SELF AVOIDING WALKS
AND
LATTICE POLYMERS

DIPLOMARBEIT

zur Erlangung des
akademischen Grades

Magister rerum naturalium
an der Formal- und Naturwissenschaftlichen
Fakultät der Universität Wien

vorgelegt von

Alexander RENNER

Wien, im Dezember 1994

This work was carried out in the time from February 1994 to December 1994 at the Institute of Theoretical Chemistry of the University of Vienna.
First of all I want to mention Peter Schuster for giving me the chance to join his group. At this point I would like to thank all my friends and colleagues for their support and help to achieve this goal.
Peter Stadler supported me with all his tremendous knowledge and wisdom. Erich Bornberg-Bauer was of great help during the work and supported me in computational as well in theoretical work. A lot of the data here achieved was a collaboration with him (when you find the symbol © he had a finger in the pie). Ivo Hofacker the great hacker :⊃ assisted my first steps on workstations and debugged with me endless times.
Last but not least all the other members of the group: Ronke Babajide, Walter Fontana, Josef Frömcke, Thomas Griesmacher, Robert Happel, Robert Hecht, Bärbel Krakhofer, Herbert Kratky, Kai Neumann created a superb working (and not only working) atmosphere.
I also wish to thank my parents for their support and appreciation during my whole study.

Abstract

During most of the time to establish this diploma thesis a toolkit for Self Avoiding Walks (SAWs) [34] and Lattice Polymers (LPs) was created. The package works independently from the chosen lattice and is extendible. Various methods to characterize and compare structures as well as to detect secondary structure elements, in the sense of Dill's lattice protein [6] analogy to natural proteins are introduced. Different types of data structures for handling of LPs based on relative moves, distance matrices and contact matrices are provided.
The toolkit was used to calculate several characteristic properties of SAWs, such as the average length ℓ of growing Self-Avoiding Walks (gSAWs on regular lattices often terminate by self-trapping) and the shape of the distribution. The minimum length of gSAWs has been computed using exact enumeration as well as the distribution of relative directions. Also a first glimpse on landscapes [46] was taken.
One of the most important applications of the SAW is a model for linear polymer molecules in chemical physics. And recognizing that proteins are polymers one can see the aim and importance of this work to define models for proteins.

Zusammenfassung

In dieser Diplomarbeit wird ein Software Paket für "Self Avoiding Walks" (SAWs) [34] und Gitter Polymere präsentiert. Das Paket is gitterunabhängig und erweiterbar. Verschiedene Methoden, um Strukturen (nach DILL [6]) und Sekundär-Strukturen zu vergleichen sind implementiert. Gitter Polymere werden anhand von Daten Strukturen basierend auf relativen "Zügen" als auch Distanz Matrizen und Kontakt Matrizen verglichen.
Mit Hilfe dieses Paketes wurden wichtige Eigenschaften von "Self Avoiding Walks" (SAWs) charakterisiert. Die mittlere Länge ℓ von "growing Self-Avoiding Walks" (gSAWs auf regulären Gittern terminieren sich oft selber) und die Form der Verteilung. Die minimale Länge von gSAWs sowie die Verteilung der relativen Richtungen wurde auch mit Hilfe von "exact enumeration" berechnet . Zum Abschlußwurde das Augenmerk auch noch auf erste Ausblicke über Landschaften [46] gelegt.
Man kann SAWs auch als ein Modell für Polymere in der physikalischen Chemie betrachten, welches eine der wichtigsten Anwendungen von ihnen ist. Wenn man realisiert, daß Proteine Polymere sind, wird man das Ziel und die Relevanz dieser Arbeit erkennen, nämlich ein Modell für Proteine zu erschaffen.

Contents

List of Figures

1 Introduction

1.1 SAWs

Imagine yourself standing at an intersection in the center of a large city whose streets are laid out in a square grid. You choose a street at random and begin walking away from your starting point, and at each intersection you reach you choose to continue straight ahead or to turn left or right. There is only one rule: you must not return to any intersection already visited in your journey. In other words, your path should be self-avoiding. It is possible that you will lead yourself into a trap, reaching an intersection whose neighbors have been visited already, but barring this disaster you continue walking until you have walked some large number N of blocks. There are two basic questions:

1. When will you get trapped ?

2. How many possible paths could you have followed ?

The above model is also called a Self-avoiding walk (SAW) [34]. It is a path on a lattice that does not visit the same site more than once. SAWs play a major role in polymer physics, where the main interest centers around equilibrium properties such as the number of configurations or the end-to-end distance of a polymer consisting of a fixed number of monomers n [17, 45]. Recently SAWs have received considerable attention as models for the folding of biopolymeres, in particular of proteins, see e.g. [7].

