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

The aim of this Master Thesis is to facilitate the preparation of alchemical free energy calculations. Such free energy calculations estimate free energies 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 differences, e.g., the 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[FWB21].

This Master Thesis works on specific methods of employing these unphysical atoms. A central part will consist in 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[keya]. In particular, additional functions will optimize the employment of the aforementioned dummy atoms.

In the next chapter, the basic principles of alchemical free energy calculations are explained. The third section presents the workflow of Transformato. Subsequently, the tasks for improving the software package are 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 are discussed.

2 Free Energy Calculations

2.1 Basics

In the last years, accuracy and feasibility of free energy calculations improved significantly [KALL21]. Main reasons are developments in the accuracy of force fields[CAS17], 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, CHARMM or GROMACS, offer functions for alchemical free energy calculations which facilitates the set-up of the necessary simulations.

Possible applications can be found in rational drug design and drug discovery; e.g. in lead optimization, the binding free energy differences between compounds are of interest.[CAS17]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 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)[BTLK03][JBBTR88], 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 [KALL21].

Relative free energy calculations harness the concept of the thermodynamic cycle [Kol93](fig. 2.1): The horizontal arrows indicate the path from the unbound to the bound state of one of the two ligands, the vertical arrows indicate the transformation from one molecule to the other one. According to the thermodynamic cycle, 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, i.e. the energy difference has to be 0, hence

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

The change in free energy is assessed by the double free energy differences:

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

[CAS17]. To obtain knowledge about $\Delta G_B - \Delta G_A$, the evaluation of the alchemical transformations $\Delta G_1$and ΔG_2 is hence sufficient. (In practice, the determination of ΔG_A or ΔG_B usually would require an experimental set-up, thus the alchemical calculation

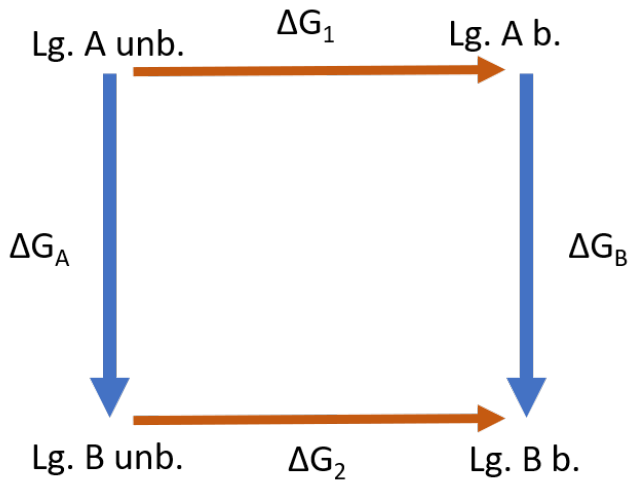

Figure 2.1: Thermodynamic cycle; red arrows indicate transitions between two states (unbound - bound) of each ligand, blue arrows indicate 'alchemical' transformations (Ligand A - Ligand B)

could 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 (or the evaluation of the vertical legs needs experimental determination) to compute because the change between both states is much smaller (depending on the molecules of interest, only some atoms have to be annihilated or created) 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. Hence the free energy difference between states i and j can be described as:

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

, where Z denotes the partition function $Z = \int dr \exp(-\beta U)$ with $\beta = \frac{1}{k_B T}$[SM13].

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 sum of the difference between these intermediate states: $\Delta F = F_1 - F_0 = \sum_n \Delta F_n$ [MAM+20].

2.2 Methods for evaluating free energy differences

Various approaches for calculating free energy differences exist, e.g. thermodynamic integration. Alternative methods are implementation using the Zwanzig formula or Bennett's acceptance ratio (which is used in the Transformato package described below). In the following, these three basic approaches will be shortly described. (For a more comprehensive comparison and estimation of performance differences see [BB11b] and [dRBO13]).

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 states.Thermodynamic Integration 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. $0 > \lambda < 1$, gives rise to the unphysical, 'alchemical' intermediate states.

