Random State Selection in Proton-Transfer Transition Networks

Dennis Dörrich

Matrikelnummer: 4667460

Mono-Bachelor of Science in Physics

dennis.doerrich@fu-berlin.de

August 23, 2016

Supervisor:

Prof. Dr. Petra Imhof

Marco Reidelbach

Bachelor-Thesis at Free University of Berlin

submitted to the department of physics at Free University of Berlin

Abstract

Random state selection in transition networks provides an alternative sampling method to examine proton transfer reactions in molecules. Instead of a uniform sampling over a specified volume in configuration space, the intermediate states of a certain volume are randomly selected. By the implemented algorithm it can be specified how many states per degree of freedom are allowed. This allows to save computational time, as for the same allowed degrees of freedom a much smaller number of initial states are generated and less edges need to be calculated. Proton transfer transition networks of the 5w-model, a dummy model consisting of 4 hydrogen molecules, 1 hydronium molecule and aspartate-like residues at the ends, were generated and computed in order to assess the power of random state selection.

By comparing the best paths from reduced sets among each other it was shown that in the given model the implementation is a powerful method for saving CPU power without loosing valuable results.

Contents

Acknowledgment

I would like to thank my supervisor Marco Reidelbach who never hesitated to share his expertise when I asked for help and professor Petra Imhof who's door were always open for me when the fight against computer bugs seemed to be hopeless alone. Gratitude goes also to my friends and family who enabled a relaxed realization of this work and gave me the drive when needed.

Statement of Originality

This document is written by Dennis Dörrich who declares to take full responsibility for the contents of this document. I declare that the text and the work presented in this document is original and that no sources other than those mentioned in the text and its references have been used in creating it. The department of physics at Free University of Berlin is responsible solely for the supervision of completion of the work, not for the contents.

23rd August 2016

Dennis Dörrich

1 Introduction

The proton transfer in molecules is an important process for nearly all chemical reactions catalyzed by enzymes. It is therefore indispensable for many physiological processes such as immune response, bacterial motions or the function of antibiotics like gramicidin[2]. Cytochrome c oxidase for instance is the final enzyme in the respiratory chain and reduces oxygen to water for which 4 protons need to be transfered through the D and K channel [10]. A better understanding of proton transfer processes can then for example be helpful to design bio-inspired devices which make use of this chemical process[4] or as an MRI technique for imaging the human brain[5]. Whereas the molecule's configuration before and after the reaction is often known, the configuration in between is not known a priori.

In order to explain a molecular transition from one conformation to another computational simulations can give a deeper insight because even movements on a relatively small time scale (femtoseconds) can be simulated. Molecular Dynamic (MD) simulations don't cover bond-breaking and costly Quantum Mechanical (QM) simulations would take years with the computational state of art as more complex transitions like the folding of proteins occurs over a timescale from microseconds up to minutes. The time-scale problem can be overcome by applying a QM-subsystem on the fly [20] for the proton transfer or by supposing the proton moves with a Grotthus-like hop [11].

Using a potential energy function enables the sampling of as many pathways with as few pre-limitations as possible and provides a sufficiently unbiased solution. Instead of directly sampling [6] a huge amount of possible pathways *Tranistion Networks* are a faster and more exact method to find the most favorable paths which is normally equal to the most probable paths to happen.

To understand Transition Networks and what the **path** represents exactly one has to become familiar with **configuration space**, **graph theory** and the **energy landscape** which is treated in the *Theoretical Basics* 2. We can then examine the practical limitations of Transitions networks and how an implementation of random state selection can help.

2 Theoretical Basics

2.1 Transition Networks

Transition Networks allow us to predict a molecule's best paths between two configurations[1], in the following called reactant and product state. Configurations can be represented in *configuration space* which has as many dimensions as degrees of freedom that are necessary to describe the configuration entirely. A symmetric water molecule for instance can be described by two degrees of freedom, the distance between H and O and the angle between the two O-H bonds. By applying an empiric potential energy function[2] [7][16] and assigning an energy value to every point in configuration space we can draw in 3 dimension the corresponding energy landscape (see fig 2.1).

A transition network includes different transition pathways in configuration space. Each pathway is a sequence of subtransitions between two relatively close configuration states. To describe such a pathway in a mathematical way *graph theory* is a very powerful tool.

Graph Theory as a sub-branch of mathematics is an abstract way to represent networks and the inter-correlation of elements within these networks. It is applied to model a wide range of scientific problems, such as in communication, informatics, physical and biological systems. A *graph* describes a set of *nodes* or *vertices* and *edges* whereas one edge always connects two nodes. This pairwise relation between two nodes can be quantified by giving the edge a *weight*. In our model we deal with *simple* and *undirected* graphs which means that only one weight is assigned to an edge and that $w_{i,k} = w_{k,i}$ where $w_{x,y}$ describes the edge connecting node x with node y.

An important problem in many systems is to find the shortest or cheapest *path* between two nodes. In our case this would be the way with the lowest energy barrier to get from reactant to product state. This path might not be the direct connection, but an indirect

[1] The configuration of a molecule is defined by a set of coordinates, typically distances and angles, that describes the molecule's atom positions unambiguously [12].

[2] An empirical molecular mechanical energy function does not consider electrons explicitly. Bonds and bond angles are described by harmonic potentials [16]. The potential energy function supposes 0K hence no thermodynamics are considered.

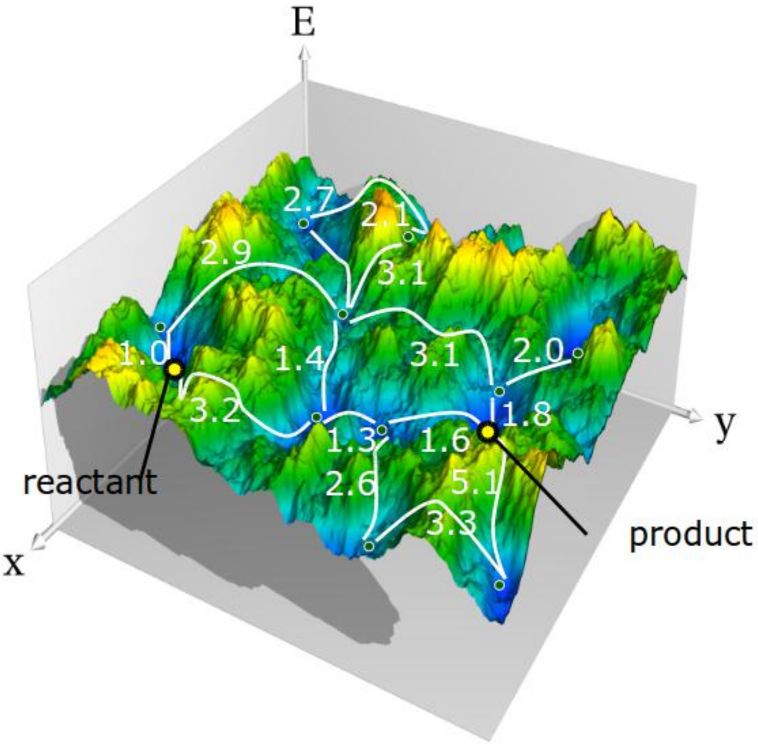


Figure 2.1: **Two Dimensional energy landscape:** Schematically each point in 2-dimensional configuration space is assigned to an energy value. The resulting energy surface visualizes the Transition network from reactant to product configuration. The network's vertices or nodes correspond to local minima on the landscape and are therefore meta-stable intermediate states. Edges represent subtransitions between the nodes. The edge-weights are equal to the potential energy barrier between the associated vertices. [1]

one, containing several edges (see fig. 2.1 and 2.2d). Every node on this path that is not reactant or product is called intermediate as it represents a possible temporary meta-stable configuration. *Cheap* paths connect local minima in such way that peaks are saddle points of first order which represent transition states.

The edge weight $w_{i,k}$ of two intermediate structures v_i and v_k is defined by the highest energy barrier on the path between them: $w_{i,k} = exp[\frac{E_{i,k}}{k_b T}]$, where $E_{i,k}$ describes the energy of the barrier and k_b Boltzmann's constant. An entire path, connecting reactant and product via several intermediates and sub-transitions has consequently the weight of all edge-weights summarized. Due to the exponential appearance of $E_{i,k}$ the sum is dominated by the highest energy. In the following we will indeed neglect the lower weights and take the highest energy-barrier as the cost of the path.