SAWs can be generated by a random walk on the lattice subject to the constraint that already occupied sites are inaccessible. Such a walk will get trapped whenever there are no neighboring unoccupied sites available. In fact, almost all SAWs constructed by this procedure terminate after a finite number $N = n - 1$ of steps [27]. We will call these SAWs *grown self-avoiding walks* (gSAWs). They are not only interesting in their own right [10] as restricted random walks, but also because trapping is an important source of non-ergodicity in models of polymer dynamics [35].

Myopic SAWs (sometimes also called "true" SAWs) are closely related gSAWs in that they exactly resemble gSAWs except for the trapped steps: myopic SAWs escape from a trap by moving to the neighbor that has been visited the least number of times in the past, thereby violating the self-avoidance condition [10, 15, 32].

In the present work we present a toolkit for working with lattice polymers and some properties of SAWs.

1.2 Lattice Polymers

Proteins are polymers !
One of the most important applications of the self-avoiding walk is a model for linear polymer molecules in chemical physics [34].
A *polymer* is a molecule that consists of many "monomers"(groups of atoms) joined together by chemical bounds. The *functionality* of a monomer is the number of available bonds that it has, i.e. the number of other monomers with which it must bond. If each monomer has functionality two, then a linear polymer is formed. If we denote a monomer by (A), then a linear polymer may be represented schematically as

$$\ldots\text{-(A)-(A)-(A)-(A)-(A)-}\ldots.$$

One simple example is polyethylene, where each monomer is CH_2 (one carbon and two hydrogen atoms). The pattern terminates either by bonding with a monomer of functionality one, such as CH_3, at each end, or else by closing on itself to form a "ring polymer". By way of contrast, if a polymer includes monomers of functionality three or more, then a *branched polymer* is formed; these are often modeled by lattice trees or lattice animals.
The following chapter deals only with the topological structure of a polymer. Properties of its spatial configuration are no less important. Polymers can be very large; some linear polymers consist of more than 10^5 monomers. Thus the length scale of the entire polymer is macroscopic with respect to the length scale of individual monomers. Consider a linear polymer consisting of n monomers, and label the monomers $1, \ldots, n$ from one end to the other. Let $x(i) \in \mathbf{R}^3$ denote the location of the i-th monomer. Then the i-th (monomer-monomer) bond may be represented by the line segment joining $x(i-1)$ to $x(i)$. Typically, the length of each bond is essentially constant throughout the chain, as is the angle between each pair of consecutive monomer-monomer bonds. However, there is some rotational freedom for the i-th bond around the axis determined by the $(i-1)$-th bond. In some cases a reasonable good approximation may be obtained by allowing the rotational angle of the i-th

bond around the $(i-1)$-th bond to take three different values, say 0 and ± 120 degrees, perhaps with different probabilities (an angle of 0 degrees means that the i-th,$(i-1)$-th,and $(i-2)$-th bonds all lie in one plane). These angles correspond to local configurations of minimal free energy, and depend on the details of the monomers.

We see that one possible model for the spatial configuration of a linear polymer is simply a random walk in $\mathbf{R}^3$, and in fact this model is known as the *ideal polymer chain.* Alternatively, one can work with a lattice approximation, say a random walk on $\mathbf{Z}^3$. The model can be embellished by turning it into a Markov chain (or random walk with some finite memory), and it works reasonable well in some situations. However, there is fundamental limitation of the ideal polymer chain, namely the *excluded volume* effect.

Two monomers cannot occupy the same position in space: the presence of a monomer at position x prohibits any other part of the polymer from getting too close to x, that is, other monomers are excluded from a certain volume of space. This is the excluded volume effect. When we take this effect into account, it becomes apparent that a self-avoiding walk is a more appropriate model for a linear polymer than is a random walk. The *self-avoiding walk model* is best for the case of a dilute polymer solution (where polymers are far apart, so that there is little interaction between distinct molecules) and a good solvent (which minimizes attractive forces between monomers).