Taking the derivative of the free energy, one gets $\frac{dF}{d\lambda} = \frac{d}{d\lambda} \int e^{-\beta U} = \left\langle \frac{dU(r,\lambda)}{d\lambda} \right\rangle_\lambda$ [SM13]. The free energy difference is then given by the integral between both final states

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

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[SM13]. (For a comparison of various numerical quadrature schemes, e.g. trapezoidal rule and Simpson's rule, see [BB11c]).

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. The free energy difference is then given by

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

, where the angular brackets indicate the ensemble average [GMP+15].

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 [BB11b].

As for thermodynamic integration, usually it will be 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[BW17] and there exists an extension to non-equilibrium work (Jarzynski's equation)[BW17], but usually it should be only used if one knows that the difference between the samples is very small[SM13].

2.2.3 Bennett Acceptance Ratio (BAR)

A different approach for computing the free energy difference between two states is the usage of Bennett's Acceptance Ratio[Ben76].

Extension 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 $W\left(q_1 ... q_n\right) = c\left(\frac{Q_0}{n_0}\exp\left(-U_1\right) + \frac{Q_1}{n_1}\exp\left(-U_0\right)\right)^{-1}$[Ben76].

For using 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 totally equivalently.

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 $f\left(x\right) = \frac{1}{1+\exp(\beta x)}$ [BB11b][GMP+15].

To obtain the value of the optimum shift constant C this self-consistency problem has to be solved iteratively[GMP+15],

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

. The free energy between two λ-states is then given by $\triangle F\left(\lambda_i \to \lambda_j\right) = \beta^{-1} \ln \frac{N_j}{N_i} + C$[BB11b]. 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 approach yielding the formula. It can be shown that the Bennett acceptance ratio method works as a maximum likelihood argument. Using BAR, one obtains the asymptotically unbiased (i.e. for an infinite number of measurements it yields an unbiased estimation) estimation with lowest variance[SBHP03].

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 $M = \beta^{-1} \ln \frac{N_j}{N_i}$ [SBHP03].

Also BAR depends on the overlap between the states, however it is more robust and reliable in cases of rather poor overlap [dRBO13] (especially when compared to FEP). (If 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[SM13].)

2.3 Soft-core potentials

Usually, alchemical transformations rely on a coupling parameter which is used to gradually turn off interactions. (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 van der Waals endpoint problem (or even 'catastrophe') describes several problems which can occur when a particle is removed (i.e. turned into a dummy atom by turning off the intermolecular interactions). [BB11a]

The occurence of this problem can be easily illustrated for the case of a scaling of the coupling parameter describing a linear pathway, 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 [SM13]; 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[BB11a]).

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 division by zero, the fact that 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 [BB11a].

The common way to avoid such problems is the usage of soft-core potentials. An additional term is added to 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 final states of the 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. [FWB21] 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 adding dummy atoms to the system:

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. [FWB21]

The implementation of these dummy atoms, however, is not without pitfalls. Using single topology errors caused by kept bonded interactions can emerge. 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. [FWB21]

2.5 Serial atom insertion

Alternatively to modifications of the coupling parameter, one can try to create new states by turning off atoms in one step without intermediate values. In [BB11a], this approach is called Serial atom insertion. Atoms are turned off serially (one after one or in 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 necessary condition. The feasibility of this approach relies on Bennett's acceptance ratio which is proven to can work with neighboring states created by serial atomic insertion [BB11a].

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.

It seems that most severe restriction due to the van der Waals Endpoint problem is the instability of the integrator near 0. As such states are not used in Serial atom insertion, this problem is automatically circumenvented.

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 atoms 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 attained.

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 final 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 common 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 usage of softcore potentials. The computation of the resulting energy differences is based on Bennett's acceptance ratio described above.

Therefore, setting up the mutation path between two molecules across the connecting common core and going along by means of serial atom insertion, the Transformato workflow is independent of the underlying molecular dynamics package and 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[JKII08][Bra21].

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:

So the junction between common core and dummy region has to be unique; 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).

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 have been compared.

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

Particularly intricate problems occur for ring structures. There are some especially sensitive cases, e.g. ring breakage poses especially difficult problems because it can lead to fast changes in free energy and cause significant estimation error [LWM15]. 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 has been 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 had to be validated by means of molecular dynamics simulations.