As it is impossible to know the whole landscape one limitation of Transition Networks is to choose an ensemble of dimensions and maximal displacements that covers enough volume of configuration space important for the transition. The next trade off is to fill this volume with sufficient intermediate nodes and let them propagate to local minima in order to be meta-stable (see section 2.2). Out of the resulting and remaining nodes, pairs of neighbors are assigned by applying threshold criteria like maximal distance in configuration space. Then the edge-weights between adjacent nodes are calculated. It is done by *Conjugate Peak Refinement* (CPR) which is an iterative method that starts with a linear interpolation between two nodes and then minimizes the highest peak in each step (see section 2.5). Especially the last part, calculating the energy barriers, is the most CPU demanding. The whole process is schematically shown in figure 2.2.

Because the number of edges to compute grows quadratically with the number of nodes[3] a smart selection of nodes is crucial. This can be achieved by firstly shrinking the configuration space to a set of reduced coordinates that are really necessary to describe the molecule's conformation and its upcoming displacements [13]. Out of those coordinates one option for node selection is including MD-simulations by taking snapshots in order to generate in-

[3] the maximum number of edges e in relation to the number of vertices v is in an undirected graph: $e_{max} = v \cdot (v-1)/2$

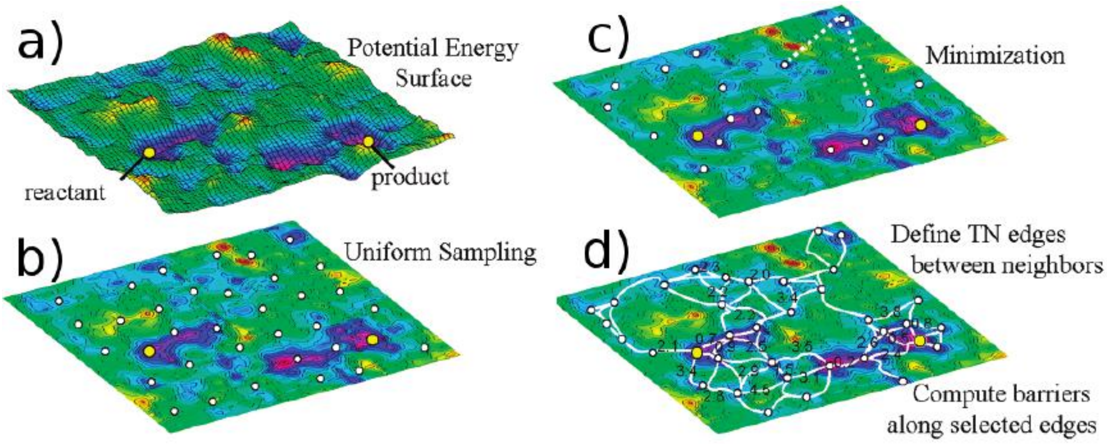


Figure 2.2: **Generation and Computation of a Transition Network**: **a)** Energy Surface of two-dimensional configuration space. Reactant and product state are located in local minima. **b)** Generation of intermediate sub-transition states by uniform sampling over energy surface. **c)** Minimization of intermediates. Several intermediates can collapse to the same valley. **d)** Computing energy barriers by assigning neighbours and calculating edge-weights between them.

termediates [2]. Without pre-knowledge out of MD simulations the most intuitive solution might be uniform sampling which evenly fills a specified volume by vertices (see figure 2.2b). Another is generating intermediates by uniform displacements from the reactant state. Parameters for the displacement are gridspace (distance between nodes) and gridsize (maximal displacement).

Based on the second method I implemented a Monte Carlo like algorithm that allows to not uniformly but randomly make displacements from the reactant state in order to generate intermediate states (see fig. 2.3). Those random displacements are still within the boundary conditions of the allowed degrees of freedom, gridspace and gridsize. This implementation is described more thoroughly in section 3.3.

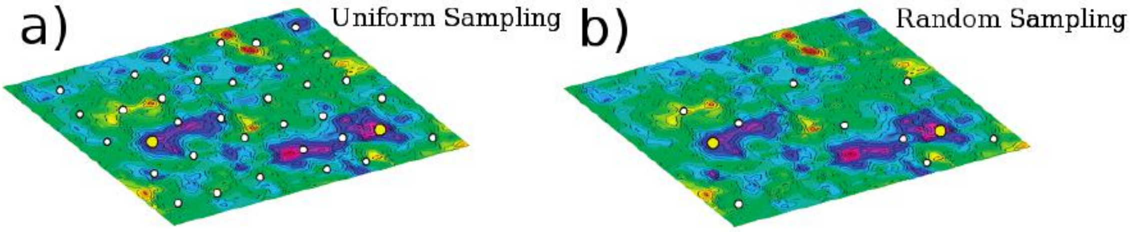


Figure 2.3: **Sampling of initial intermediate states**: **a)**: Uniform Sampling over a grid that is defined by gridsize(maximal displacement) and gridspace(distance or density between displacements). **b)** Random Sampling: From the grid in a) randomly 9 intermediates are generated.

2.2 Minimization

Steepest descent (SD) is probably the most intuitive way of finding local minima on the energy surface. Starting from an initial point it simply follows the direction with the biggest (negative) gradient for a certain stepsize. If x_k is the initial point and $x_{k+1} = x_k + d_k$ the point after one step then the displacement-direction d_k and x_{k+1} of the described method read like this:

$$d_k = -\frac{\mathbf{g}_k}{|\mathbf{g}_k|} \qquad\qquad x_{k+1} = x_k - \sigma_k \mathbf{g}_k.$$

In figure 2.4 we can see that it can take many steps until finally reaching the minimum. The method of *conjugate gradient* solves this problem by considering not only the last point but the last two points for finding the new direction to go. In addition to demanding $\mathbf{d}_k \cdot \mathbf{g}_{k+1} = 0$

and $\mathbf{d}_{k+1} \cdot \mathbf{g}_{k+2} = 0$, it also requires $\mathbf{d}_k \cdot \mathbf{g}_{k+2} = 0$. Like this it restores some minimization from before and provides to walk in a totally new direction. In figure 2.4 we can see that it reaches the minimum much faster, in this 2-dimensional example in only 2 steps.

Newton's method (NM) is mostly known for finding the root of a function in one dimension. Starting from one point it iteratively follows the tangent of that point and it's interception with the x-axis determines the new point for the next iteration. Transfered to the minimization problem, finding a function's extremum is equivalent to finding the root of $f(x)$. In multi dimensions that means to find the root of the gradient: The new point is here given by

$$x_{k+1} = x_k - \sigma \mathbf{H}^{-1} \cdot \mathbf{g}_k, \tag{2.1}$$

where $\mathbf{H}$ defines the Hessian matrix, $\mathbf{g}$ the gradient and σ the stepsize [16]. It is very powerful for quadratic or quasi-quadratic regions. Whereas the first two methods need to calculate only the gradient newton's method requires computing (and storing) the inverted Hessian matrix which becomes expensive for big systems.

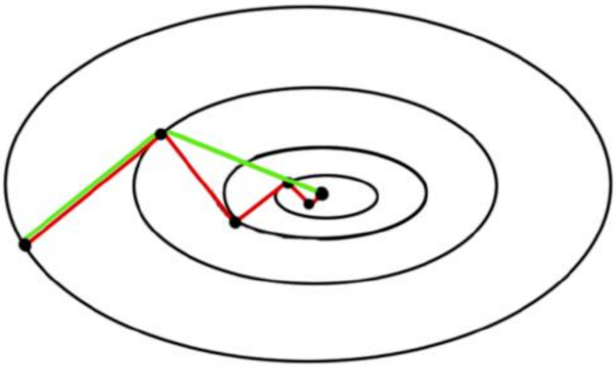


Figure 2.4: **Minimization on 2-dimensional energy landscape:** The red line describes the steps by applying steepest decent method and the green line for conjugate gradient. [16]

2.3 Conjugate Peak Refinement

Conjugate Peak Refinement (CPR) is an iterative method to find the optimal path between two states on the potential energy surface. It starts with a linear interpolation between the two states and picks the peak on this connection. This state is minimized along the conjugate gradient in each CPR-cycle. The resulting intermediate is then again connected to the other intermediates by linear interpolation and the cycle begins from the beginning for every connection path (see figure 2.5 for visualization). CPR stops when the gradients of all maxima fall below a threshold value or by maximum number of iterations. With this method one or more transition states can be found. For obtaining the optimal path the resulting states can be minimized again by for instance steepest descent method. In praxis it proofed to be most efficient if minimization cycles are nested into CPR cycles.

