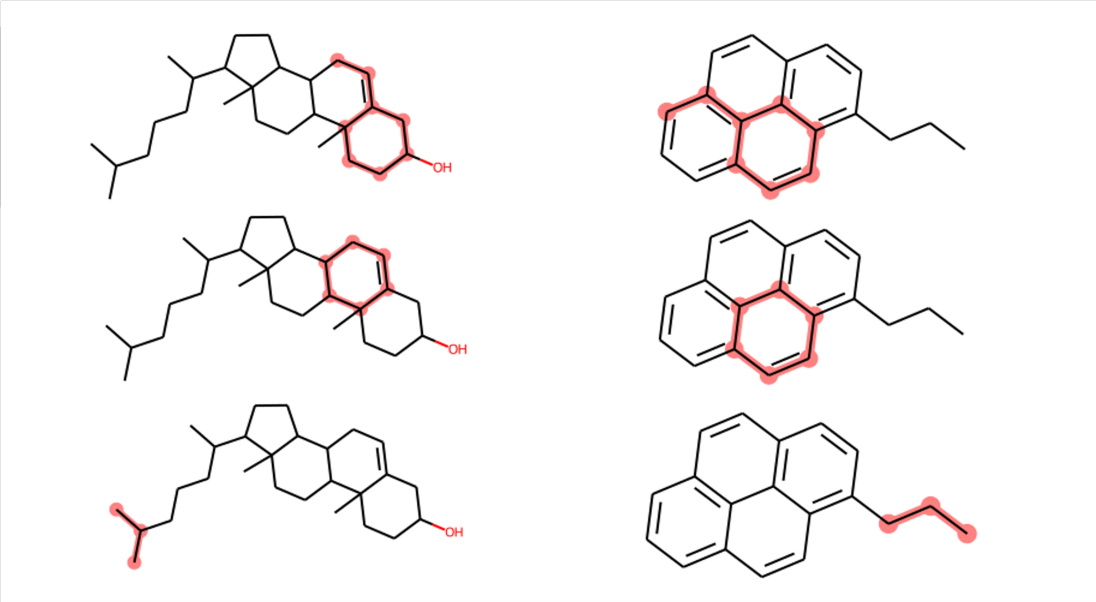


Figure 4.3: common cores of cholesterol (left) and 1-pyrenepropanoic acid (right); upper row: CompleteRingsOnly = False; middle row: CompleteRingsOnly = True; lower row: iterative approach to obtain a valid Transformato common core

solely consists of a polycyclic compound without any functional groups etc., and the other one does not have exactly the same compound, no valid common core can exist. An – maybe rather contrived – example for such a pair of molecules and the construction of valid common core via the iterative approach is given in Fig. 4.3. One sees that even the usage of the ring-related parameters of Rdkit doesn't yield a valid common core, whereas the iterative approach gives the desired result.

4.2 Graph algorithms

Using NetworkX and Rdkit, the molecules and their common core 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 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.[28] (Problems and differences of both algorithms are illustrated using several examples below.)

In each of the algorithms implemented, the graph traversal starts with the node which connects dummy region and common core as the root. Shortest paths to all nodes of the

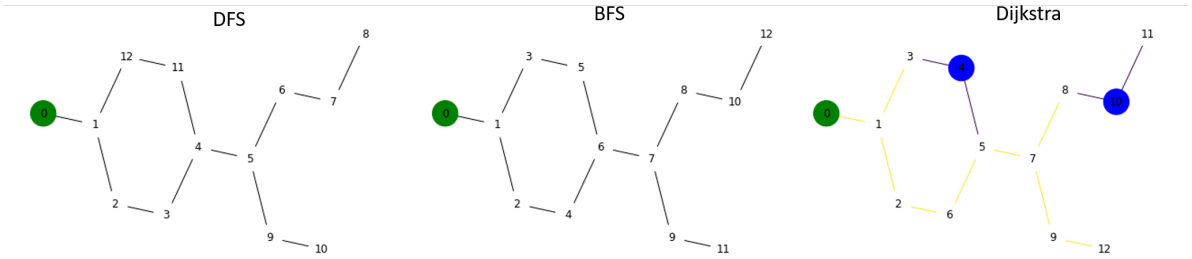


Figure 4.4: Comparison of different graph traversal algorithms; green nodes represent the root atom, blue atoms (in the figure shown right) have increased weights; left: depth first search; middle: breadth first search; right: Dijkstra algorithm, the edges connecting blue colored nodes have increased weights leading to a mutation route differing from BFS; the final processing of the nodes happens in reversed order

dummy region are determined.

As the longest shortest path pertains to the node which has the greatest distance from the root (i.e. the atom with greatest distance from the common core), the last element of the list returned by the search algorithm indicates the atoam which has to be removed first; hence, the list of mutations orders finally 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 uniformly weights.)

4.3 Added functionality

4.3.1 Functions

To use the newly implemented mutation algorithms, initially the graph is given with weights stored in a dictionary. The simulations shown below use uniform weights, however it is possible to modify these to enforce a specific mutation route (e.g., accelerating or postponing the exclusion of heteroatoms).

The Dijkstra algorithm is implemented via the `single_source_dijkstra`-function of NetworkX.

The core functionality is given by the mutation processing functions. At the moment, three such functions are implemented, in addition to the simple DFS-approach, which leads to undesired outcomes, in particular because often heavy atoms in neighborhood to the common core atoms are processed at the first steps of the graph algorithm as well as rings are opened early, but finally processed at a later stage (in a rather 'asymmetric'

manner). These functions can be further modified by passing arguments which activate some helper functions (see below).

The four main functions are:

`_calculate_order_of_LJ_mutations`: naive DFS

`_calculate_order_of_LJ_mutations_new`: BFS/djikstra-algorithm applied once for route

`_calculate_order_of_LJ_mutations_new_iter`: BFS/djikstra-algorithm applied iteratively, i.e. after each removal of an atom

`_calculate_order_of_LJ_mutations_new_iter_change`: works iteratively, i.e. after each removal of an atom, algorithm is chosen depending on current state

`_calculate_order_of_LJ_mutations` was already present in Transformato, but computes defective mutation routes and hence should only be used for test purposes. The other three algorithms are new and, in any case, resolve most of the problems of the earlier algorithm (especially isolated removal of ring atoms).

Helper functions like cycle_checks carry out tasks to ensure the desired mutation route, e.g. count the number of cycles an atom participates in. Further features of all algorithms are 'preferential removal', i.e. if two atoms would 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 excluded next.

`cycle_checks(G)`: this function checks which atoms participate in how many cycles/rings and 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 cycle and degree dictionary. If nodes have same weight (i.e. distance from root), the node participating in more cycles is removed later if nodes have same the weight (i.e. distance from root) and same cycle participation number, the node which has more neighbours already removed 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 last step, its weight get 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 True, updates are updated according to cycle participation (as the systematic processing of ring structures

is one of the central goals, this argument should always be set to True, 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` also ordercycles or ordercheck, respectively, is set to True, weight updating according to preferential removal decides that the node in which neighbourhood nodes already have been turned off is removed next if there is no possibility to decide between two nodes - i.e. the weight of both would be exactly 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.

The BFS- / Dijkstra-algorithm starting from the node connecting common core and dummy region is applied 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 large molecules the additional computational cost is negligible, in particular in comparison to the computational time needed for the search of the common core. 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`. In contrast to the other algorithms, the algorithm processes information if 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. 4.5 demonstrates the difference between the two versions of the iterated algorithm. Whereas in the non-iterated algorithm the found 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 computed mutation routes can be visualized directly via Rdkit. The route is represented by a colour 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 green.

Furthermore, an animated 3D-visualization of the mutation process is implemented using py3Dmol (`animated_visualization_3d_v2`)[29].

4.3.2 Processing of hydrogen atoms

As fig. 4.1 indicates, the presence of hydrogens can influence the common core generation dramatically. In particular, the maximum common substructure for a molecule representation with hydrogens can encompass less 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). Hence it is crucial

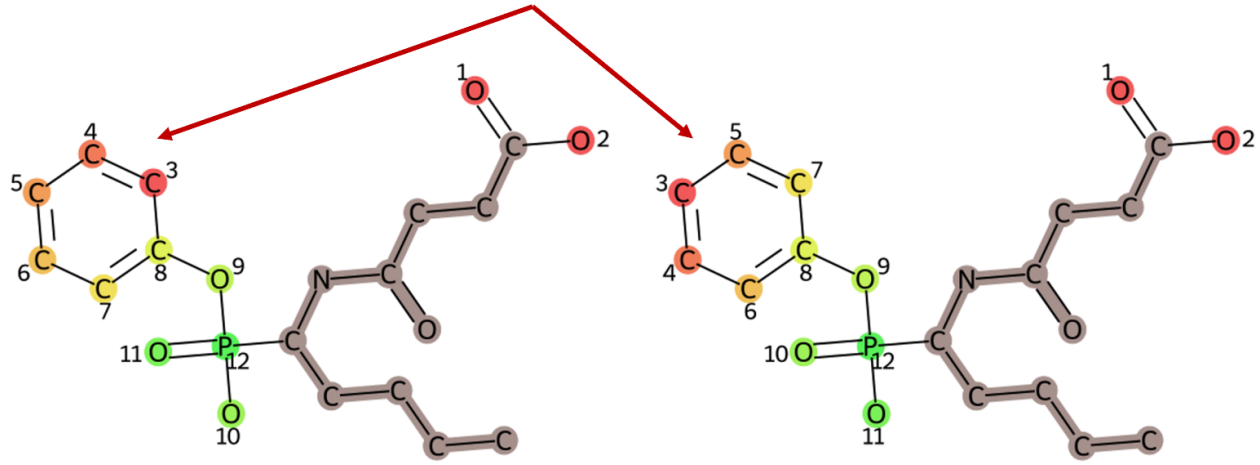


Figure 4.5: comparison of typical DFS- and BFS-mutation route for a single ring structure; left: DFS-algorithm; right: BFS-algorithm; in all depictions of mutation routes the common core is shown in dark, the color gradient starts at red indicating the atom first to be removed and ends at green; DFS starts at carbon 22 and thus ring breakage gives rise to one long chain which is processed subsequently, whereas BFS starts at carbon 22 and the two emerging chains are processed in a symmetric fashion

to determine the common core for molecule representations without hydrogens. Since hydrogen atoms are turned off in an extra step en bloc during the Transformato workflow, the mutation route algorithm should not take hydrogens into account nor must the common core be generated for a molecule representation containing hydrogens. Nonetheless, it is necessary that the molecule representations processed in Tranformato 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 common core (`_find_mcs` in mutate.py) had to be modified. To obtain the desired common core 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 and afterwards their common core is computed. For both molecules, the indices of the atoms corresponding to the common core are determined. Finally, it is checked for each common core atom of both molecules whether hydrogen atoms are in its neighbourhood. If such hydrogens are found, their indices are added to the lists of common core 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 common core atoms for both of the molecules include

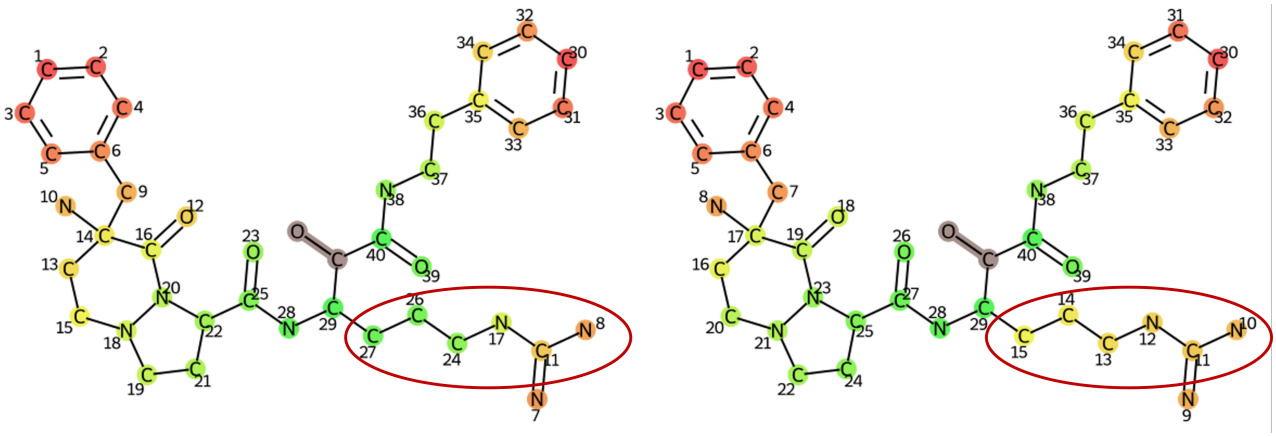


Figure 4.6: Example for the differences between iter- and iter_change-algorithm; left: iter; right: iter-change; iter-change processes all atoms within a chain or a ring at once (if possible) before switching to other parts of the molecule

hydrogens, but the maximum common substructure giving rise to the common core 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.

4.3.3 Examples for processing molecules

In the current version of Transformato, one suboptimal route finding algorithm using DFS and several versions based on BFS are implemented. For single rings, BFS (Dijkstra) automatically processes the atoms in the most symmetric way starting at the atom with the highest distance from the root (fig 4.4). 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. 4.6 shows this problem for a benzol ring which is directly attached to the common core. As DFS goes along one path until the end, i.e. a leaf node is reached, only four atoms of the ring are visited at the beginning, 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. In this case, BFS automatically produces the desired result.

Similarly, in fig. 4.7 one atom of the ring (marked by the red circle) is omitted in the first exploration using DFS.

More complex ring structures exacerbate the problems. Fig. 4.8 shows that also the processing of substituents is affected.