To obtain a reliable test set of molecules, sdf-files of ligands have been downloaded from PDB-BIND. 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 have also been used for molecule processing and graph representations. The creation of molecule objects and the determination of the maximum common substructure is done via rdkit[keyc]. NetworkX[ADP08] provides functions for graph visualization and analysis. It is easily possible to convert molecules created using rdkit into networkx graph-objects and hence utilize the functions of networkx for the molecules and common cores constructed.

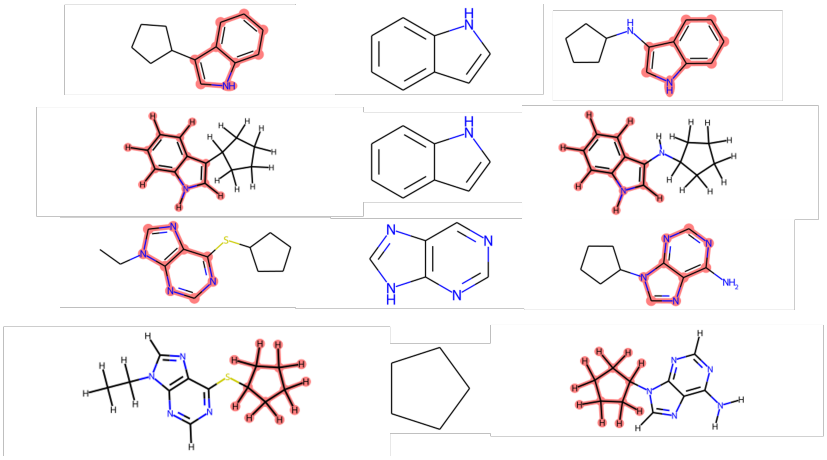

Figure 4.1: left: molecule 1; middle: common core; right: molecule 2; the first and the last two rows show the same molecules, in the upper row without, in the lower with hydrogens, in case of the lower molecule combination the common core changes when hydrogens are added to the Rdkit-molecule representation

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 of 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).

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, can even be decisive if the generated common core is valid for the Transformato workflow.

Important ring-related settings are ringMatchesRingOnly, completeRingsOnly and the ringCompare-parameter.

To obtain valid common cores 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

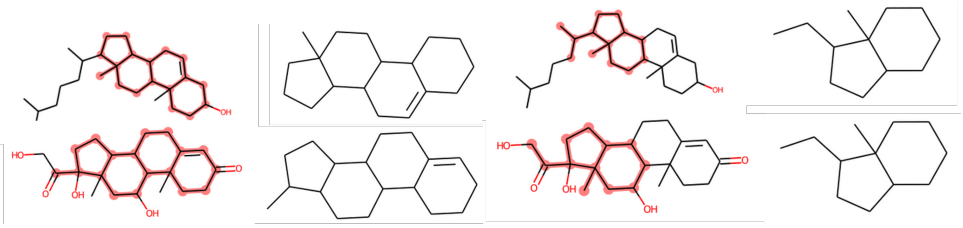

Figure 4.2: First and second column: Common core of two molecules (cholesterol and cortisol) without strict fusion; third and fourth column: Common core of two molecules with strict fusion

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 which are not valid Transformato common cores because a dummy region is connected via multiple ring atoms of the same ring with the common core.

(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. Alternatively, one could 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. Alternatively, one could prohibit the involvement of ring atoms for this particular molecule combination. Probably the most efficient solution would be to remove the rings which contain atoms which participate in the partial ring causing the invalid common core from the representation used for creating the maximum common substructure and afterwards start a new search for a valid common core. (Of course, in the worst scenario, i.e. if both molecules only consist of ring-participating atoms, no valid common core is conceivable.) At the moment, 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.