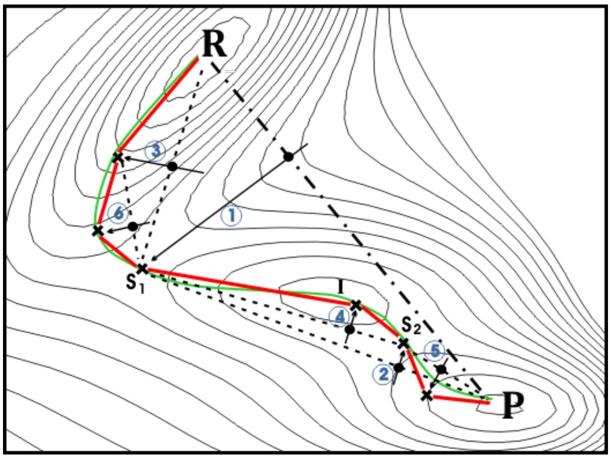


Figure 2.5: **Conjugate Peak Refinement (CPR):** reactant(R) and product(P) are stable states, hence local minima on the energy landscape. The CPR method finds a path by iteratively connecting the intermediates by linear interpolation and minimizing the maxima along the conjugate gradient. The resulting path is the **red** one. In order to get the optimal path (**green**) minimization of pathpoints needs to be done [17].

3 Framework

3.1 5w-model

When creating an algorithm it is useful to first check it in a small test environment that is easy and fast to analyze and where preferably the expected output is known in advance. In our case the test model is a proton transfer in a dummy molecule consisting of 5 flexible water molecules + 1 excess proton for the transfer and fixed aspartate-like residues at the end. The residues on both sides contain each 3 methyl groups connected by a carbon atom (see fig. 3.1).

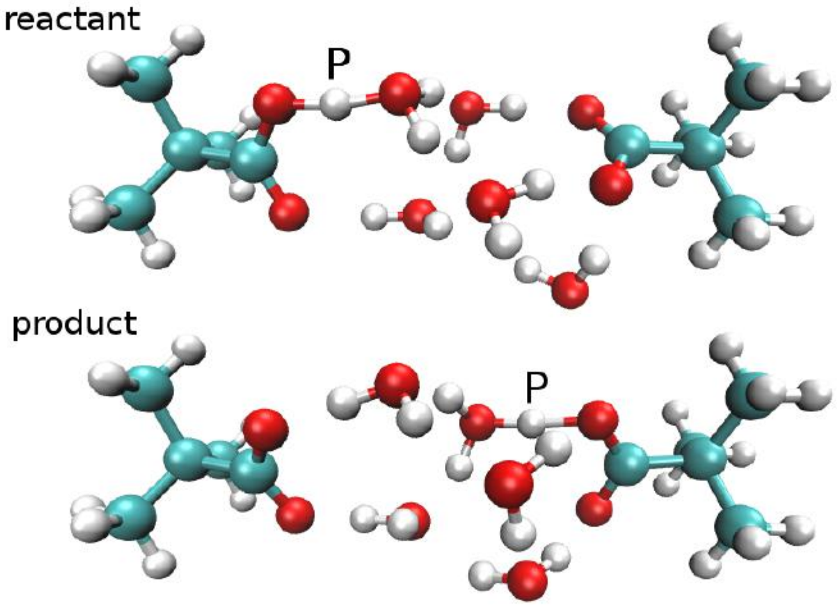


Figure 3.1: **5w-model:** 5 H2O + 1 excess proton for the transfer and fixed residues at the end. The aspartate-like residues contain each a carboxyl group which represent the side chains. Hydrogen atoms are shown gray, oxygen atoms in red and carbon atoms in blue. The excess proton is labeled with a P. **Top:** reactant state is shown with the transition proton on the left side. **Bottom:** product state, the proton is transfered to the right hand side.

The residues are interfaced with the water molecules by 2 reactive carboxyl groups which simulate the side chain of aspartate amino acids. The distance between the residues is 7.51 Å. Around the axis that points through the carboxy groups a cylindrical potential was applied in order to restrain the water molecules in a cylinder around the axis. The harmonic potential starts within a cut-off distance of 3 Å and has a force-constant of $500 kcal/mol/\mathring{A}^2$. Like that the dummy molecule simulates a water-filled protein channel.

This model is small enough to compute the entire transition network with uniform sampling over the whole volume of configuration space. The best paths from a smaller networks with fewer intermediates can then be compared among each other and with the best path from the entire network. The trade-off between CPU-time and accuracy can thereupon be evaluated.

3.2 Used Software

For minimization and CPR the CHARMM (CHemistry at HARvard Macromolecular Mechanics [14]) program was used.

Visual representation of the molecule was done by VMD(Visual Molecular Dynamics [19]) with a representation mode of CPK and dynamic bonds. All Plots are generated by python using the matplotlib package [18].

Initial node generation, node and edge assignment, determination of neighbor pairs, path-reading, further compilation and analysis of the network were done by using local java code and libraries based on the work of Noe et al. [1] and Petra Imhof [15].

3.3 Monte-Carlo State Selection

Like shortly mentioned in section 2.1 intermediate states can be generated by uniform displacement from the reactant state. Theoretically, for every degree of freedom (DOF) an extra gridspace and gridsize could be defined. In practice it makes more sense to define these values for a set of dimensions. DOFs were assigned to the following sets: water displacement, protonation, sidechain torsion. For instance one can allow all oxygen atoms of water molecules to move 1 Å in 6 directions while fixing all atoms from residues.

The new algorithm (see fig. 3.4) allows to randomly select states out of all possible ones from the setting above. For selecting them it needs as input how many water movements etc. it picks. For understanding this selection process it helps to imagine the generated states with a tree diagram (see figure 3.3): every branch symbolizes one state. On every junction

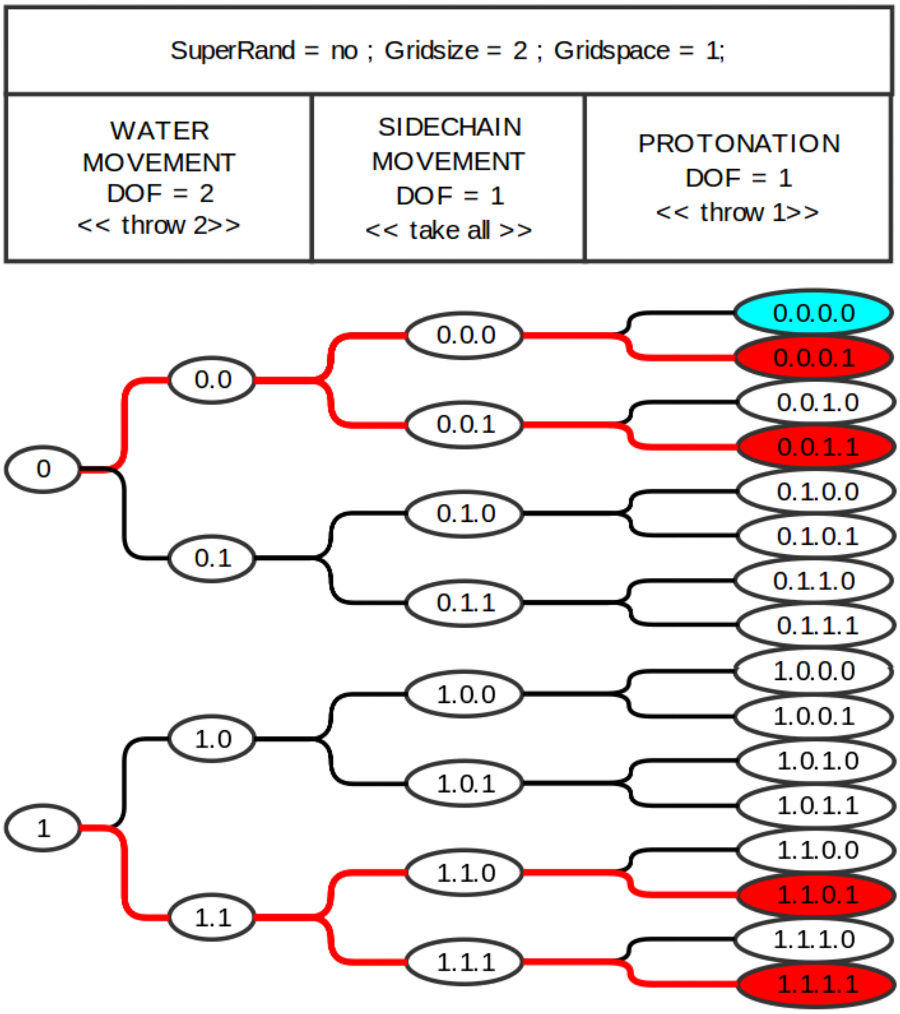
$\frac{gridsize}{gridstep}$ sub-branches lead off for the respective DOF. This hierarchy is divided into water displacement, protonation, sidechain torsion. The algorithm creates the requested number of branches for the water movement set and concatenates to every branch the wanted number of sidechain-rotation branches and so on. It can be configured whether the same sub-branches are concatenated to every super-branch or whether they are *thrown* freshly for each of those. If the latter setting is valid than this option is called **superRand** in the float char below.