Multiple rings pose special difficulties for the mutation algorithms because the processing

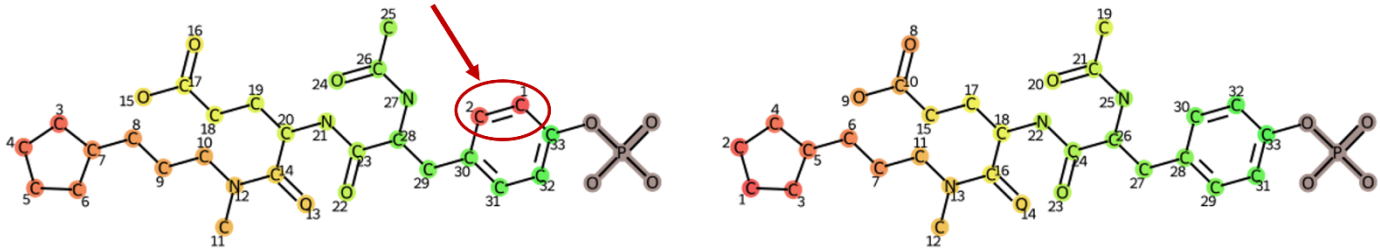


Figure 4.7: left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms

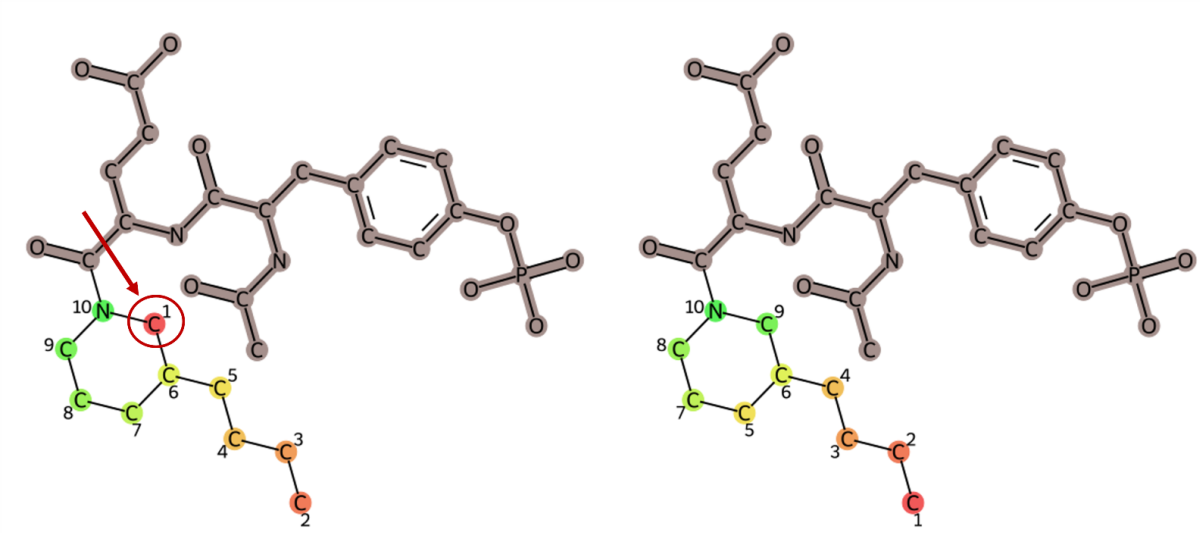


Figure 4.8: left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow and circle indicates the undesired processing of the ring atoms

of one of the rings can easily led to gaps in the adjacent ones. Fig. 4.9 and 4.10 show that using DFS aggravated problems occur. As in the case of one ring, the exploration route implies that one of the rings is opened in a way that it gives rise to a lengthy chain. However, it can even happen that the atom explored atom participates even in two or multiple rings, so that both ring structures are opened and teared apart (fig. 4.9). Alternatively, one half of each of the rings is turned off first (fig. 4.10).

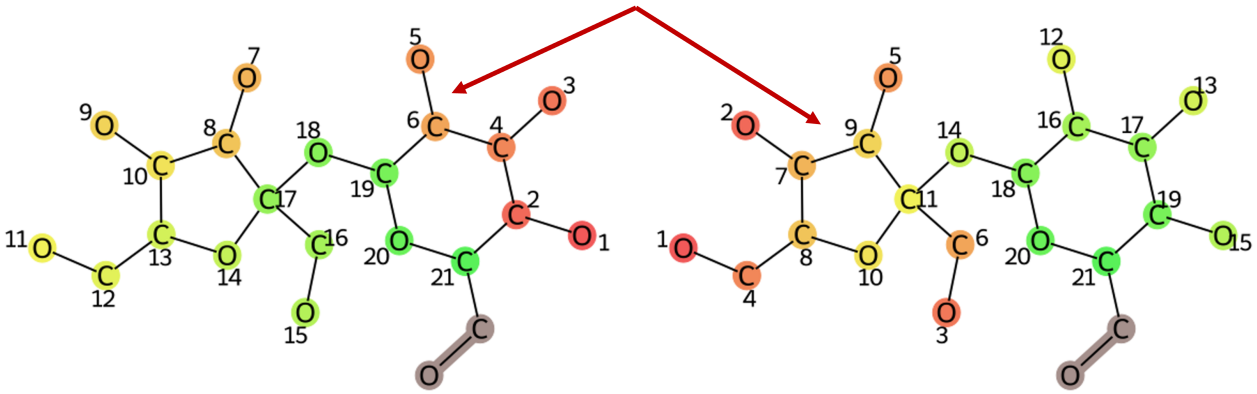


Figure 4.9: left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms

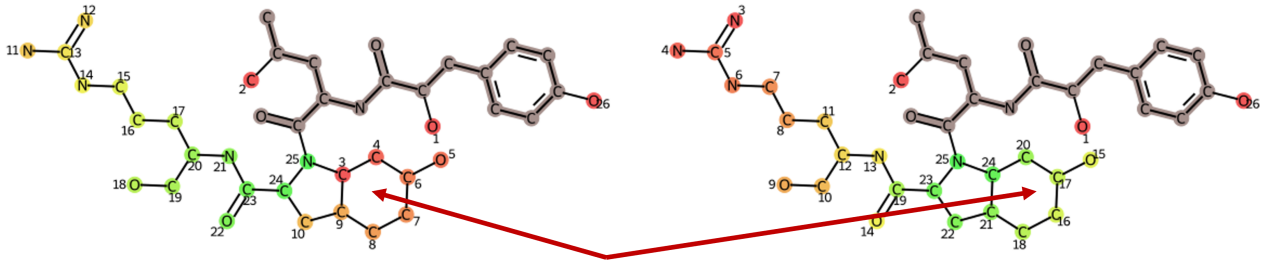


Figure 4.10: left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms

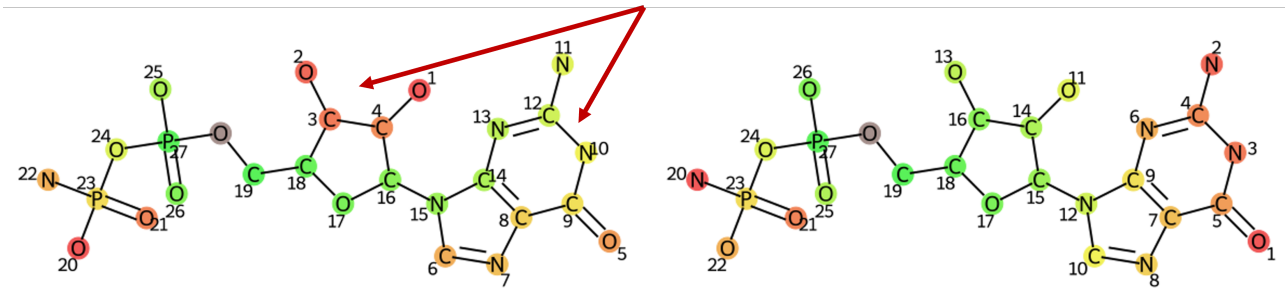


Figure 4.11: left: DFS-algorithm; right: BFS-algorithm; common core in dark

5 Results

A set of twenty ligands from the PDBbind-CN database has been downloaded and used for testing purposes (common core construction as well as mutation routes). It should be noted, however, that the common core of pairs of these ligands in some cases violate the rules of Transformato concerning a valid maximum common substructure, i.e. dummy regions are connected by more than one atom to the common core (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 common core. For the following processing of the mutation algorithm, this connection is arbitrarily distinguished and the other ones removed.

5.1 Visualizations

The mutation route can be visualized using a color gradient (additionally to numbering, see figures above).

Py3dMol is used for a 3D-animation of the mutation process. Fig. 5.1 shows two molecules and their shared common core.

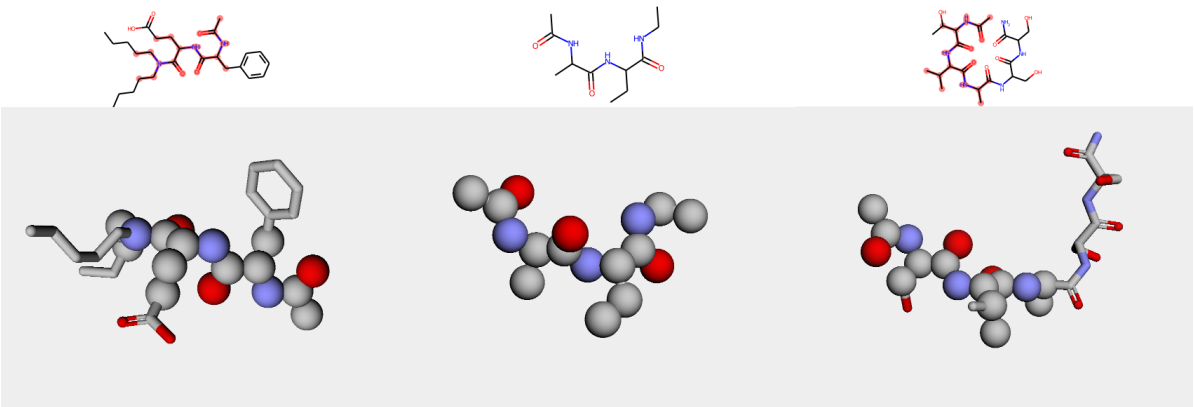


Figure 5.1: Visualization of the mutation route using Py3dmol; upper row: Rdkit-representations of both molecules and the common core; lower row: Py3Dmol visualizations; left and right: molecules; middle: common core of both molecules

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[30]. More central atoms will have a higher centrality coefficient, whereas atoms remote from the common core will have a lower (e.g., 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.

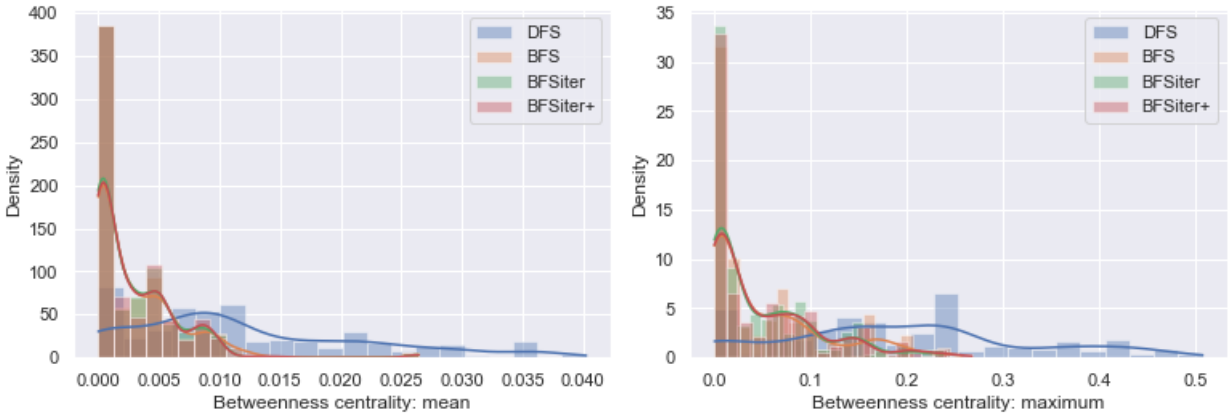


Figure 5.2: Betweenness centrality

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[30]. Atoms more distant from the common core have a lower closeness centrality. For using this centrality measure as scoring function for the mutation algorithms, the dummy region with the greatest 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 removed node for the full graph (i.e. all atoms, including already removed ones) is computed. 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 common core atoms) are removed. To evaluate this relation, linear regression of the computed centrality values is performed. A higher slope indicates a 'better' mutation route. Furthermore, correlation coefficients (Spearman, Kendall's Tau) have been computed. Again, a higher correlation shows that the closeness centrality of the atoms removed is increasing. Correlation coefficients as well as regression coefficients are reported in the tables below.