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

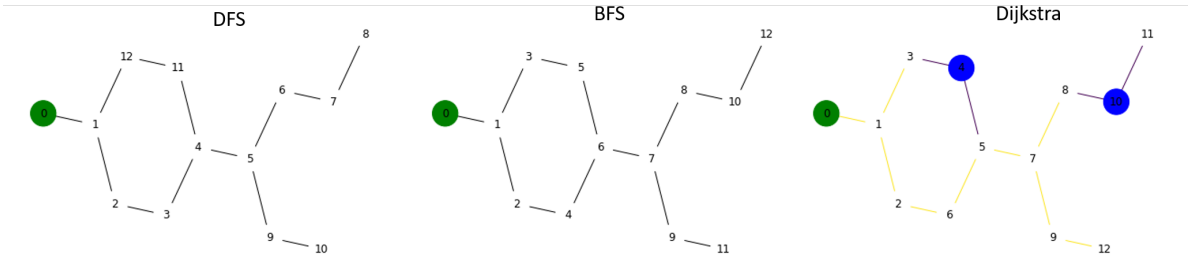

Figure 4.3: Comparison of different graph traversal algorithms; left: depth first search; middle: breadth first search; right: Dijkstra algorithm, the edges connecting blue colored nodes have increased weights leading to a mutation route differing from BFS; the final processing of the nodes happens in reversed order

simultaneously.[EE12] (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 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 list of mutations orders has to be reversed.

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

4.3 Added functionality

4.3.1 Functions

To use the newly implemented mutation algorithms, initially the graph is endowed with weights stored in a dictionary. The simulations shown below use uniform weights, however it is possible to modify them to enforce a certain mutation route (e.g. accelerating or postponing the exclusion of specific 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, additional to the simple DFS-approach which leads to undesired outcomes. 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` has already been implemented 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 of nodes to be 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 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, respect-

ively, 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 a 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_ite` 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, even in comparison to the computational time the creation of the common core needs). This has the advantage that after each mutation step weights can be updated or even the search algorithm modified.

The latter option allows for different mutation strategies depending on the current state. This 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 both iter-algorithms. 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`).

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

4.3.2 Examples for processing molecules

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.

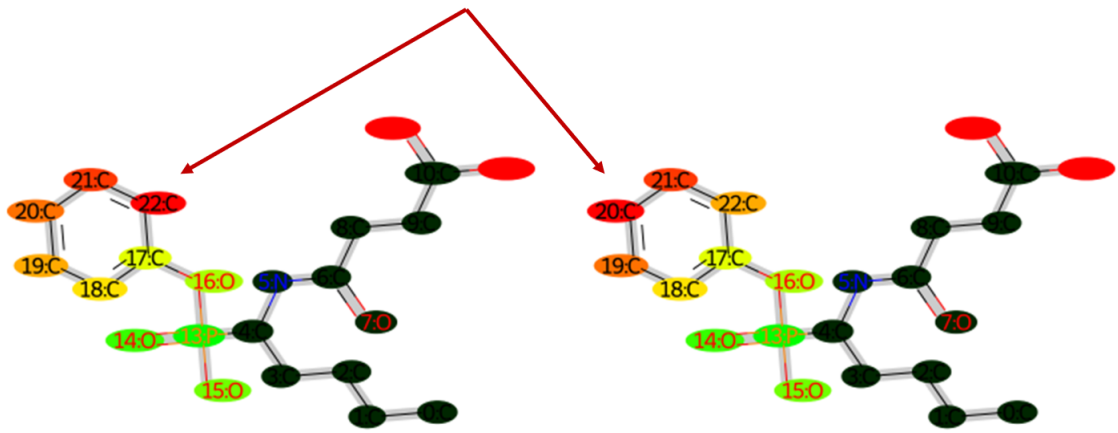

Figure 4.4: left: dfs-algorithm; right: bfs-algorithm; common core in dark; dfs starts at carbon 22 and thus ring breakage gives rise to one long chain which is processed subsequently, whereas bfs starts at carbon 22 and the two emerging chains are processed in a symmetric fashion