It's not even necessary to generate the whole network and discard the states we don't want. Instead of that a branch can directly be randomly selected and checked if it already exists. This is done iteratively until the amount of required branches is reached. Doing it this way it is computationally faster if the number of thrown branches is relatively small compared to the number of possibly generated branches.

A flow chart that visualizes the program structure and how it processes input configurations in order to generate the states by the selection process above is shown in figure 3.4 and 3.5.

3.4 Settings

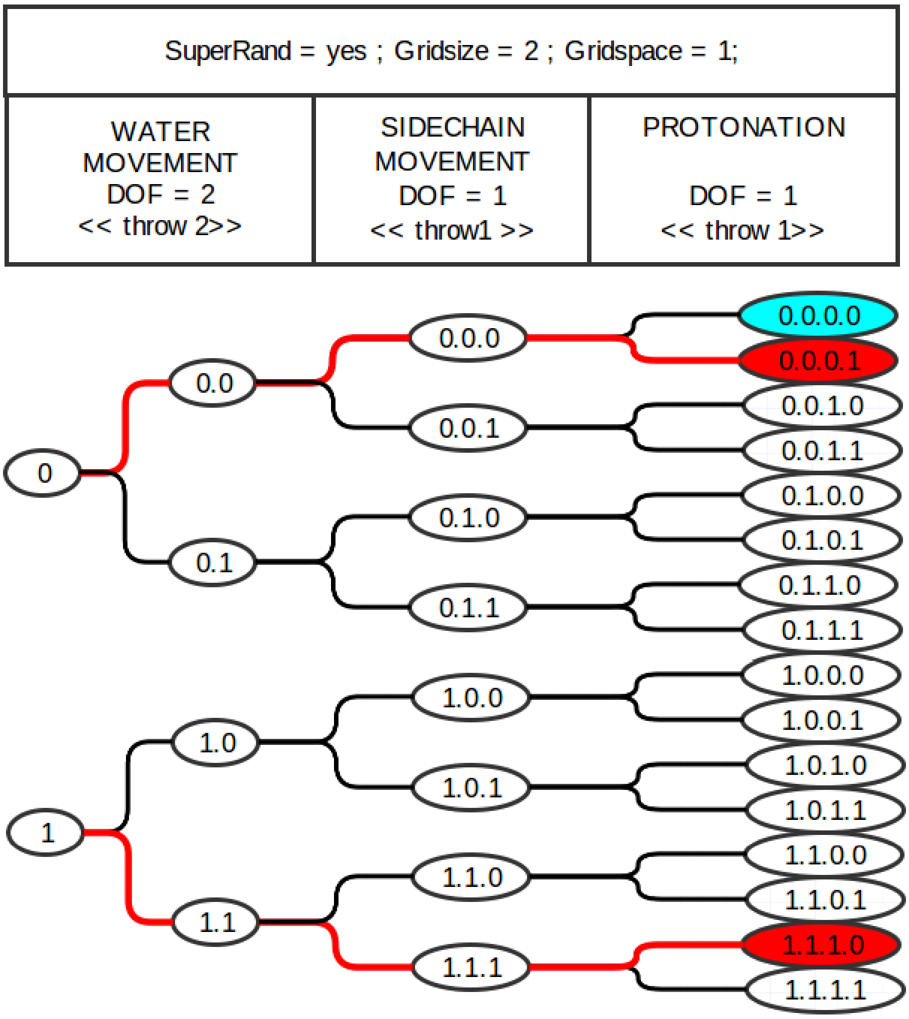
Minimization was done by Newton Raphson method (ABNR)[3] with 1000 steps or until a total gradient of 0.001 kcal/mol/Å reached. CPR starts with a linear interpolation and does 10 loops of minimizing the interpolation's peak with 10 cycles each. The CPR's minimization convergence criteria was 0.001 kcal/mol/Å using the semi-empirical OM2 method [9].
In all the networks being considered in the results the following criteria were kept constant: For the neighbor assignment a maximum difference of 2 steps per DOF, including water displacement, protonation state and sidechain rotation by given rotation-stepsize, is allowed and a maximum RMSD difference of 2.5. Gridsize and Gridspace are equal to 2 Å. Sidechain rotation stepsize along the molecule's axis is 90 degree for initial state generation and 60 degree when reading in the outcome after minimization. The *superRand* method (see 3.3) was used for all examples except for figure 4.4, which explicitly covers the superRand function. In all chapters below networks with two different initial settings for the state generation are compared. In one setting only water-movement was allowed and in the second all degrees of freedom were permitted, but only 2 branches were picked for sidechain movement and protonation each.



SuperRand = no ; Gridsize = 2 ; Gridspace = 1;		
WATER MOVEMENT DOF = 2 << throw 2>>	SIDECHAIN MOVEMENT DOF = 1 << take all >>	PROTONATION DOF = 1 << throw 1>>

Figure 3.2: **Tree schema of *semi*-random state selection:** The tree shows all possibly generated states by uniform displacement from reactant (marked in **azure**) due to gridsize and gridstep for each DOF. Out of all possible branches 2 are randomly selected for water movement. Concatenated to those are all possible sub-branches from sidechain-rotation and 1 sub-branch from protonation. The configuration superRand = no means that the same sub-branches are concatenated. The resulting 4 initial states are marked in **red**.
0 stands for no displacement and 1 for a displacement of 1 Gridspace in the respective DOF.

example: state 0.0.1.1 represents an initial state that differs from reactant state by no displacement of waters but by a sidechain rotation and by a change in protonation state.



SuperRand = yes ; Gridsize = 2 ; Gridspace = 1;		
WATER MOVEMENT DOF = 2 << throw 2>>	SIDECHAIN MOVEMENT DOF = 1 << throw1 >>	PROTONATION DOF = 1 << throw 1>>

Figure 3.3: **Tree schema of random state selection:** The tree shows all possibly generated states by uniform displacement from reactant (marked in **azure**) due to gridsize and gridstep for each DOF. 0 stands for no displacement and 1 for a displacement of 1 Gridspace in the respective DOF. Out of all possible branches 2 are randomly selected for water movement. Concatenated to those are all possible sub-branches from sidechain-rotation and 1 sub-branch from protonation. The configuration superRand = yes means that sub-branches are randomly thrown again for every super-branch. The resulting 2 initial states are marked in **red**.

remark: Note that also the reaction state could be randomly selected.