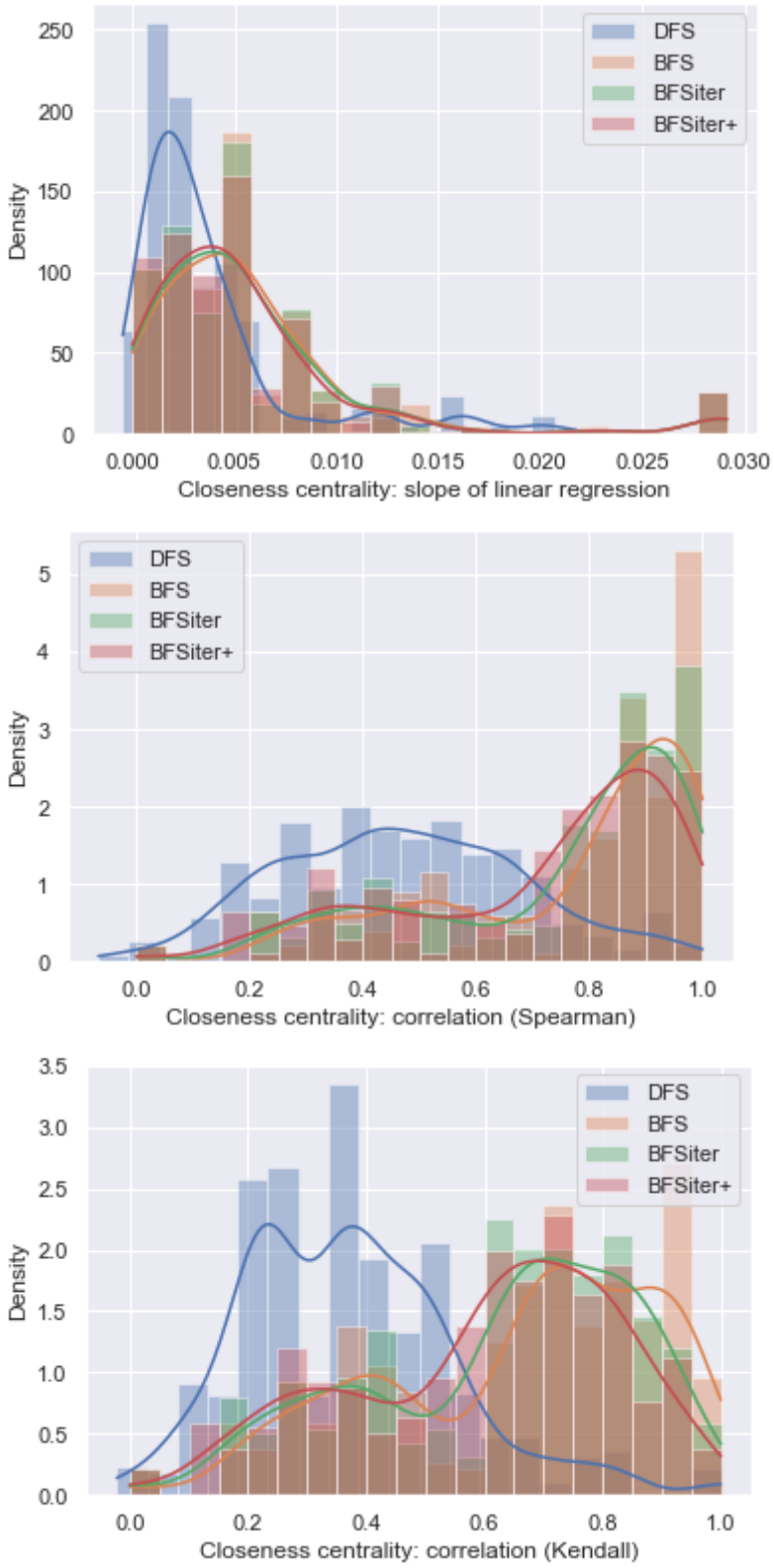


Figure 5.3: Closeness centrality

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: statistics of PDBbind selection; 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 in 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 cycles in all mutation routes; polycyclic: percentage of mutation routes which involve polycyclic structures, i.e. there are atoms present that participate in multiple cycles

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 disassemby'-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 completely process a ring (until only one atom of the former ring structure is present).

It should be noted that even using the new algorithms it is possible that a 'broken' ring stays for a while in the system because atoms from other areas of the dummy region (e.g., a longer chain) are processed meanwhile. However, in contrast to DFS, it should not happen that a ring near to the common core is opened initially and hence the mean and maximum time should be significantly shorter.

In general, calculation of the scoring functions for the selection of ligands from the PDBbind data set (statistics are presented in table 1) match the expectations.

In the plots presenting the scoring-functions, all molecule combinations from the PDBbind data set are used. It could be insightful to use only a subset (e.g. only molecules with dummy regions involving multiple ring structures ore a minimum number of dummy atoms) or to try even a larger selection from the PDBbind database.

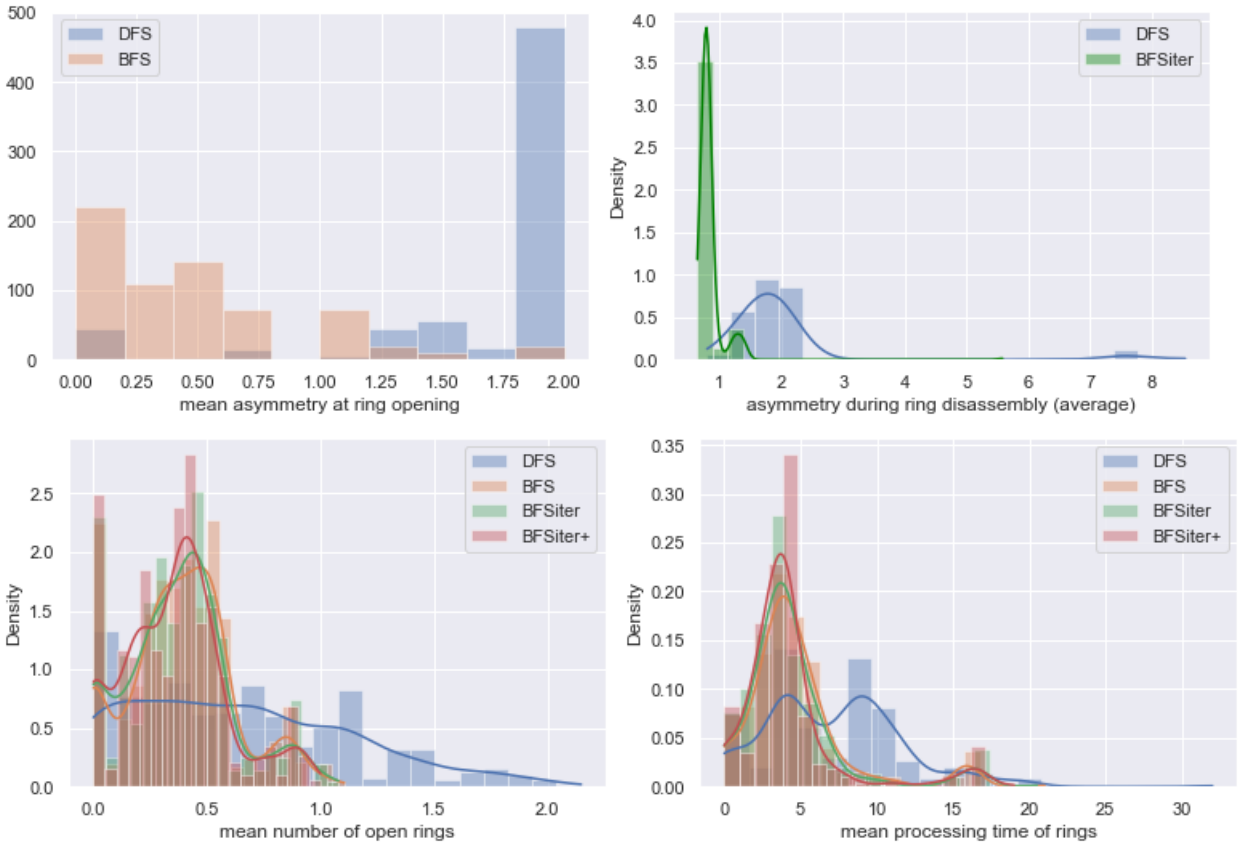


Figure 5.4: Ring-related scores

5.3 Results for selected molecule pairs

For selected pairs of small molecules, the relative solvation free energy differences have been determined using Transformato. An overview of the molecules and the corresponding mutation routes can be found in the appendix.

(Vergleiche conclusion: Bis jetzt gab es 3 Versuche! Der erste mit größeren Molekuelen scheiterte - vielleicht auch, weil hier aufgrund der Wasserstoffproblematik teils falsche common cores berechnet wurden, vielleicht aber auch, weil die Molekuele einfach zu groß waren. Der zweite Versuch wurde fuer die 5 Molekuele in fig 5.5 angestellt, die Resultate sind aber nicht ganz vertrauenswuerdig. 3 Ergebnisse können nicht passen, da ein falscher common core verwundert wurde, weil die Wasserstoffe noch nicht richtig prozessiert wurden, also noch nicht vor common core-Generation und Mutationsfindung entfernt wurden (siehe 'processing of hydrogen atoms')! In einem 3. Versuch wurden zwei der Molekuele, naemlich Toluen und 2-Naphthol nochmals berechet, auch hier ist aber nicht ganz klar, ob alles passt...)

Two such example pairs are toluene→methanol and 2-naphthol→methanol. In both cases, the corresponding common cores solely consist of one heavy atom, carbon or oxygen, respectively.

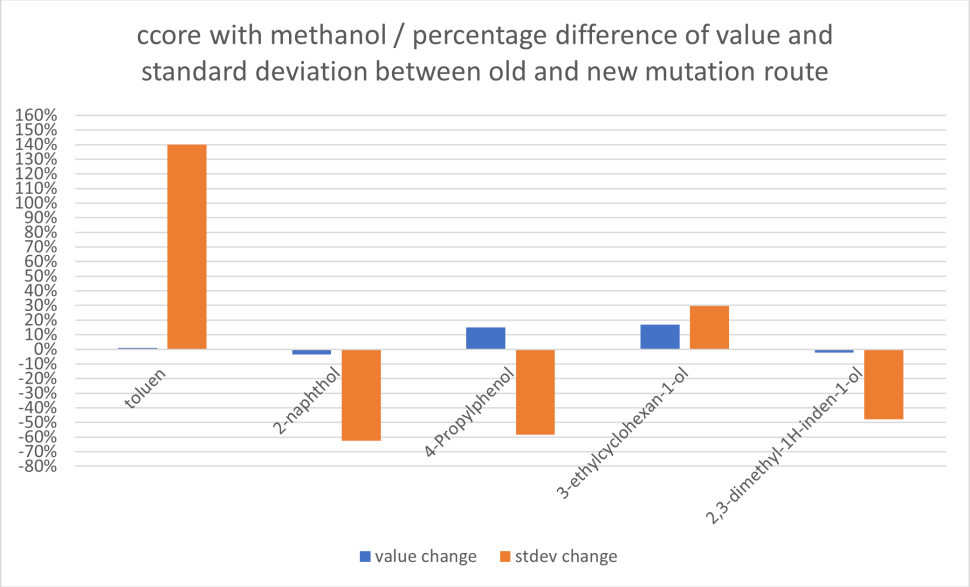
Calculation of the free energy difference indicates that for toluene→methanol the differences between the algorithms are negligible, but for 2-naphthol→methanol they are at least more pronounced (fig. 5.5 and table 5.2). Overlap plots for these mutations can be found in fig. 5.7,8 and 10. showing the differences between old and new mutation algorithms. Fig 5.9 also shows the evolution of the free energy estimate across the lambda states for 2-naphthol (vacuum and waterbox for old and new mutation algorithm).

The overlap plots for Naphthol→Methanol exhibit a different pattern depending on the mutation route, but only for the simulation in vacuum, not for the waterbox, whereas for Toluene-Methanol hardly any difference is visible. (There are two adjacent states in the middle of the heavy atom transformation for naphthol/waterbox with a comparatively high overlap. Correspondingly, the plotting of the lambda states shows to steps in the middle of the simulation with almost no free energy change.)

Since the mutation route is fundamentally different depending on the algorithm (see fig. 7.2; the old algorithm starts next to the atom of the phenyl group that is connected to the methyl substituent - which serves as common core - and processes the rest of the atoms in a chain-like manner, whereas the new one starts at the atom with maximum distance from the substituent and proceeds in a symmetric way until the common core is attained), the result for toluene is highly astonishing. For the double ring structure of 2-naphthol stronger deviations occur; thus it would be insightful to determine which structural properties are of importance.

mutation partners	old algorithm (DFS)	new algorithm (BFS)
toluene/methanol	-4.97 ± 0.05	-4.73 ± 0.06
2-naphthol/methanol	1.04 ± 0.05	0.85 ± 0.22

Table 5.2: results for the mutation pairs methanol and toluene / naphthol



differences between old and new algorithm for selected molecules / 1. VERSUCH, DIESE RESULTATE SIND VERMUTLICH FEHLERHAFT; AUF JEDEN FALL IST BEI PROPYLPHENOL, ETHYLCYCLOHEXAN und DIMETHYLINDENOL DER FALSCHE CCORE VERWENDET WORDEN (aufgrund nicht entfernter Wasserstoffatome!)

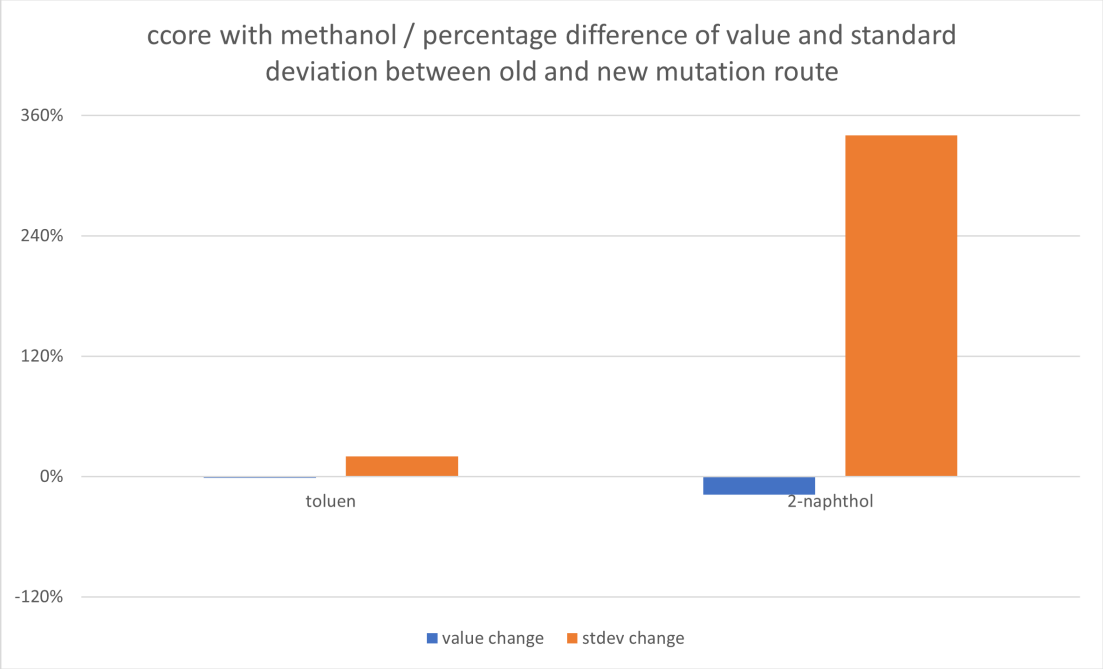
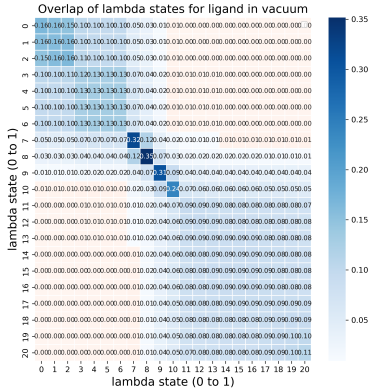
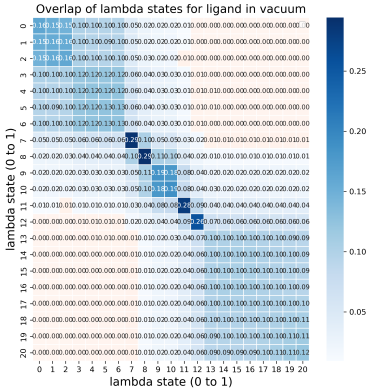