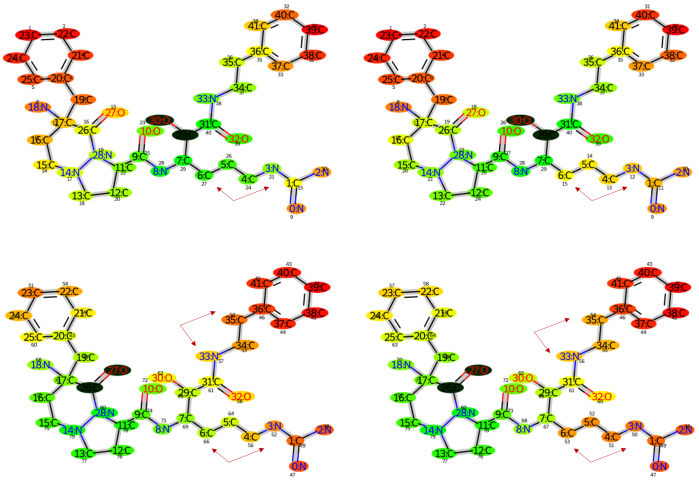

Figure 4.5: Example for the differences between iter- and iter_change-algorithm; left: iter; right: iter-change; iter-change processes all atoms within a chain or a ring at once (if possible) before switching to other parts of the molecule

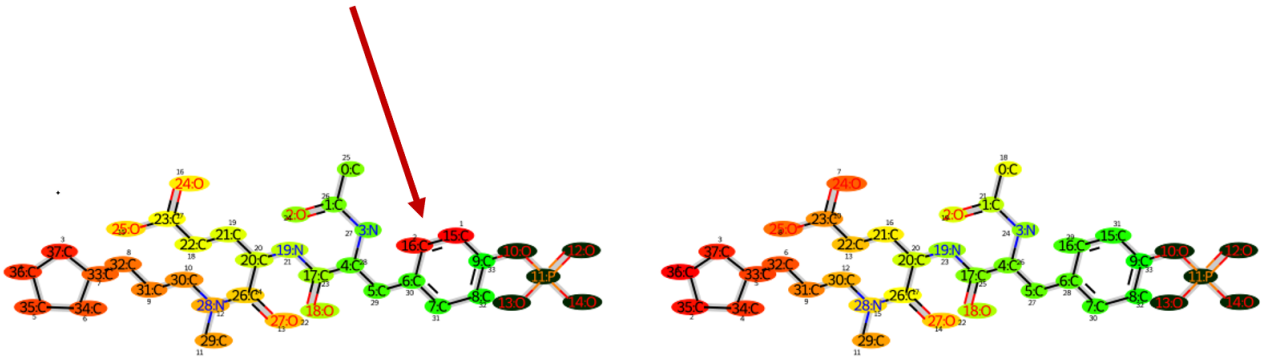

Figure 4.6: left: dfs-algorithm; right: bfs-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms

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

As fig. 4.8 shows, also the processing of substituents is affected.

Multiple rings pose special problems for the mutation algorithms because the processing 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).

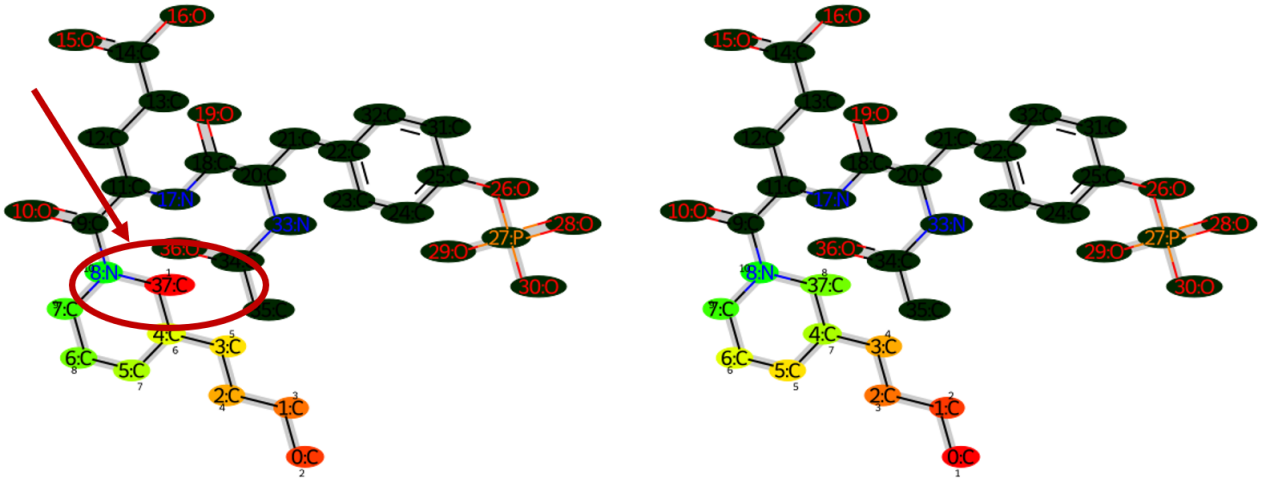

Figure 4.7: left: dfs-algorithm; right: bfs-algorithm; common core in dark; the red arrow and circle indicates the undesired processing of the ring atoms