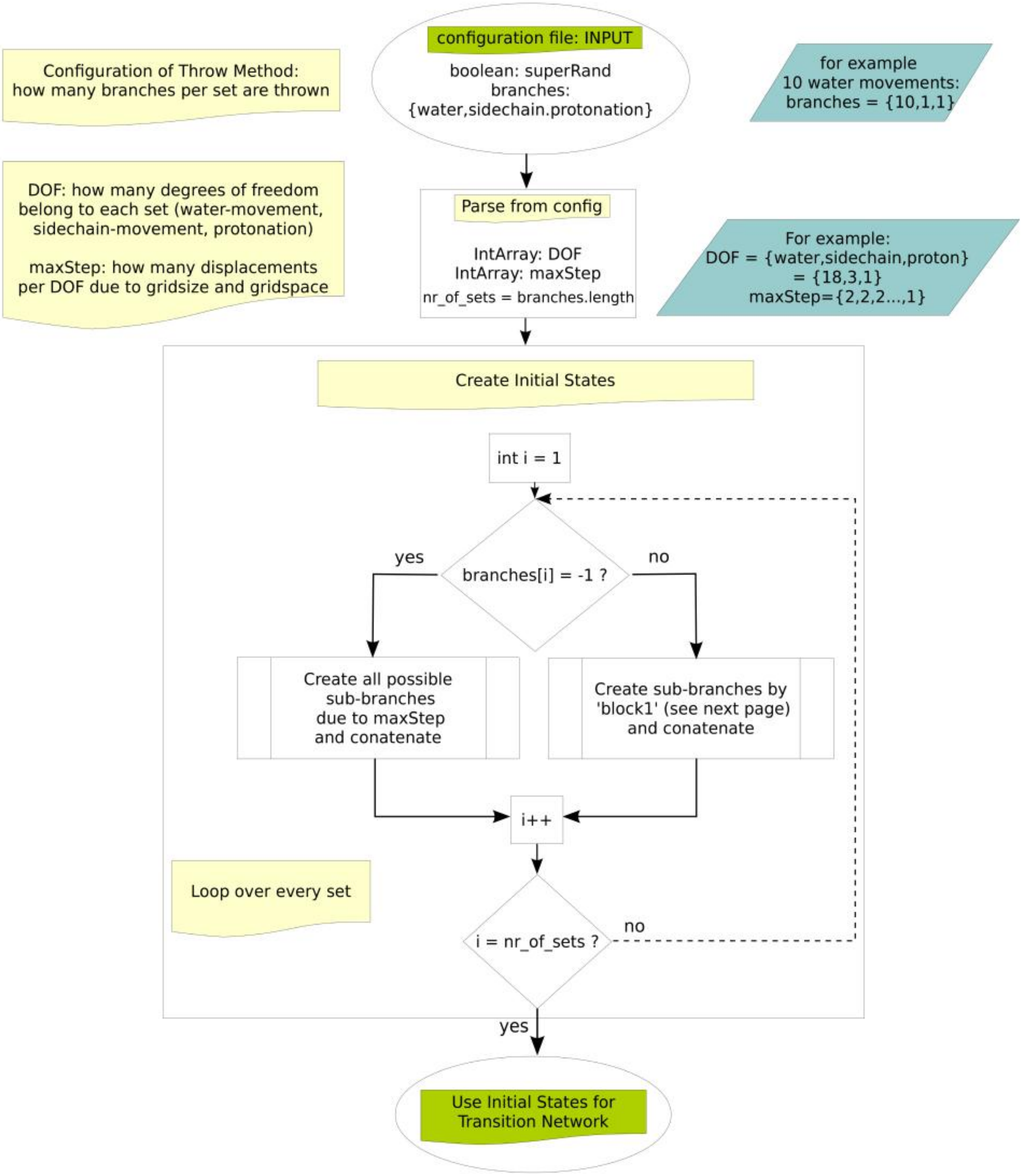


Figure 3.4: **Essential Flowchart of Initial State Generation**

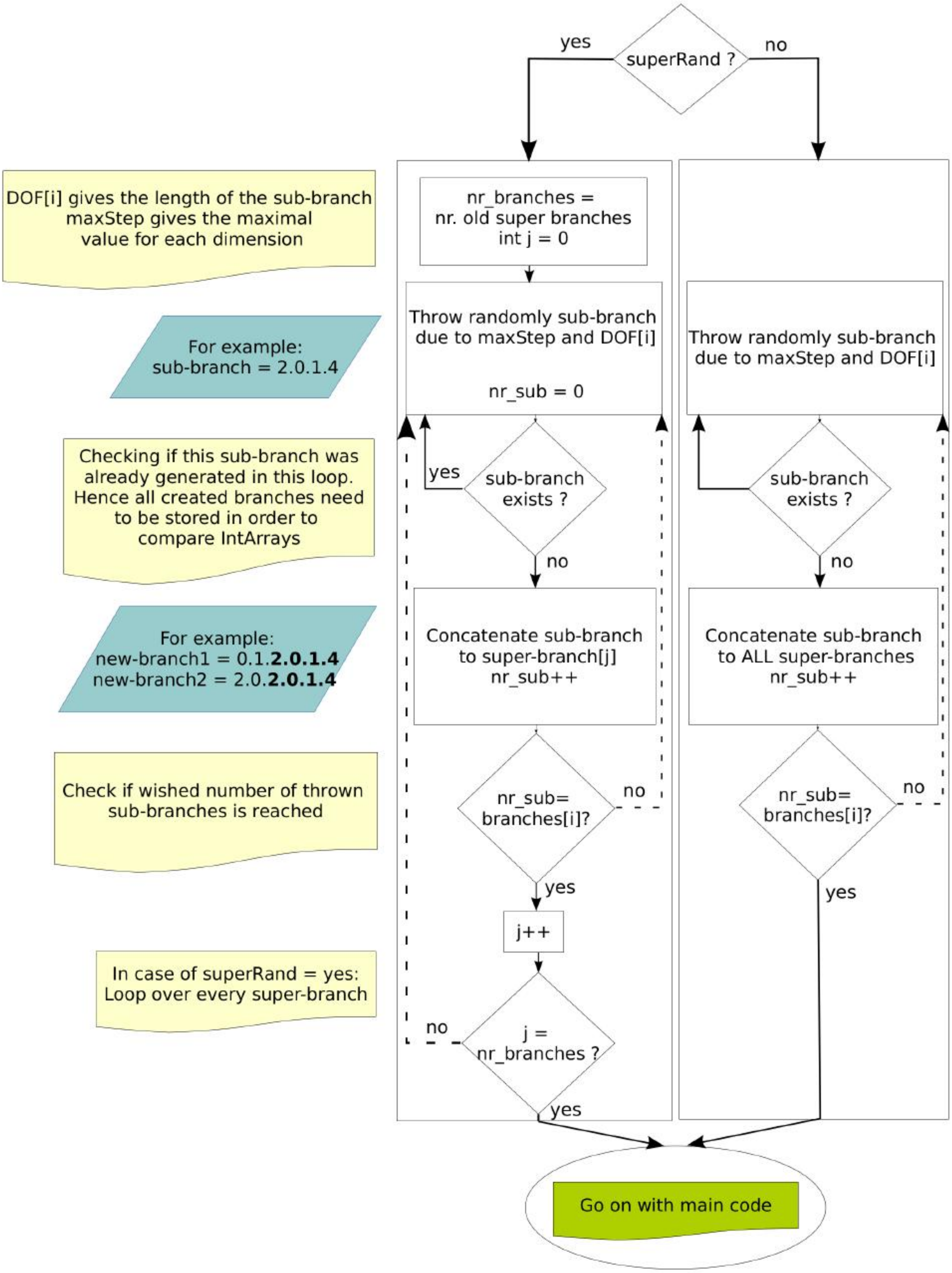

superRand ?
yes
no
DOF[i] gives the length of the sub-branch maxStep gives the maximal value for each dimension
nr_branches = nr. old super branches int j = 0
Throw randomly sub-branch due to maxStep and DOF[i]
nr_sub = 0
Throw randomly sub-branch due to maxStep and DOF[i]
For example: sub-branch = 2.0.1.4
Checking if this sub-branch was already generated in this loop. Hence all created branches need to be stored in order to compare IntArrays
yes
sub-branch exists ?
sub-branch exists ?
no
no
For example: new-branch1 = 0.1.**2.0.1.4** new-branch2 = 2.0.**2.0.1.4**
Concatenate sub-branch to super-branch[j] nr_sub++
Concatenate sub-branch to ALL super-branches nr_sub++
Check if wished number of thrown sub-branches is reached
nr_sub= branches[i]?
no
nr_sub= branches[i]?
no
yes
yes
j++
In case of superRand = yes: Loop over every super-branch
no
j = nr_branches ?
yes
Go on with main code

Figure 3.5: **Flowchart Block 1**

4 Results and Discussion

The best paths of different transition networks are represented and are compared among each other. The transition networks were generated by given initial settings for the intermediate node creation (see 3.4), using the Monte-Carlo method of chapter 3.3.

For making life easier I shortened the name of the settings that were used to generate the networks by the nomenclature:
«allowed DOF(w=water,s=sidechain,p=protonation).#initial states».
Consequently wsp.100 means that all degrees of freedom were allowed and in total 100 initial states are generated [1]. For the sake of readability the magnitude **kcal** is used as a shortcut for **kcal/mol** in this section.

All calculated networks that were taken into account for the analysis are listed in the attachment in table 6.1. The RMSD value of all created states of one network with respect to the reactant state are for all networks between 0.65 and 0.68. This holds before and after minimization.
The highest energy-barrier of the direct path from reactant to product is already only 2 kcal. This means that the best path can't really be improved much by creating larger networks with more nodes. Henceforth rather than taking only the very best path into account the number of paths up to an energy-barrier of 3 kcal and up to 10 kcal are compared. The three best paths of the largest network calculated are shown in figure 4.1. It is interesting to note that some nodes and edges have a negative relative energy. This means that neither the reactant nor the product state, which has a small positive relative energy, are located in a global minimum.