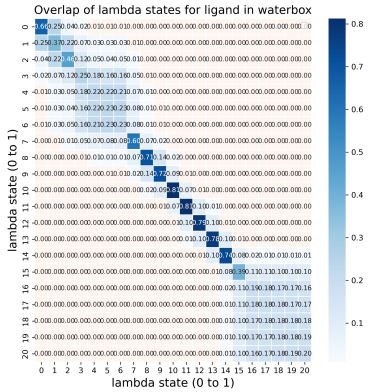
Figure 5.5: differences between old and new algorithm for selected molecules / 2. VERSUCH, muss ebenfalls nochmals ueberprueft werden



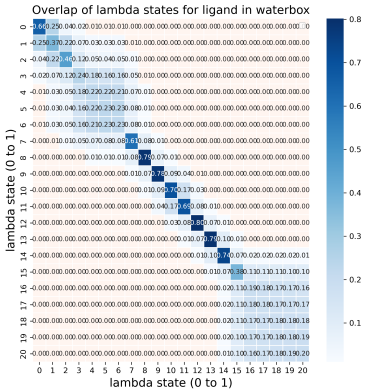
(a) naphthol/vacuum old



(b) naphthol/vacuum new

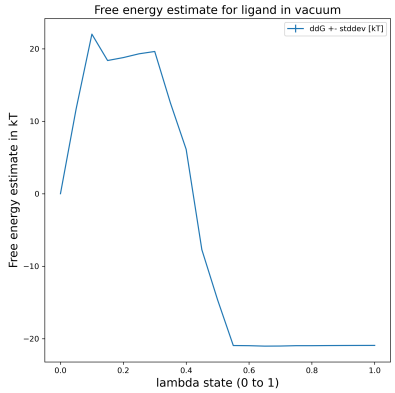


(c) naphthol/waterbox old

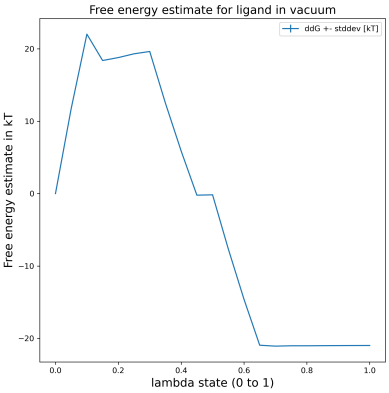


(d) naphthol/waterbox new

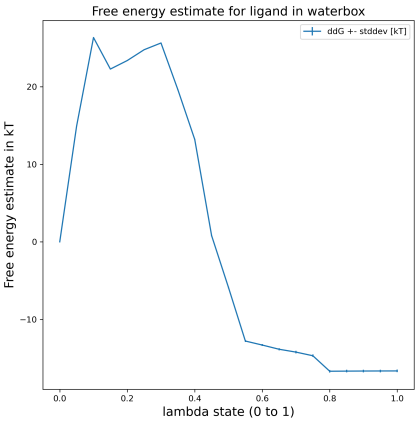
Figure 5.6: Overlap plots for naphthol: upper row: vacuum, lower row: waterbox; left: old mutation algorithm, right: new mutation algorithm



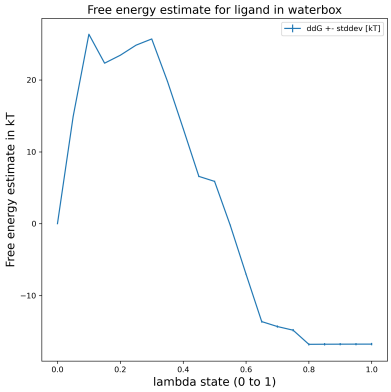
(a) naphthol/vacuum old



(b) naphthol/vacuum new

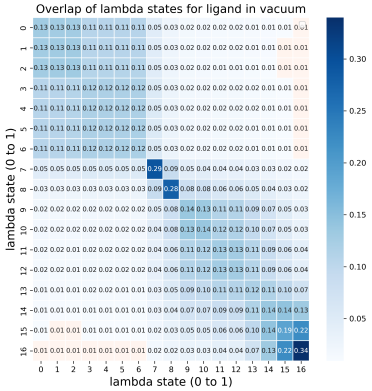


(c) naphthol/waterbox old

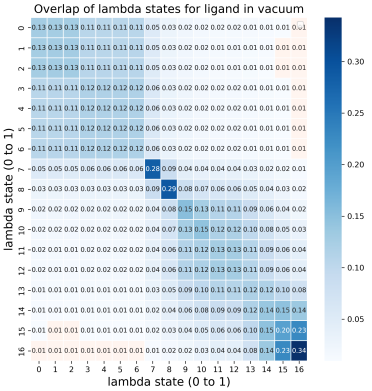


(d) naphthol/waterbox new

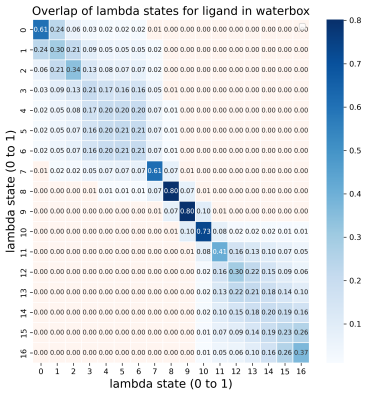
Figure 5.7: Free energy states at different lambda states for naphthol: upper row: vacuum, lower row: waterbox; left: old mutation algorithm, right: new mutation algorithm



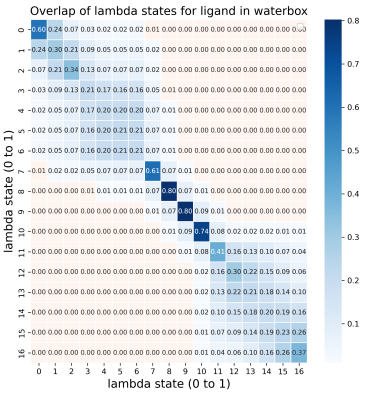
(a) toluene/vacuum old



(b) toluene/vacuum new



(c) toluene/waterbox old



(d) toluene/waterbox new

Figure 5.8: Overlap plots for toluene: upper row: vacuum, lower row: waterbox; left: old mutation algorithm, right: new mutation algorithm

6 Conclusion

The results assessed so far are highly ambiguous. In most cases (NAJA, es sind eigentlich nur 2 cases bis jetzt, wenn man den ersten, wahrscheinlich fehlerhaften run wegzaehlt und davon ausgeht, dass der zweite Versuch funktioniert fuer Toluen und Naphthol funktioniert hat, was jedenfalls noch ueberprueft werden sollte - auch hier gibt es einige Zweifel!!!), no profound differences have been found. In any case, additional comparisons of mutation routes for other molecules and closer examination of the obtained results are necessary.

Perhaps the most promising approach would be a simulation for the molecule pairs used in [31, 3]. (This would also allow a direct comparison with the results stated therein and a validity check of the new results.) Routes for the most interesting (i.e., molecules with non-trivial mutation orders involving rings etc.) molecule pairs (toluene, 2-methylfuran and 2-methylindole to methane and 2-cyclopentylindole to 7-cyclopentylindole) can be found in the appendix 7.3. On the other hand, a more systematic evaluation of the emergence of differences between the mutation algorithms could use molecules with exactly defined, small modifications (for example, one added/removed atom or ring structure) to investigate if the size of the dummy regions or the common core and the combination of some elements, e.g. rings, have an effect.

A further path for detecting differences between the outcome of the mutation algorithms is to compare runs of different sampling length. The simulation length used so far gives good results for any mutation order, but for shorter simulations it could be that differences increase. It could be that the increase in standard deviation which is expected for short sampling lengths is different for different mutation orders. There are several possible outcomes: Either the standard deviation increases slower for one of the algorithms with decreased simulation length, which suggests its superiority, or the resulting free energy differences diverge indicating the importance of the mutation route at least. Finally, it could be that the changes in the free energy results occur at the same simulation length for both algorithms implying that there are no significant differences between the mutation orders. For toluene, simulations with the same set-up, but of shorter length have already been carried out; however, the results do only slighter differ. Basically, the shorter run for toluene→methanol showed identical results (but, as expected, a slightly higher standard deviation) for both algorithms. It would be necessary to decrease the sampling length even more until the results for different mutation orders start do differ significantly.

Especially for more complex transformations an evaluation of the relation between sampling length and standard deviation for different mutation routes could be insightful. (Choosing an algorithm which allows for shorter sampling size by minimizing variance would be advantageous as well.)

Given the results obtained so far, it seems that there is no general superiority of a more 'symmetric', breadth-first-based algorithm. (For toluene and 2-naphthol, the results for

the new algorithms are even worse measured as increase in standard deviation. For both molecules, it is higher for the new algorithm, in the case of 2-naphthol the increase is above 300%.) More data would be required to get more conclusive results (e.g. to check if the 'best' way of processing a ring system depends on its substituents or if the size of the atom is influential etc.). Nonetheless, the results partly indicate that the mutation order could play a more complex role. Although the new algorithm leading to a more 'logical' and systematic processing doesn't give better results, the mutation order seems to influence the free energy estimation, especially the standard deviation for a given simulation length.

The tf-routes package presented in this work also allows to process heavy non-carbon atoms in an individual manner, 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 (the mutation route between methanol and toluene or naphthol) 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).
Perhaps it would be useful to apply different methods for optimizing the mutation route, e.g. learning the route which minimizes the variance per step via machine learning etc. (there are several possibilities, e.g. provide the molecule - as graph representation etc. - and changes in free energy for one atom turned off as training set etc.).

7 Appendix

7.1 First set of molecules (set-up failed)

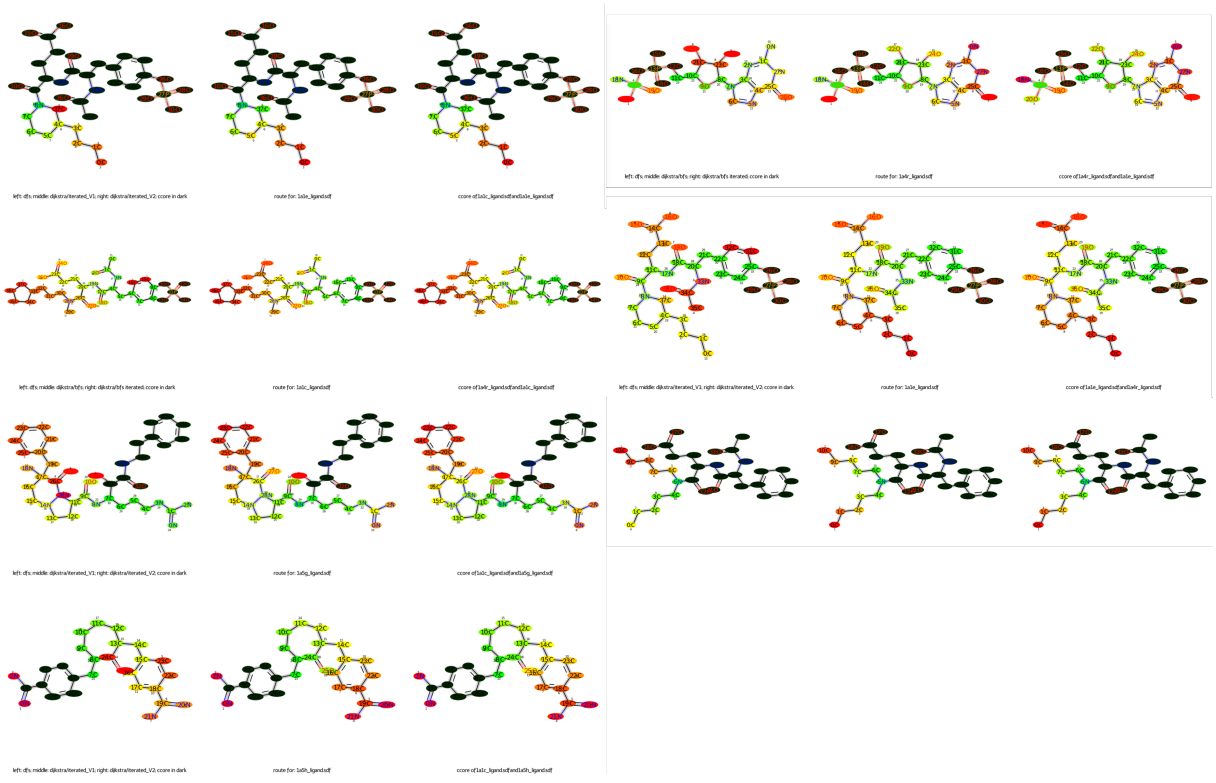


Figure 7.1: left: DFS-algorithm; middle: BFS-iter-algorithm; right: BFS-iter+-algorithm; common core in dark

7.2 Second set of molecules

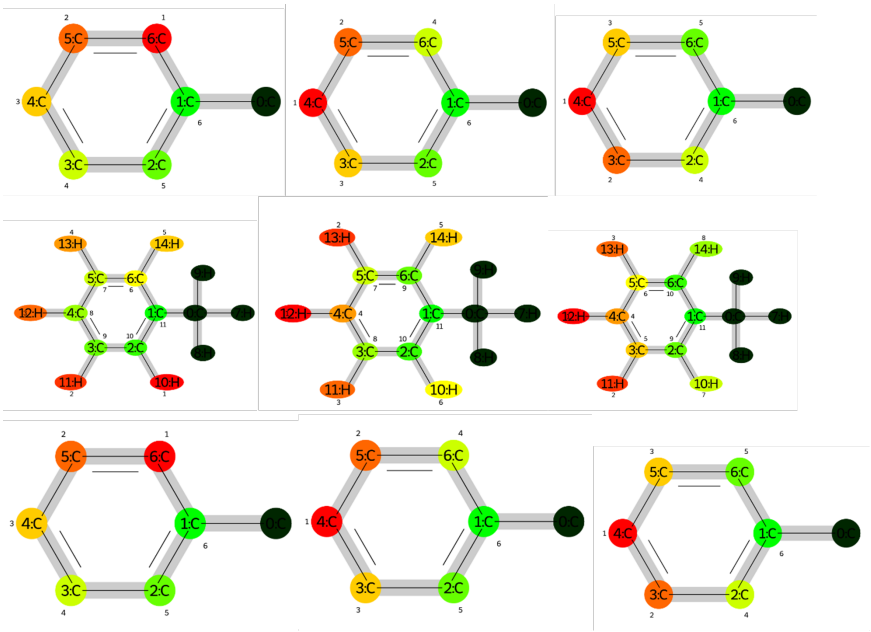


Figure 7.2: Toluene->Methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible