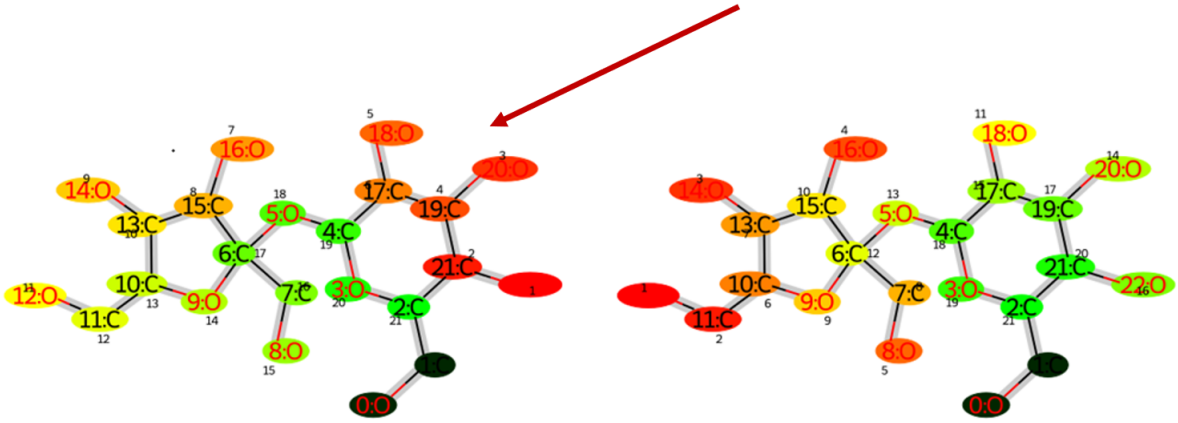

Figure 4.8: left: dfs-algorithm; right: bfs-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms

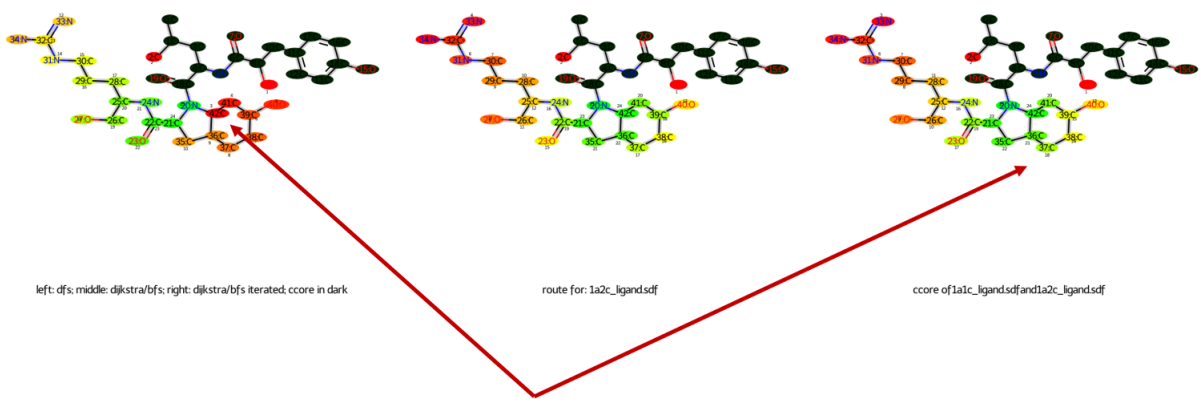

Figure 4.9: left: dfs-algorithm; right: bfs-algorithm; common core in dark; the red arrow indicates the undesired processing of the ring atoms