[1] Knowing that $DOF_{sidechains} = 2$ and $DOF_{Protonation} = 2$, we can deduce that $DOF_{water} = 25$

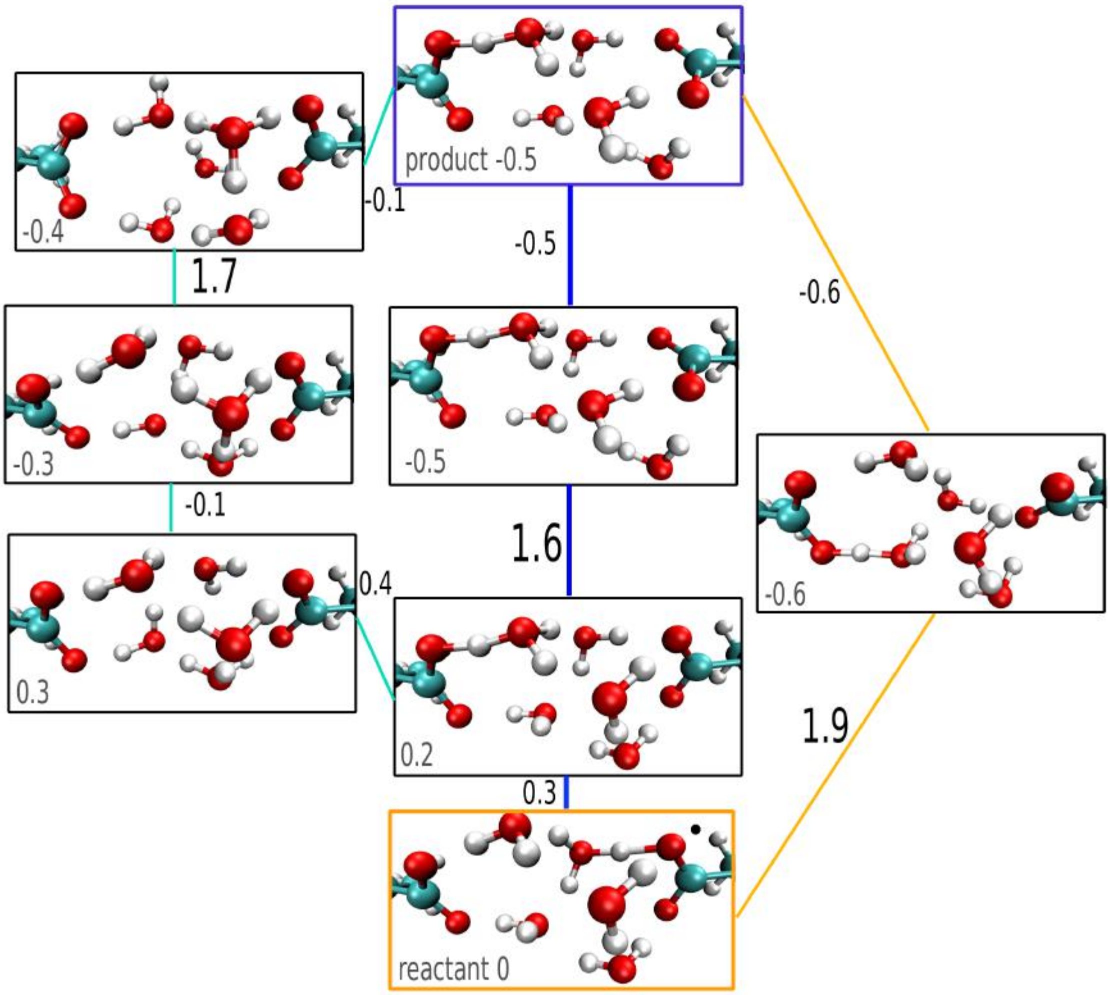


Figure 4.1: **3 best paths of wsp.2136 network** Allowed DOF: Watermovement + 2 Sidechain Movement + 2 Protonation States. The figure shows the 3 best paths with its intermediate nodes and edges. The energy barrier of an edge is written in black. The potential energy of a node relative to the reactant state is written in grey. The best path is labeled in blue, 2nd best in azure and 3rd in orange.

4.1 Convergence

In order to test if the number of paths below a certain energy-barrier converges, the total number of paths below 3kcal and below 10 kcal are plotted against the total number of initial states (see figure 4.2). As expected the number of paths grows with the number of initial states and seems to reach a plateau. In both figures it is remarkable that the number of paths grows in a logistic way, hence reaches a saturation of paths with this low energy and can barely be improved by expanding the network to more initial nodes. Both functions were fitted with a logistic function (see figures) which represents indisputably a prior belief. Indeed this prior belief is verified by firstly evaluating the goodness of the fit by RMSD-value and R^2[2] value, which is between 0.9 and 1 in all examples. Secondly the logistic function which is characterized by converging against a saturation level is based on empiric knowledge based on former work of M. Reidelbach et al. [2]. The expected saturation levels regarding to the fitted functions are: $w^{max}_{<3kcal} = 12$, $w^{max}_{<10kcal} = 13.2$, $wsp^{max}_{<3kcal} = 17.1$, $wsp^{max}_{<10kcal} = 50.9$. 2 larger networks with over 2000 initial states were created to test if the expected saturation level holds true: $w^{2027}_{<3kcal} = 8$, $w^{2027}_{<10kcal} = 12$, $wsp^{2136}_{<3kcal} = 49$, $w^{2136}_{<10kcal} = 91$. For the water networks the prediction holds true whereas for the wsp networks more data points are required to proof a convergence.

In addition to the analysis of the number of best paths vs. the number of initial states we also tested the dependence of the number of best path on the number of nodes and on the number of edges. Instead of the anticipated reduction of the variance in the number of best paths an even larger variance was observed that way (not further represented). However, one can conclude that the quality of the nodes determines the number of best paths rather than the number of nodes.

Note also the difference in scaling between the two figures below. Indeed the number of paths of the wsp-model is 2-3 times higher in comparison to the water-model. This difference becomes even clearer in the accumulated visualization form.

[2] R^2 is a relative value between 0 and 1 for assessing the fit of a regression model. 1 represents a perfect fit and 0 no correlation. Note that R-square does not include a penalty function for growing number of predictors, hence overfitting can in general be a pitfall but is in our example neglected due to the a posteriori logistic approach.

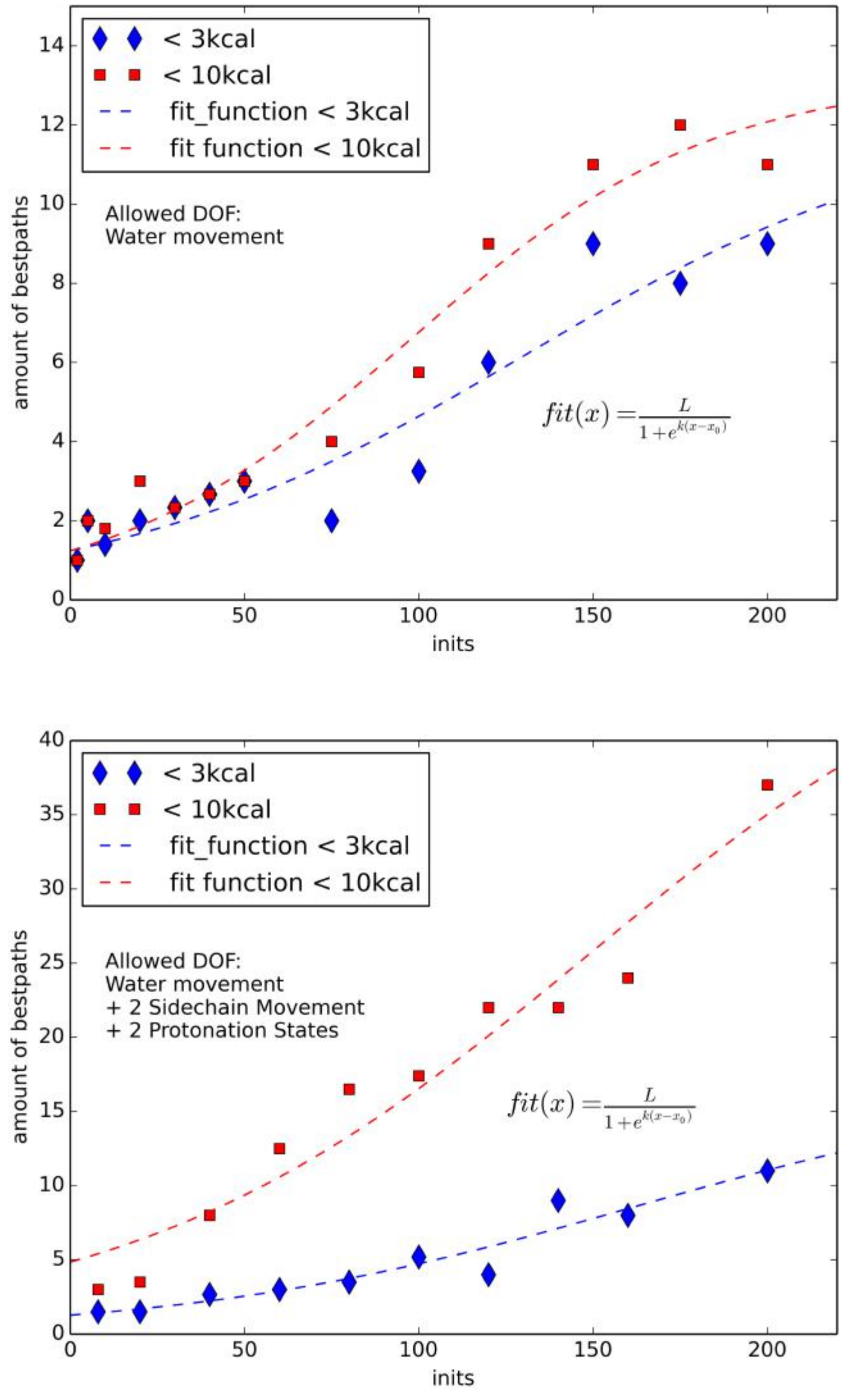


Figure 4.2: **Paths under 3kcal and under 10kcal for water and wsp networks:** The graph shows the total number of paths that have an energy lower than 3kcal or 10kcal in relation to the number of initial states. The dashed lines show fitted functions according to the measured values. The goodness of the fits are for the water-model: $<$3 kcal: $R^2 = 0,921$ RMSD$= 0,879$ $<$10kcal: $R^2 = 0,946$ RMSD$= 2,32$
and for the wsp-model: $<$3 kcal: $R^2 = 0,913$ $RMSD = 0,837$ $<$10kcal: $R^2 = 0,966$ $RMSD = 0,713$