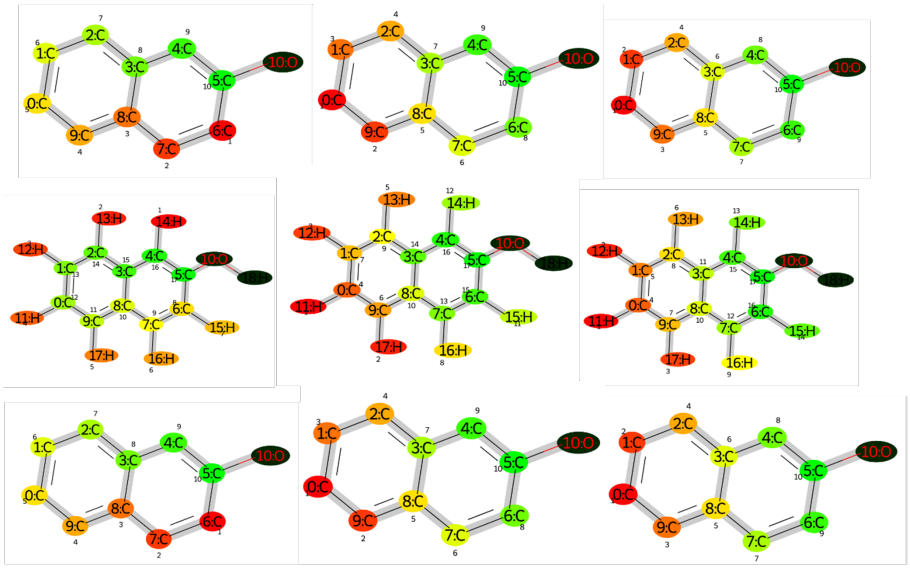


Figure 7.3: mutation route for 2-naphthol for 2-naphthol->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible

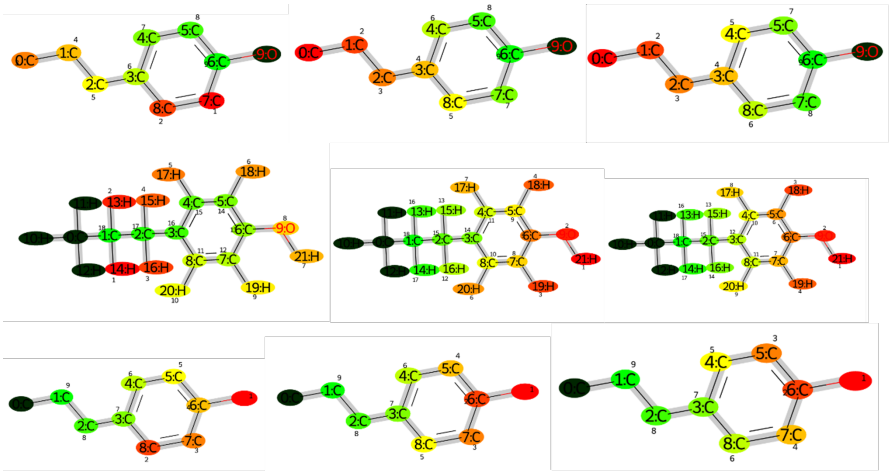


Figure 7.4: mutation route for 6-propylphenol for 6-propylphenol->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible

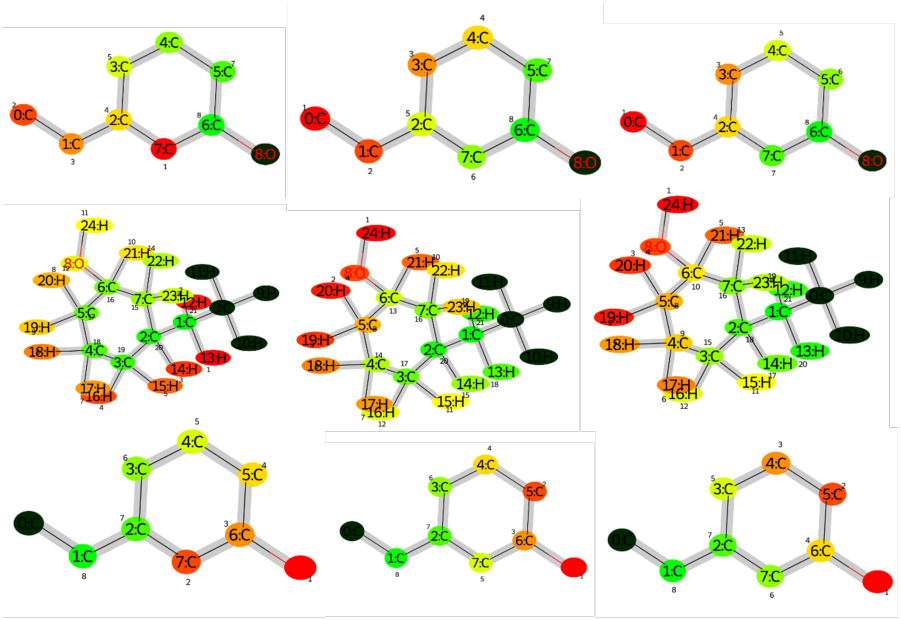


Figure 7.5: mutation route for ethylcyclohexane for ethylcyclohexane->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible

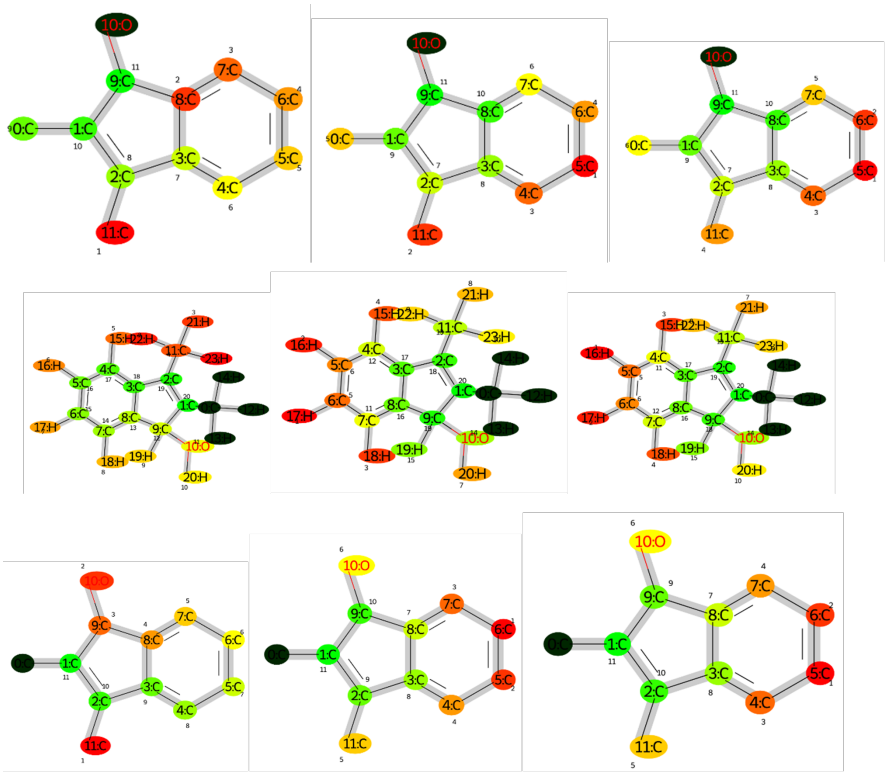


Figure 7.6: mutation route for dimethyl-inden-ol for dimethyl-inden-ol->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible

7.3 Routes for molecules from Transformato paper

SMILES representations of the molecules used:

methane: C
Toluene: CC1=CC=CC=C1
2-Methylfuran: CC1=CC=CO1
methylindole2: CC1=CC2=CC=CC=C2N1
2-cyclopentylindole: C1CCC(C1)C1=CC2=CC=CC=C2N1
7-cyclopentylindole: C1CCC(C1)C1=C2NC=CC2=CC=C1

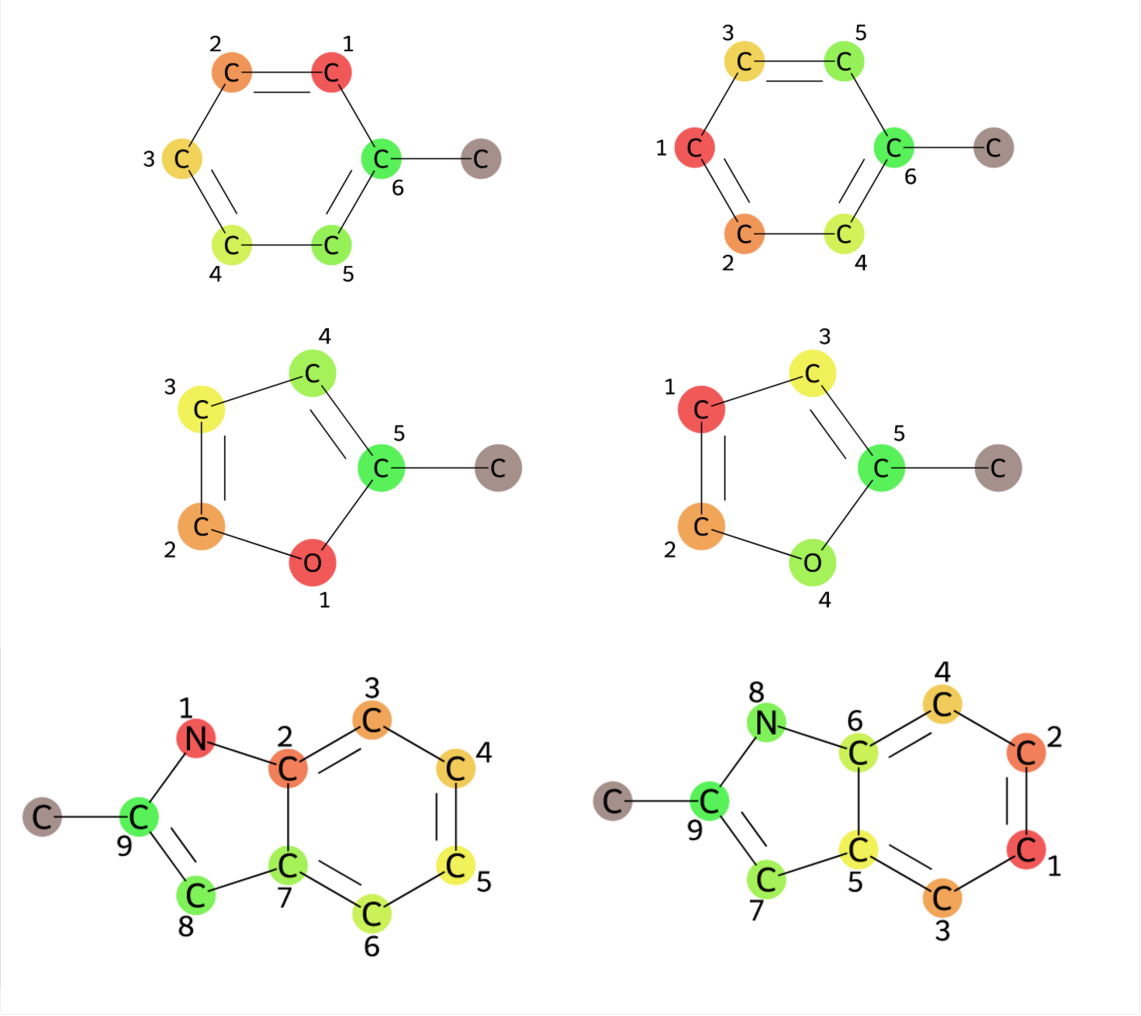


Figure 7.7: left: DFS-algorithm; right: BFS–algorithm; common core in dark; from top to bottom row: mutation routes for toluene/methane, 2-methylfuran/methane, 2-methylindole/methane