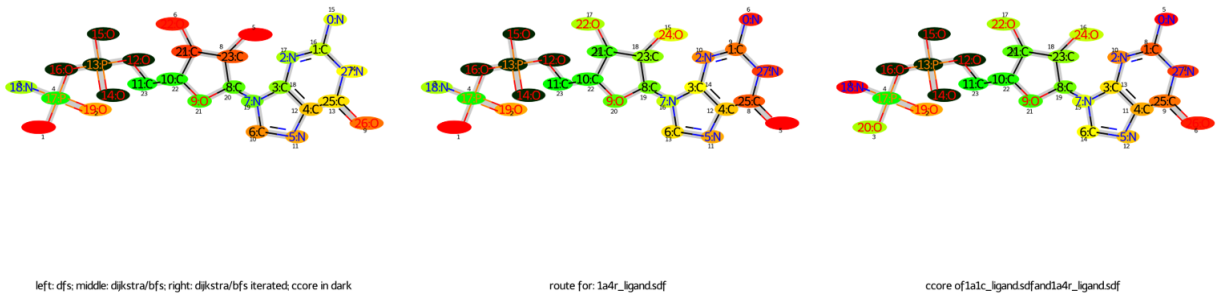

Figure 4.10: left: dfs-algorithm; right: bfs-algorithm; common core in dark

5 Results

A set of ligands from PDBbind 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.

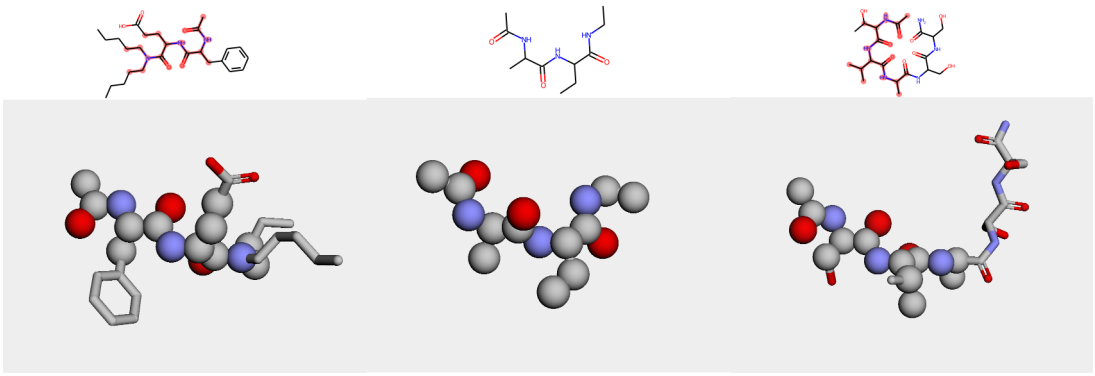

Figure 5.1: Visualization of the mutation route using py3dmol; upper row: rdkit-representations of both molecules and the common core; lower row: py3Dmol visualizations; left and right: molecules; middle: common core of both molecules

6 Conclusion

....

List of Figures

Bibliography

[ADP08] 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.

[BB11a] 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.

[BB11b] 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.

[BB11c] 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.

[Ben76] Charles H. Bennett. Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, (22):245–268, 1976.

[Bra21] Benedict Braunsfeld. *Implementation and Testing of CHARMM as Backend to the Free Energy package Transformato.* Master's thesis, University of Vienna, Vienna, 2021.

[BTLK03] 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.

[BW17] Stefan Boresch and H. Lee Woodcock. Convergence of single-step free energy perturbation. *Molecular Physics*, 115(9-12):1200–1213, 2017.

[CAS17] 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.

[dRBO13] 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.

[EE12] Shimon Even and Guy Even. *Graph algorithms*. Cambridge University Press, Cambridge, second edition edition, 2012.

[FWB21] 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.

[GMP+15] 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.

[JBBTR88] 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.

[JKII08] 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.

[KALL21] 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.

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

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

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

[Kol93] Peter Kollman. Free energy calculations: Applications to chemical and biochemical phenomena. *Chem. Rev.*, (93):2395–2417, 1993.

[LWM15] 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.

[MAM+20] 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.

[SBHP03] 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.

[SM13] 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.