4.2 Accumulated Paths

Let's suppose we wanted to examine deeper how the paths are distributed with respect to a certain initial sampling. As the sampled states are gained by a random selection several networks have to be computed with the same setting in order to take the mean value out of them. Hence in the following each visualized network that is treated in this section refers to the average values out of at least 5 computed networks (see table 6.1).
With the intention to compare easier two networks with each other in the figures below the *accumulated* number of paths up to an energy threshold is plotted against this energy value. The error bars were computed by standard deviation. In the first plot 4.3a only water-movement was allowed. The number of paths for 10 initial states and 100 are not significantly different within the error interval. In both networks (w10, w100) most paths are found with an energy up to 3kcal. This is different in the second plot 4.3b, were all DOFs are allowed. Here the distribution of paths is much wider spread, which makes sense as the forced testing of paths including sidechain-movement and especially protonation-hopping is expected to deliver a wide range of good paths. This is the reason why also the total number of paths in the wsp-model is more than 3 times higher than in the w-model.

4.3 SuperRand

The influence of concatenation of different sub-branches due to the selected degrees of freedom were tested in the sections before. Here the initial setting of the compared networks are equal except for the function *superRand*(see section 3.3) which is turned off in one example. The biggest difference that results of disabling the superRand function is that the error bars are getting bigger than the values itself in the plot. Consequently there is no reasonable comparison possible between the paths from super-randomly selected initials. In fact this variance is logic if we consider what this function actually does. If disabled it concatenates always the same sub-branches to the superior branches. In our case the sub-branches mean different configurations distinguished by sidechain and proton position. Now if by chance these *thrown* configurations are favorable then the result is a lot of good paths in the final network. Because the same applies for bad configurations a big variance among the networks' path energies but a poor variance within one network is caused.

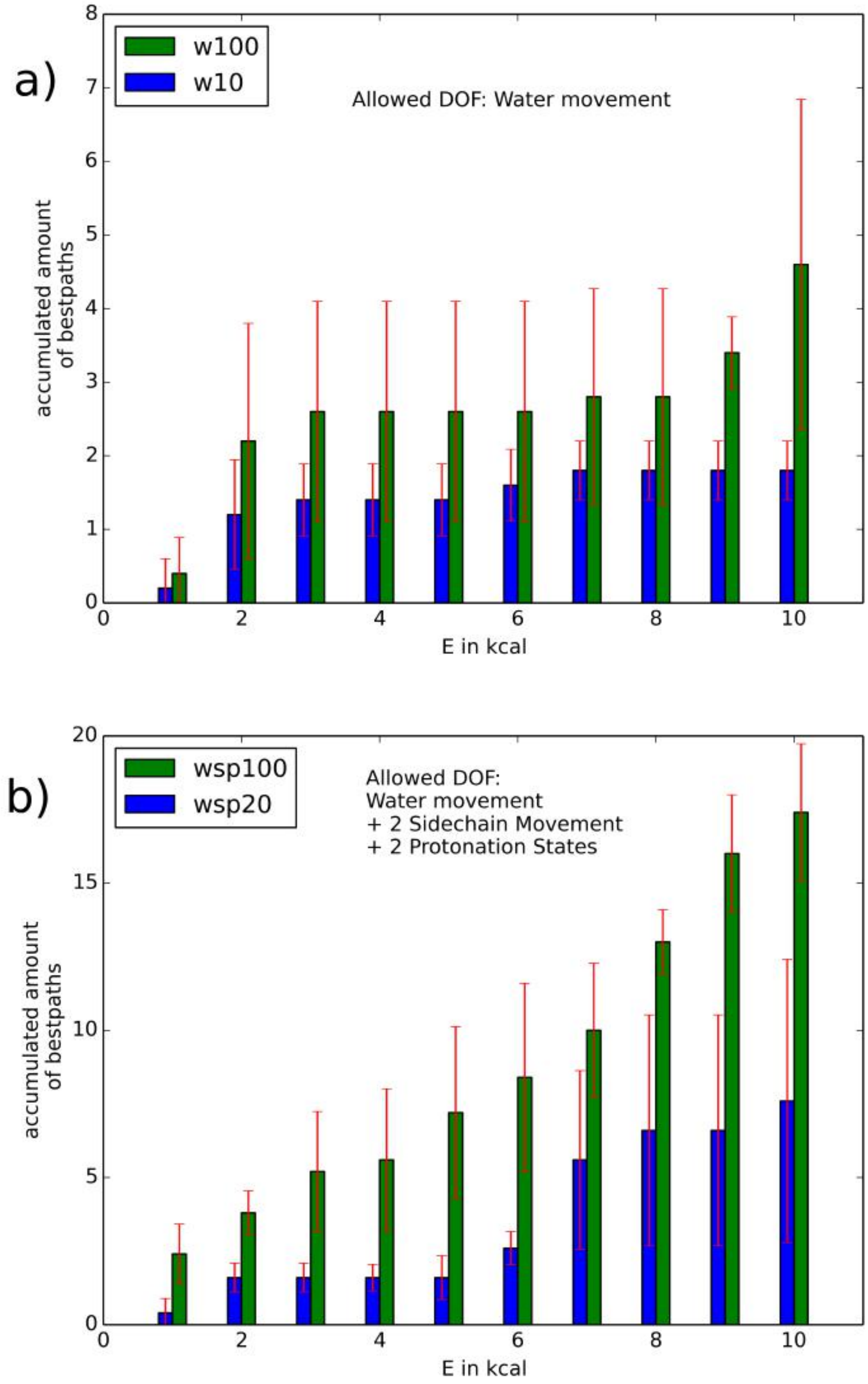


Figure 4.3: **Accumulated number of paths for 10 and 100 initial states in water and wsp networks:** The graph shows the accumulated number, i.e how many paths exist up to a certain energy level, with respect to the potential energy. Two bars next to each other belong to the same energy value and are only visualized like that for a better readability of the error-bar. The error was computed by standard deviation. Each network represents the mean out of 5 samples with the same setting.