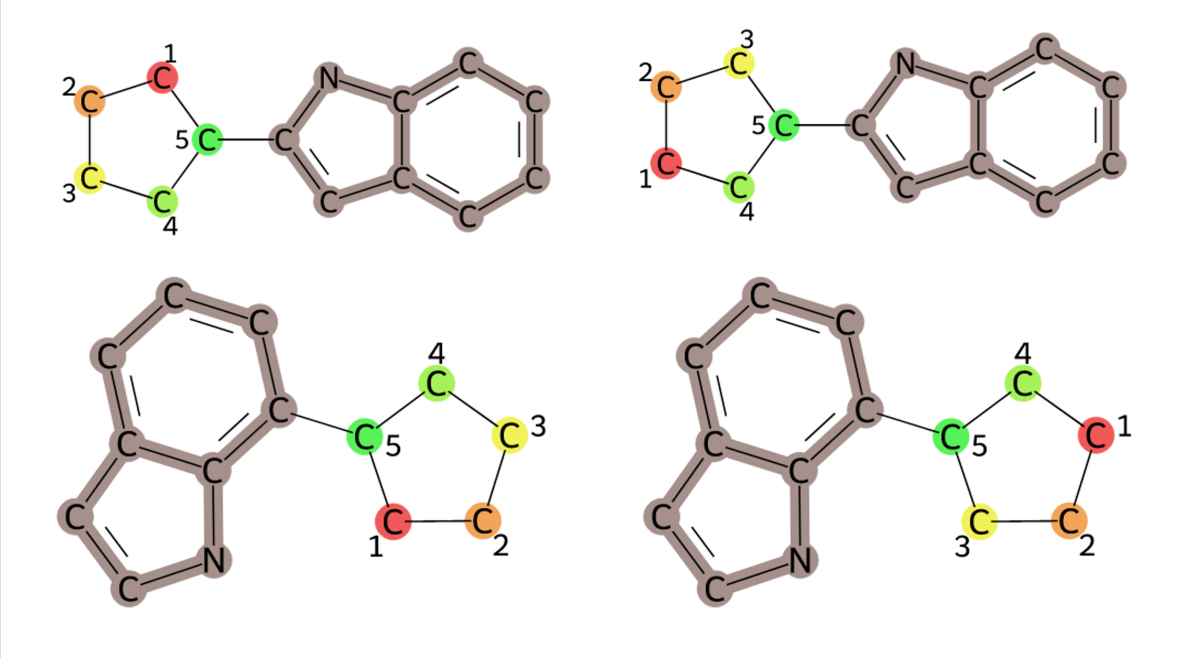
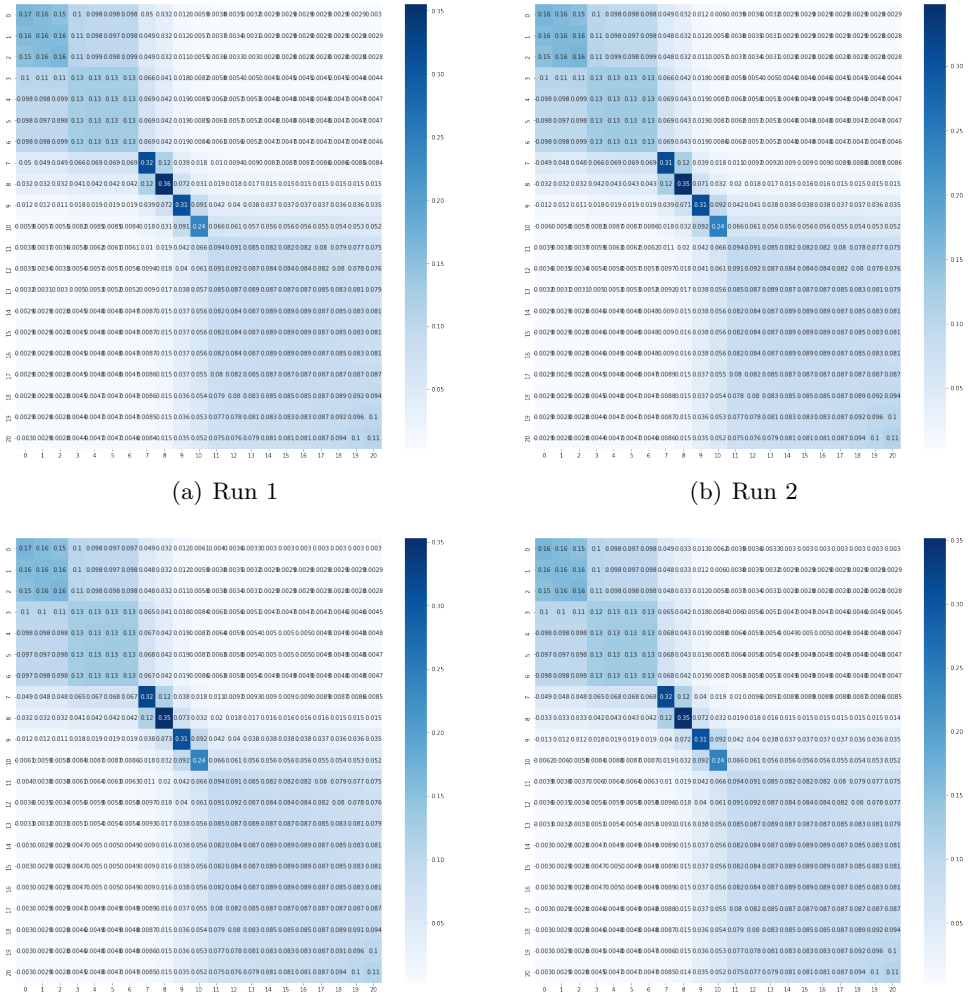


Figure 7.8: left: DFS-algorithm; right: BFS-algorithm; common core in dark; mutation routes for 2-cyclopentylindole/7-cyclopentylindole

7.4 Results for individual runs



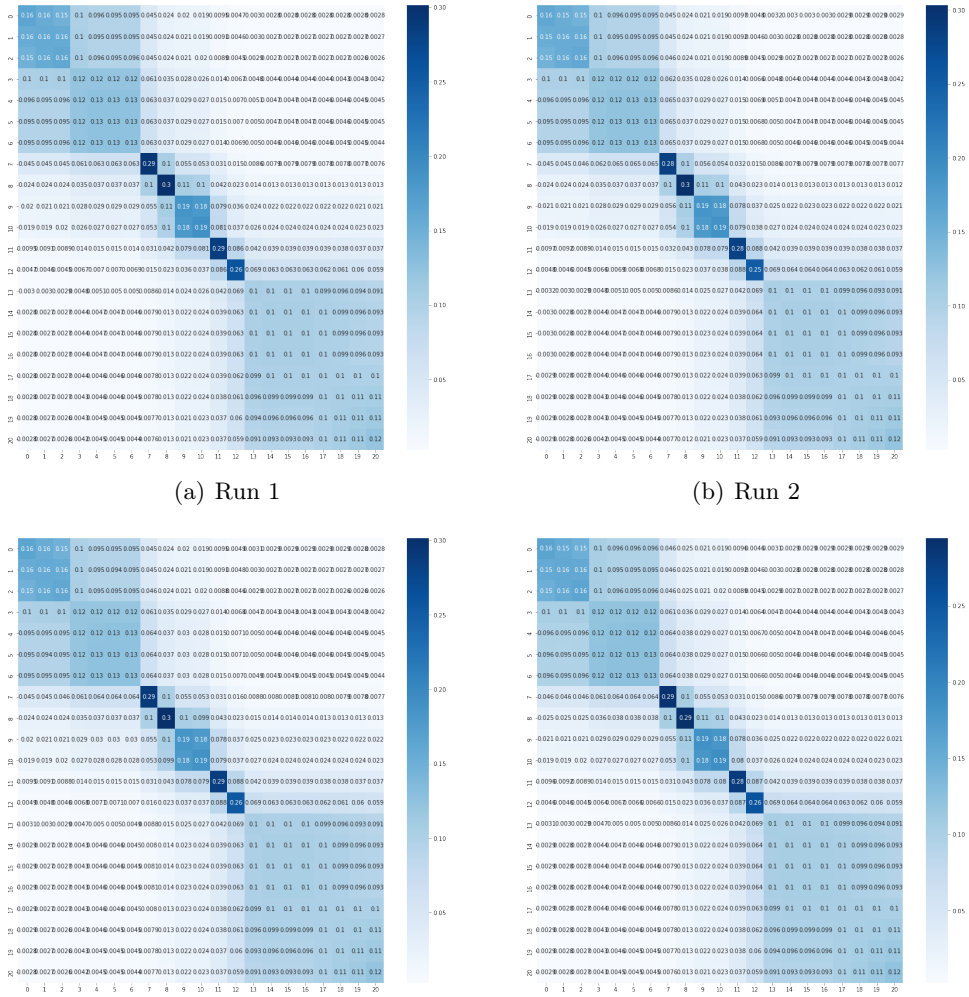
(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.9: Overlap plots for Naphthol/Vacuum, old algorithm



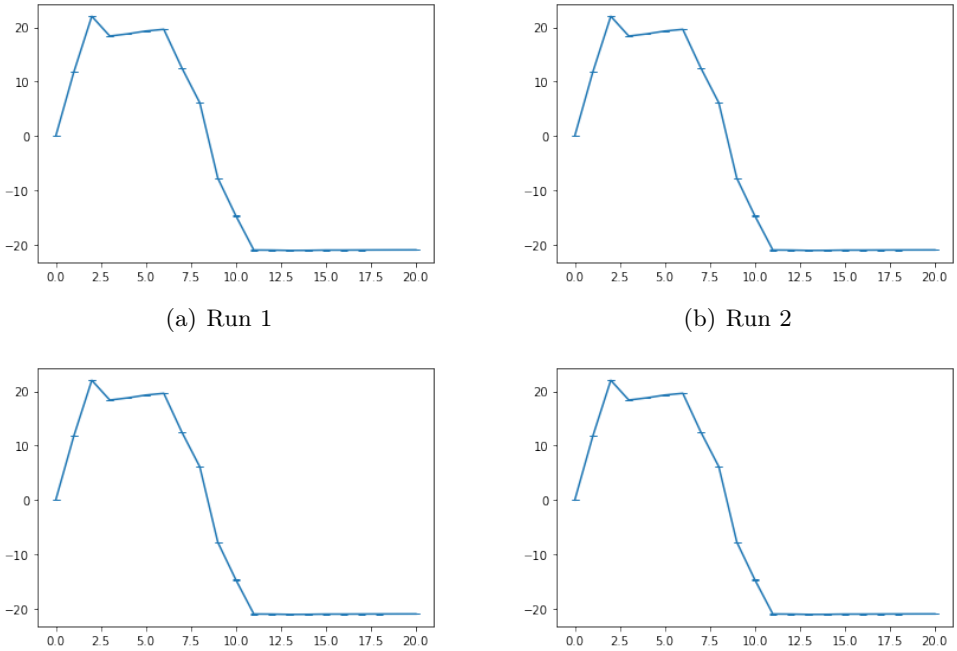
(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.10: Overlap plots for Naphthol/Vacuum, new algorithm



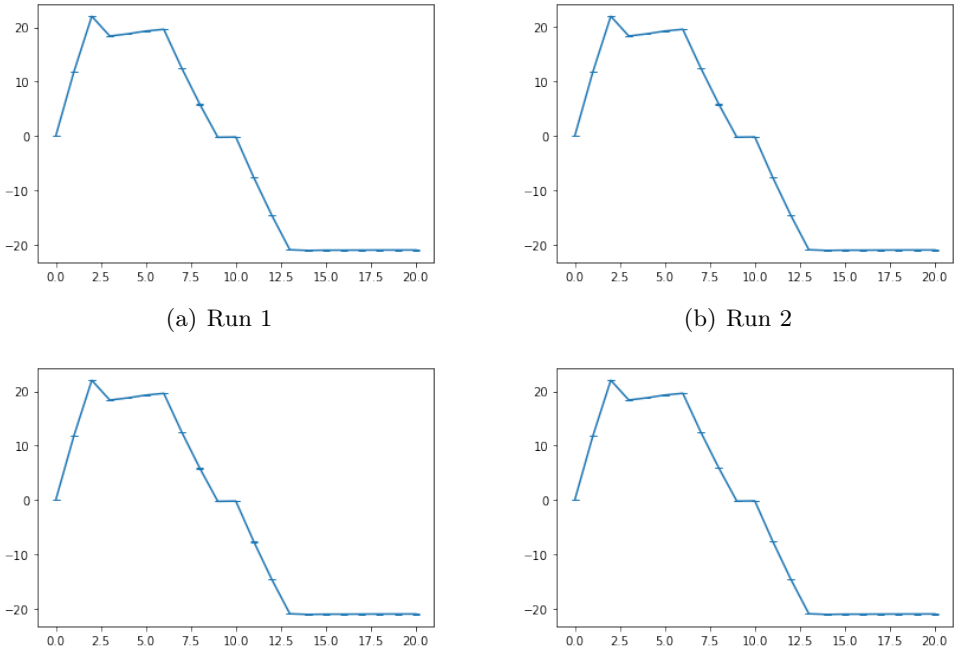
(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.11: Lambda states for Naphthol/Vacuum, old algorithm



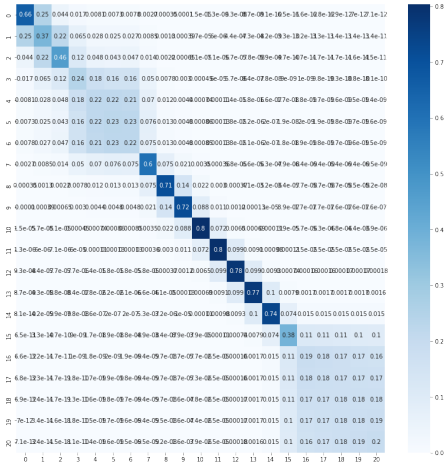
(a) Run 1

(b) Run 2

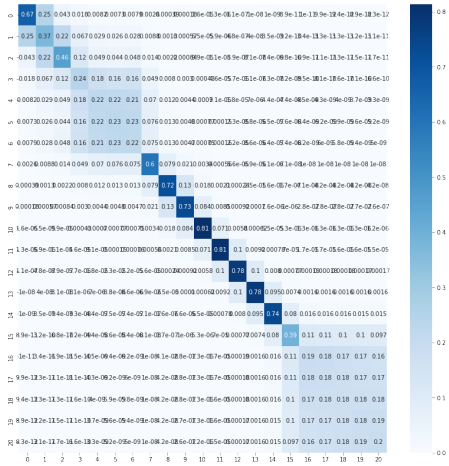
(c) Run 3

(d) Run 4

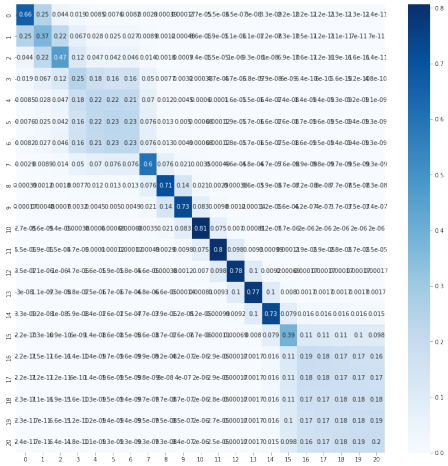
Figure 7.12: Lambda states for Naphthol/Vacuum, new algorithm



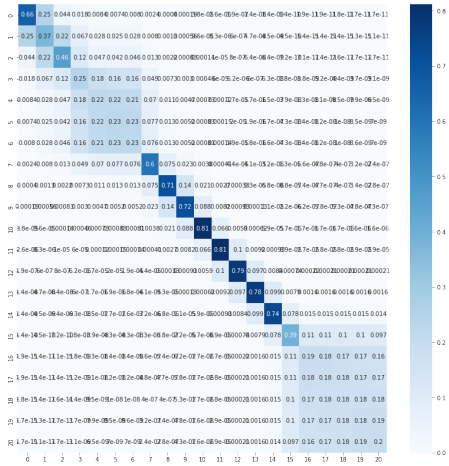
(a) Run 1



(b) Run 2

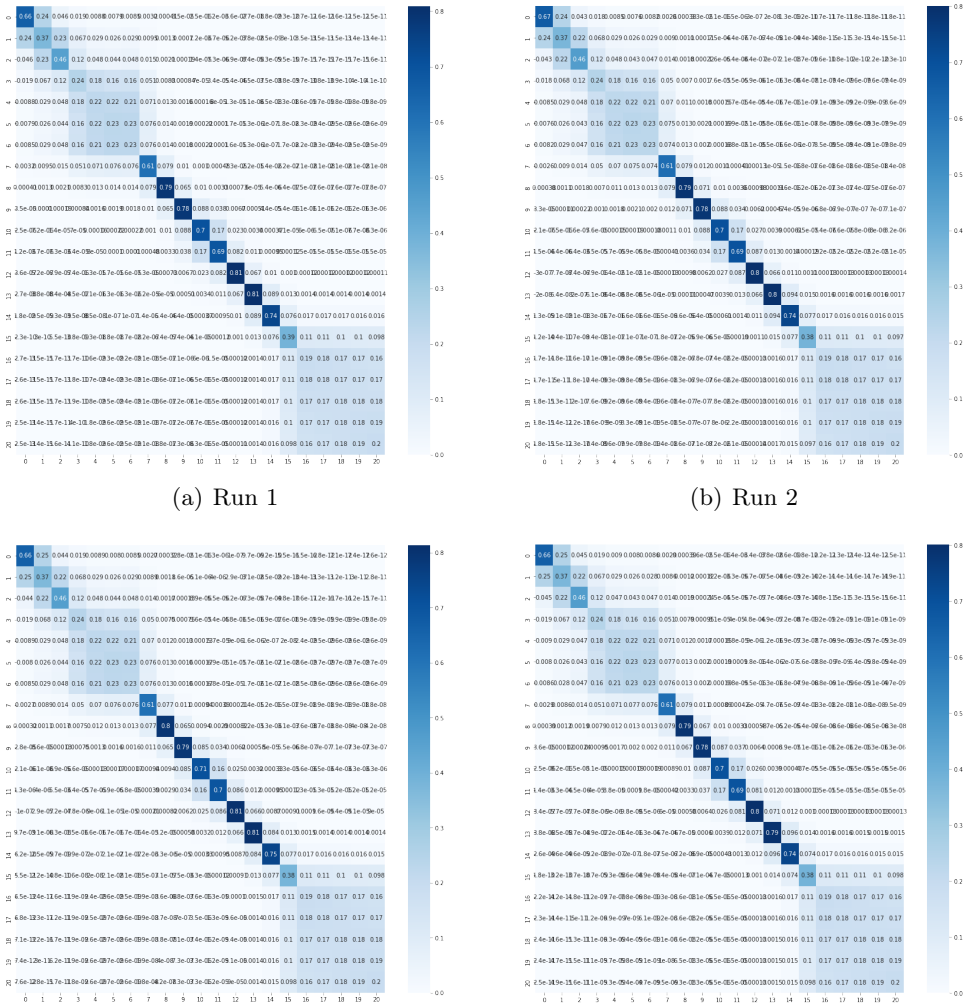


(c) Run 3



(d) Run 4

Figure 7.13: Overlap plots for Naphthol/Waterbox, old algorithm



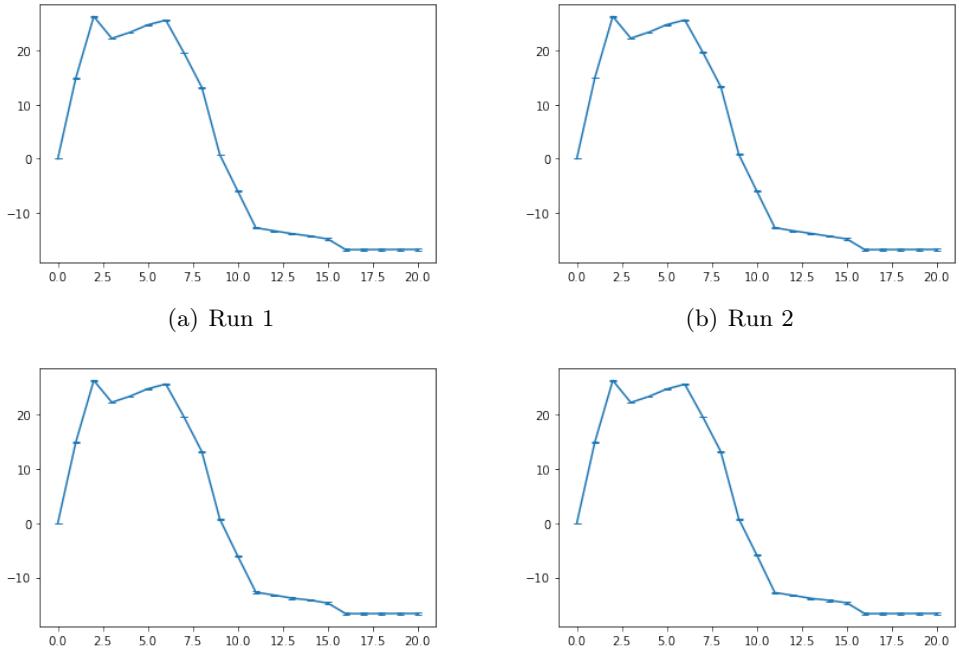
(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.14: Overlap plots for Naphthol/Waterbox, new algorithm



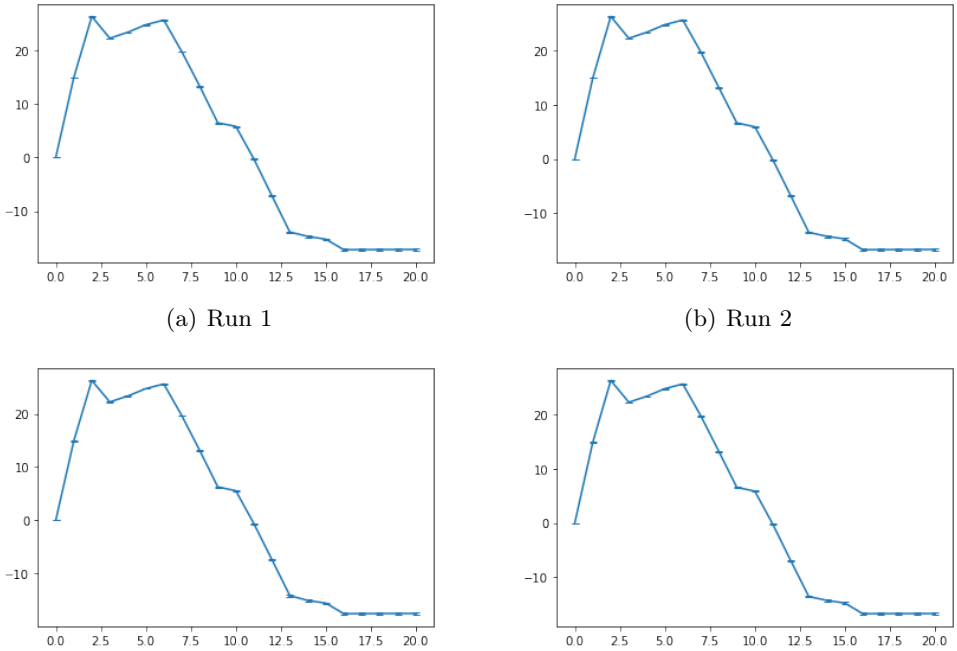
(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.15: Lambda states for Naphthol/Waterbox, old algorithm



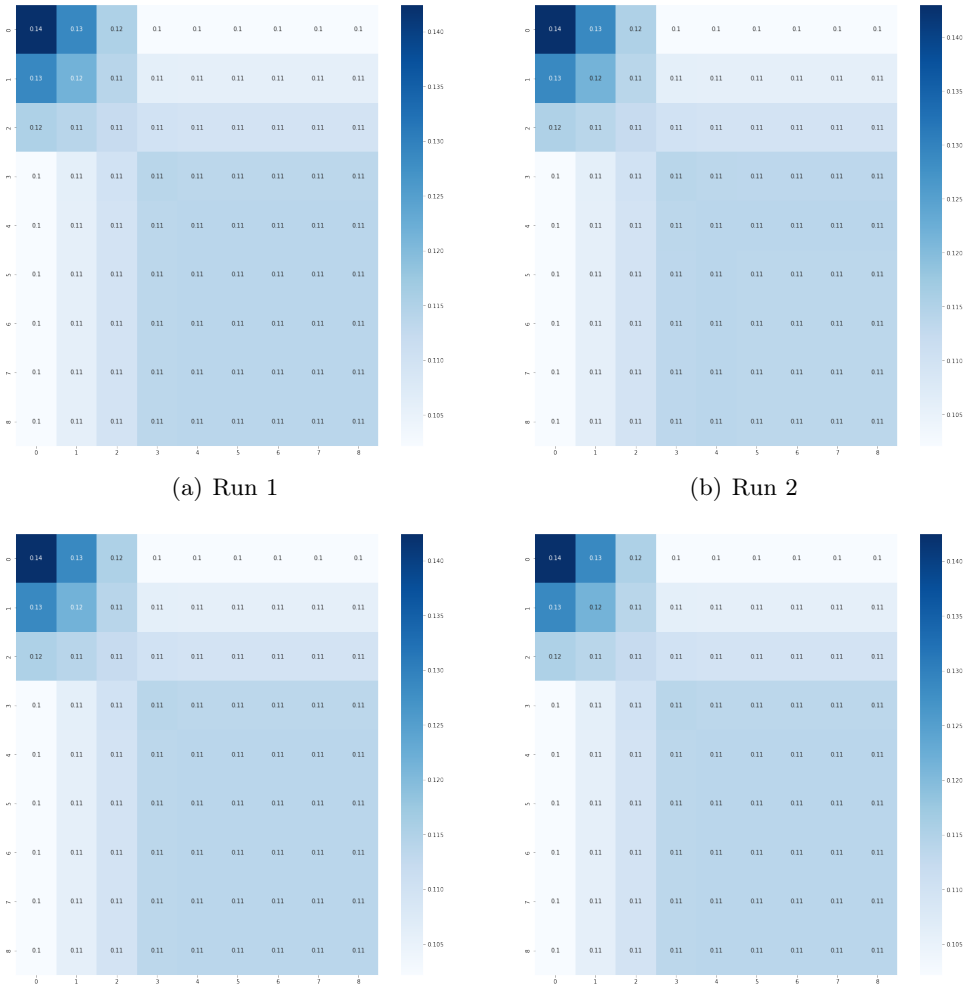
(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.16: Lambda states for Naphthol/Waterbox, new algorithm



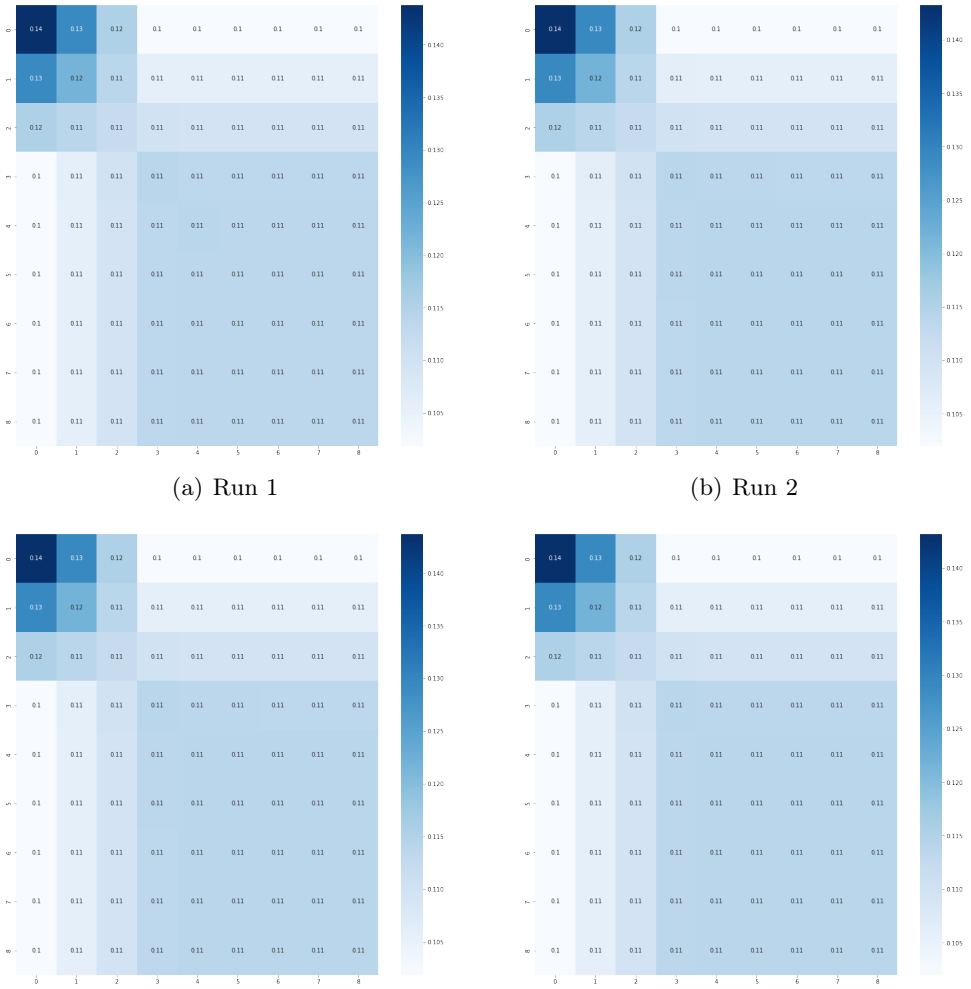
(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.17: Overlap plots for Methanol/Vacuum (from transformation 2-naphthol->methanol), old algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.)