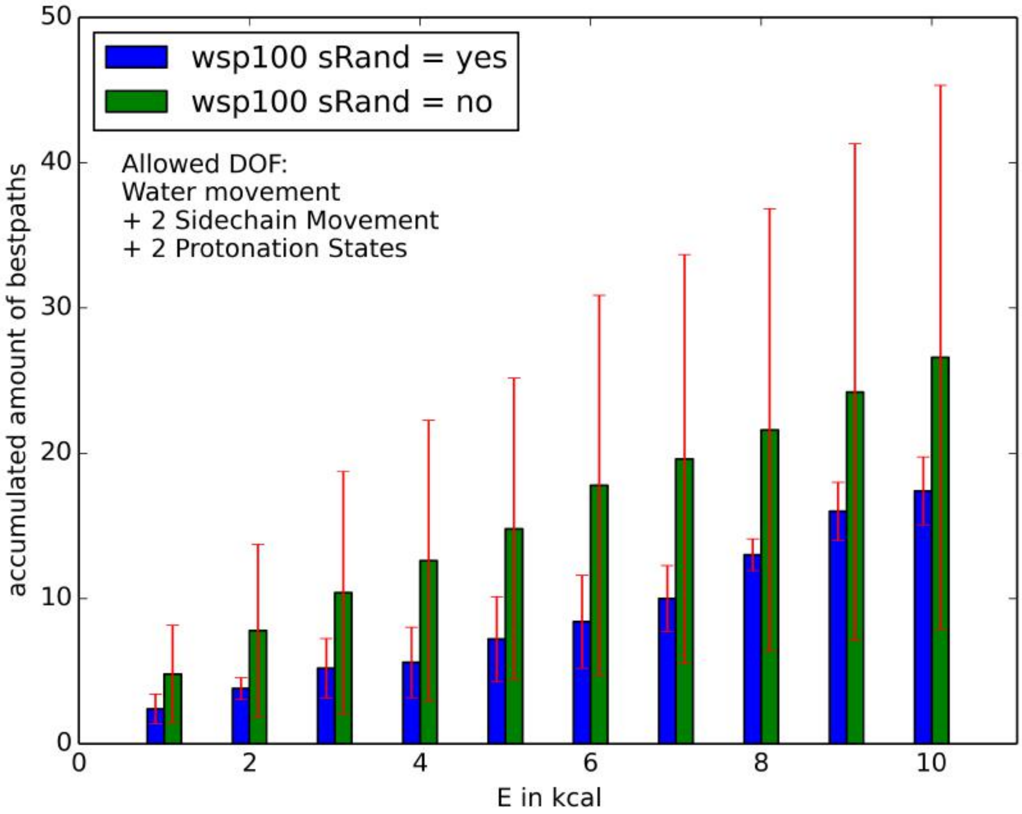


Figure 4.4: **Accumulated number of bestpaths for wsp100 network with *superRand* function on and off:** Allowed DOF: Watermovement + 2 Sidechain Movement + 2 Protonation States. The graph shows the accumulated number, so how many paths exist until a certain energy level, with respect to the potential energy. This was done for 2 networks which have the same allowed degrees of freedoms but where once created with and once without the *superRand* function enabled (see 3.3). Two bars next to each other belong to the same energy value and are only visualized like that for a better readability of the error-bar. The error was computed by standard deviation. Each network represents the mean out of 5 samples with the same setting.

5 Conclusion

Paths with low energies of 1kcal or lower were found for the proton transfer already with very small networks, created by e.g 10 initial states. Calculating bigger networks is justified if one is not only interested in the best but also in the second, third best paths and so on. The networks, generated by different initial settings determined by the allowed degrees of freedoms and number of initial states, were compared in terms of the total number of best-Paths below 3kcal and below 10 kcal and by visualizing the distribution of selected networks that were sampled 5 times. Within the range between 5 and 200 initial states a convergence of paths below the given energy level could be witnessed. Comparing networks with different degrees of freedom showed that a smart selection of DOFs is just as important as the total number of initial states. Examining the different outcome of enabling or disabling the *superRand* function emphasized the influence of different sub-branches that are concatenated.

Especially in this part, the concatenation of sub-branches, I see the biggest potential for saving computational time in further studies. An iterative method could begin with calculating small, coarse, networks and based upon their outcome create larger networks by increasing the sampling density in dimensions that had a big influence or by explicitly keeping those nodes that are often involved in good paths. This option of *growing networks* would be even more beneficial when the described random state selection method is applied to bigger molecules, where a comprehensive, uniform sampling is computationally expensive.

6 Apendix

SETTING	Energy bestPath	#Paths <3kcal	#Paths <10kcal	Trials
reac_prod	2	1	1	1
w2	2	1	1	1
w5	5	2	2	1
w10	10	1.4	1.8	5
w20	20	2	3	2
w30	30	2.333	2.333	3
w40	40	2.667	2.667	3
w50	50	3	3	3
w75	75	2	4	1
w100	100	3.25	5.75	5
w120	120	6	9	1
w150	150	9	11	2
w175	175	8	12	1
w200	200	9	11	2
WSuper[1]	2027	x	x	1
wsp8	8	1.5	3	2
wsp20	20	1.5	3.5	5
wsp40	40	2.67	8	3
wsp60	60	3	12.5	2
wsp80	80	3.5	16.5	2
wsp100	100	5.2	17.4	5
wsp100_SRAND	100	10.4	26.6	5
wsp120	120	4	22	1
wsp140	140	9	22	1
wsp160	160	8	24	1
wsp200	200	11	37	1
wspSuper[2]	2136	x	x	1

Table 6.1: **Overview of generated networks:** The Setting refers to the used configuration for sampling the initial states beofre minimization and is named by the scheme. *allowedDOFs . nrInitialstates.* reac_prod describes the direct edge between reactant and product state and SRAND marks the only network were the superRand funciton was turned off.

[1] created by minimized initials of all water networks

[2] created by minimized initials of all wsp networks

Glossary

CPR Conjugated Peak Refinement. 5, 8, 10, 11

DOF Degree of Freedom. 10–13, 16, 17, 20, 22–24

MD Molecular Dynamics. 2, 5, 6

QM Quantum Meachanics. 2

RMSD route-mean-square deviation. 11, 16, 18, 19

SD Steepest Descent. 6

throw The term *throwing* states in this work means randomly selecting states and originates from *throwing the dice*. 11, 13, 20

Bibliography

[1] F. Noe, D. Krachtus, J. C. Smith and S. Fischer: *Transition Networks for the Comprehensive Characterization of Complex Conformational Change in Proteins*, 2006, Journal of Chemical Theory and Computation p.840-857

[2] M. Reidelbach, F. Betz, R. M. Mäusle, P. Imhof: *Proton transfer pathways in an aspartate-water cluster sampled by a network of discrete states*, 2016, Chemical Physics Letters vol. 659 p.169-175

[3] P. Deuflhard, A. Hohmann: *Numerische Mathematik I: Eine algorithmisch orientierte Einführung*, 3rd version, Berlin, New York, 2002, ISBN 3-11-017182-1, p.97ff

[4] P. Meredith, C.J. Bettinger, M. Irimia-Vladu, A.B. Mostert, P.E. Schwenn :*Electronic and optoelectronic materials and devices inspired by nature*, 2013, Rep. Prog. Phys. Soc. 76 034501.

[5] Zhou J. *Amide proton transfer imaging of the human brain*, 2011, Methods Mol Biol. vol. 711 p.227-37

[6] D.J. Wales *Discrete path sampling*, 2002, Molec. Phys. vol.100(20) p.3285-3305

[7] C. L. Brooks III, J. N. Onuchic, D. J. Wales: *Taking a Walk on a Landscape*, 2001, Science vol. 293 p.612f

[8] M. R.A. Blomberg, P. E.M. Siegbahn: *Different types of biological proton transfer reactions studied by quantum chemical methods*, 2006, Biochimica et Biophysica Acta (BBA) Bioenergetics vol. 1757 p.969-980

[9] W. Thiel, W. Weber *Orthogonalizaion corrections for semiempirical methods*, 2000, Theor. Chem. Acc. vol.103 p.495

[10] M. Bagherpoor Helabad Ghane, M. Reidelbach, A. L. Woelke, E.W. Knapp and P. Imhof : *Protonation state dependent communication in Cytochrome c Oxidase*, 2016, Biophysical Journal

[11] S. Cukierman *Et tu Grotthuss!*, 2006 Biochimica et Biophysica vol. 1757 (8) p. 876-878

[12] D. Chen : *Bachelorarbeit: Auswahl und Evaluation von Reaktionskoordinaten für den Protonentransfer*, Freie Universität Berlin Fachbereich Physik, 2016

[13] F. Betz: *Identifizierung von Freiheitsgraden in Übergangsnetzwerken*, Freie Universität Berlin Fachbereich Physik, 2016

[14] B. R. Brooks et al. : *CHARMM: The Biomolecular Simulation Program*, 2009, Journal of Computational Chemistry vol. 30, p.1545-614

[15] P. Imhof: *A Networks Approach to Modeling Enzymatic Reactions*, 2016, Institute of Theoretical Physics, Free University Berlin, Berlin, Germany

[16] P. Imhof and Nuria Plattner : *Methods of Molecular Simulations - Lecture Notes*, Freie Universität Berlin Fachbereich Physik, 2014

[17] Stefan Fischer und Martin Karplus : *Conjugate peak refinement: an algorithm for finding reaction paths and accurate transition states in systems with many degrees of freedom*, 1992, Chemical physics letters vol. 194.3 p. 251-261

[18] John Hunter: *matplotlib (Version 1.5.1)*[Software]. Available from http://matplotlib.org/

[19] William Humphrey, Andrew Dalke, and Klaus Schulten: *VMD - Visual Molecular Dynamics*, 1996, Journal of Molecular Graphics vol. 14, p. 33-38

[20] S. Pezeshki, H. Lin:*Adaptive-partitioning QM/MM for molecular dynamics simulations: Proton hopping in bulk water*, 2015, J. Chem. Theor. Comput. vol. 11, p.6 2398âĂŞ2411.