(a) Run 1

(b) Run 2

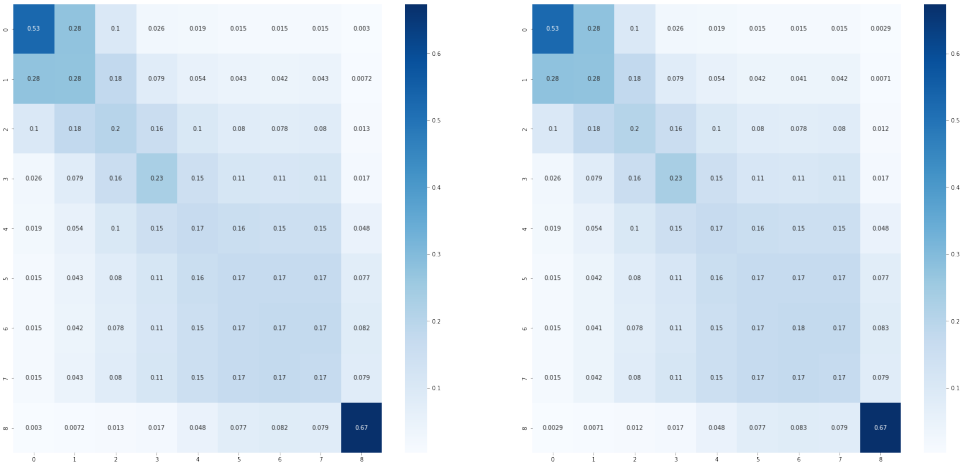
(c) Run 3

(d) Run 4

Figure 7.18: Overlap plots for Methanol/Vacuum (from transformation 2-naphthol->methanol), new algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.)

(a) Run 1

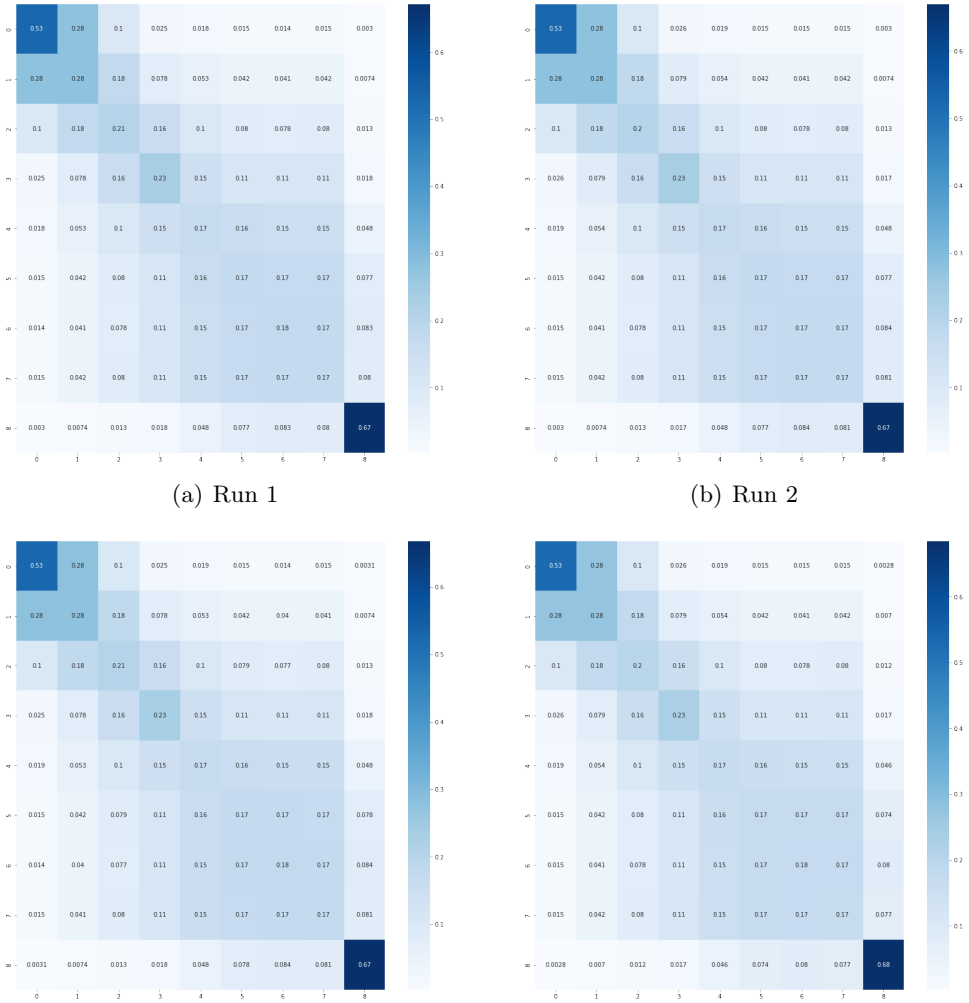
(b) Run 2



(c) Run 3

(d) Run 4

Figure 7.19: Overlap plots for Methanol/Waterbox (from transformation 2-naphthol->methanol), old algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.)



(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure 7.20: Overlap plots for Methanol/Waterbox (from transformation 2-naphthol->methanol), new algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.)

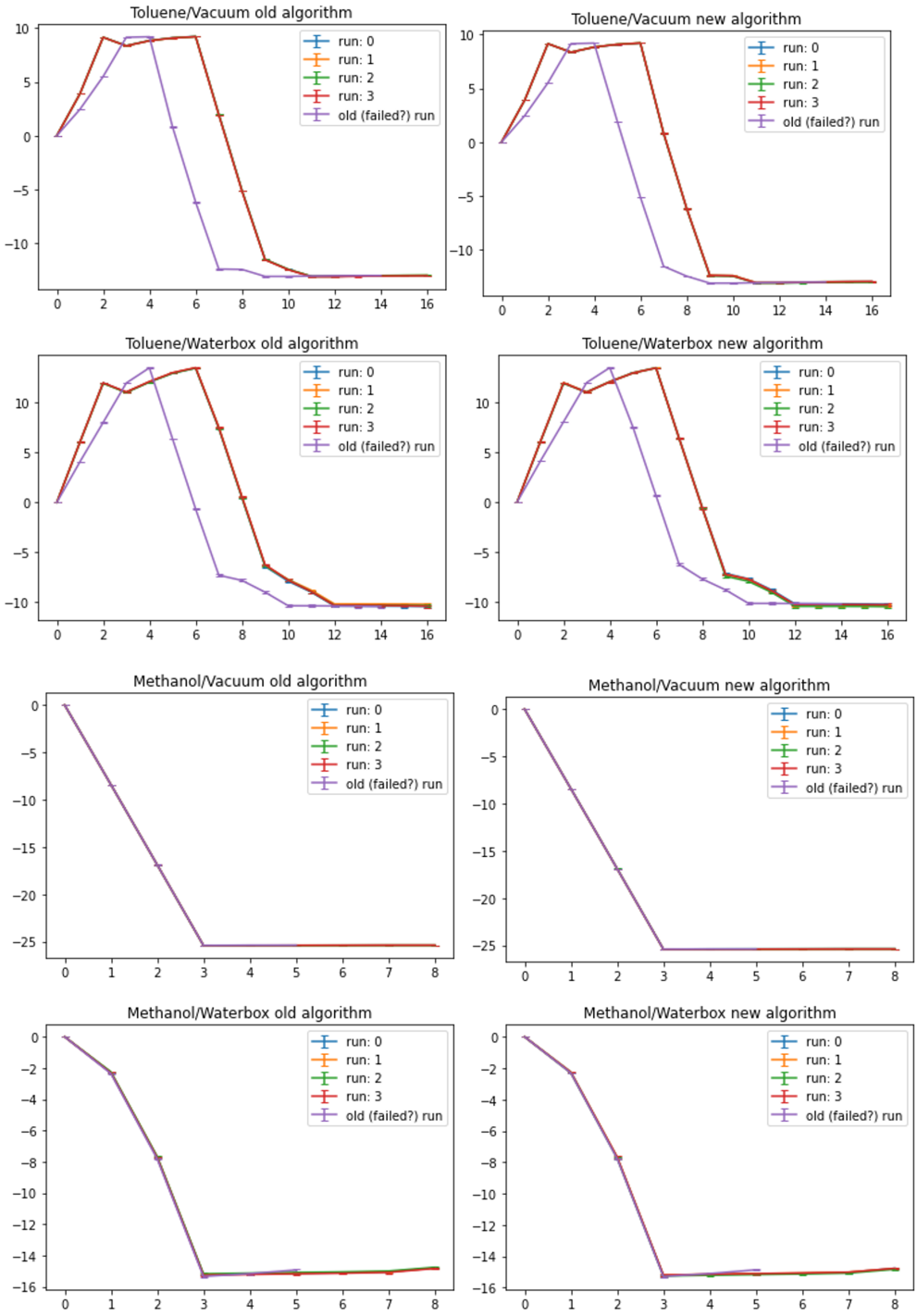


Figure 7.21: comparison of the lambda states of the first simulation (possibly wrong results) and 4 runs of second simulation for toluene->methanol

List of Figures

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) 4

4.1 These examples demonstrate that the addition of hydrogens can influence the construction of the common core; left: molecule 1; middle: common core; right: molecule 2; in the representations of the full molecule, the common cores are marked in red; the first and the last two rows show the same molecules, in the upper row without, in the lower with hydrogens, in case of the lower molecule combination the common core changes when hydrogens are added to the Rdkit-molecule representation 15

4.2 'Strict fusion' can affect the construction of the common core drastically; first and second column: Common core of two molecules (cholesterol and cortisol) without strict fusion; third and fourth column: Common core of two molecules with strict fusion; in the images of the full molecule graph common cores are marked in red. In this case, none of the settings yield a valid common core, the iterative approach would be necessary to obtain one 16

4.3 common cores of cholesterol (left) and 1-pyrenepropanoic acid (right); upper row: CompleteRingsOnly = False; middle row: CompleteRingsOnly = True; lower row: iterative approach to obtain a valid Transformato common core 17

4.4 Comparison of different graph traversal algorithms; green nodes represent the root atom, blue atoms (in the figure shown right) have increased weights; left: depth first search; middle: breadth first search; right: Dijkstra algorithm, the edges connecting blue colored nodes have increased weights leading to a mutation route differing from BFS; the final processing of the nodes happens in reversed order 18

4.5 comparison of typical DFS- and BFS-mutation route for a single ring structure; left: DFS-algorithm; right: BFS-algorithm; in all depictions of mutation routes the common core is shown in dark, the color gradient starts at red indicating the atom first to be removed and ends at green; DFS starts at carbon 22 and thus ring breakage gives rise to one long chain which is processed subsequently, whereas BFS starts at carbon 22 and the two emerging chains are processed in a symmetric fashion 21

4.6 Example for the differences between iter- and iter_change-algorithm; left: iter; right: 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 . 22

4.7 left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms 23
4.8 left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow and circle indicates the undesired processing of the ring atoms . . . 23
4.9 left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms 24
4.10 left: DFS-algorithm; right: BFS-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms 24
4.11 left: DFS-algorithm; right: BFS-algorithm; common core in dark 24

5.1 Visualization of the mutation route using Py3dmol; upper row: Rdkit-representations of both molecules and the common core; lower row: Py3Dmol visualizations; left and right: molecules; middle: common core of both molecules 26
5.2 Betweenness centrality 27
5.3 Closeness centrality 28
5.4 Ring-related scores 30
5.5 differences between old and new algorithm for selected molecules / 2. VERSUCH, muss ebenfalls nochmals ueberprueft werden 32
5.6 Overlap plots for naphthol: upper row: vacuum, lower row: waterbox; left: old mutation algorithm, right: new mutation algorithm 33
5.7 Free energy states at different lambda states for naphthol: upper row: vacuum, lower row: waterbox; left: old mutation algorithm, right: new mutation algorithm 34
5.8 Overlap plots for toluene: upper row: vacuum, lower row: waterbox; left: old mutation algorithm, right: new mutation algorithm 35

7.1 left: DFS-algorithm; middle: BFS-iter-algorithm; right: BFS-iter+-algorithm; common core in dark 39
7.2 Toluene->Methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible 40
7.3 mutation route for 2-naphthol for 2-naphthol->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible 41

7.4 mutation route for 6-propylphenol for 6-propylphenol->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible 42
7.5 mutation route for ethylcyclohexane for ethylcyclohexane->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible 43
7.6 mutation route for dimethyl-inden-ol for dimethyl-inden-ol->methanol; first row: correct common core and mutation route, hydrogens excluded; middle row: hydrogens included; right: hydrogens included, but removed before drawing; left: DFS-algorithm; middle: BFS-algorithm; right: BFS-iter-algorithm; common core in dark, for these small molecules the differences between the new mutation algorithms (i.e. middle and right) are negligible 44
7.7 left: DFS-algorithm; right: BFS–algorithm; common core in dark; from top to bottom row: mutation routes for toluene/methane, 2-methylfuran/methane, 2-methylindole/methane 45
7.8 left: DFS-algorithm; right: BFS-algorithm; common core in dark; mutation routes for 2-cyclopentylindole/7-cyclopentylindole 46
7.9 Overlap plots for Naphthol/Vacuum, old algorithm 47
7.10 Overlap plots for Naphthol/Vacuum, new algorithm 48
7.11 Lambda states for Naphthol/Vacuum, old algorithm 49
7.12 Lambda states for Naphthol/Vacuum, new algorithm 50
7.13 Overlap plots for Naphthol/Waterbox, old algorithm 51
7.14 Overlap plots for Naphthol/Waterbox, new algorithm 52
7.15 Lambda states for Naphthol/Waterbox, old algorithm 53
7.16 Lambda states for Naphthol/Waterbox, new algorithm 54
7.17 Overlap plots for Methanol/Vacuum (from transformation 2-naphthol->methanol), old algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.) 55
7.18 Overlap plots for Methanol/Vacuum (from transformation 2-naphthol->methanol), new algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.) 56
7.19 Overlap plots for Methanol/Waterbox (from transformation 2-naphthol->methanol), old algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.) 57
7.20 Overlap plots for Methanol/Waterbox (from transformation 2-naphthol->methanol), new algorithm (As only one heavy atom is involved, the mutation algorithm obviously plays no role.) 58

7.21 comparison of the lambda states of the first simulation (possibly wrong results) and 4 runs of second simulation for toluene->methanol 59

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

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

[27] 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.

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

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

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

[31] 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.