On the other hand, there are some situations in which polymers really do behave ideally on large length scales, even though excluded volume effects are present. One is in a dense system (or "melt") of many polymers, where monomers fill three dimensional space uniformly and a given polymer interacts with many other monomers besides its own. Another is at certain values of temperature and solvent quality were roughly speaking the attractive forces between monomers exactly balance the excluded volume repulsion (the "Θ point").

Linear polymers in dilute solutions are believed to be in the "same "universality class" as the self-avoiding walk, which means in particular, that they have the same critical exponents. For example, consider the *radius of gyration* of a polymer, which is the average distance of the monomers from the center of the mass of the polymer. The radius of gyration of polymers can be determined experimentally, for example from light scattering properties. For a polymer consisting of n monomers, the radius of gyration is expected to be asymptotic to DN^{ν} as $n \to \infty$, where D and ν are constants. The exponent ν is believed to be *universal*: it should be the same for all linear polymers (in dilute solution with good solvents), and for the self-avoiding walk as well. Moreover the exponent ν for the radius of gyration is believed to be the same as the critical exponent ν for the mean square displacement, since polymers are expected to have only one macroscopic length scale. In contrast, the amplitude D is non-universal: it depends on microscopic details of the monomers and the solvent molecules. Paul J. Flory [16] developed an effective method for computing the exponent ν.

Dill [6] proposed also a model for lattice polymers. According to his model chains are configured on three-dimensional simple cubic lattices in which each lattice site is occupied by no more than one monomer. Monomers are numbered sequentially, $1, 2, 3, \ldots$ from one end of the chain (figure 1). Contacts are identified when a pair of unconnected monomers occur in nearest-neighboring lattice sites.

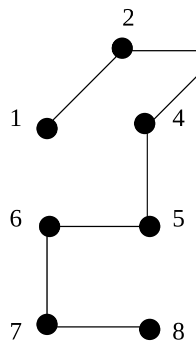


Figure 1: Sample of a 3D lattice polymer according to DILL

On the simple cubic lattice for example (as on the square lattice), only odd-order contacts are possible. Even order contacts are forbidden *a priori*. Conformational freedom is affected by the presence of presumed contacts along the chain.
One consequence is the appearance of *inferred blocks*, contacts that are forbidden by a given set of presumed contacts. This is an effect of excluded volume.
Since the importance of excluded volume diminishes with increasing spatial dimensionality, for a given set of presumed contacts, fewer blocks are implied

in three dimensions than in two dimensions. In fact, it can be proved by explicit construction that for a single presumed contact with order $k > 3$, there is *no* inferred block in the three dimensional cubic lattice. In contrast, corresponding inferred blocks are always present on two dimensional square lattices.

The exact lattice enumerations are used to determine the number of conformations c under various constraints. In particular c_n is the total number of accessible conformations with N bonds ($N + 1$ monomers), without any constraints except for the requirement of excluded volume. Due to the approximate exponential scaling of c_n as a function of N ($N \to \infty$),

$$c_n \sim N^{\gamma-1}\mu^N \tag{1}$$

only short chains can be exhaustively enumerated. Dill confirmed the scaling (1) by exact enumerations and the results are consistent with the prediction of re-normalization group analysis. c_n for the simple cubic lattice has been computed by Sykes for $N \leq 16$, based on his data, c_n for $N \geq 17$ can be determined by extrapolation of Eq.(1), with the estimates $\gamma \simeq 1.1667$ for any three dimensional chains and $\mu \simeq 4.682$ for simple cubic lattices. In the present enumerations of c, the two ends of the chains are considered to be distinguishable.

2 Biopolymeres - Biological Introduction

2.1 Protein Structures

In biopolymeres forces determining the shape of the molecule are of same nature and so of same strength as intermolecular forces. Biopolymer structures depend on those forces more than one might expect. For example globular proteins have their biologic active conformation just in a very small range of pH, temperature and ion-strength. One could suspect, that this could be a contradiction to the fact, that we obtain most of our structural information through X-ray crystallography, but if we take a closer look at the surrounding of the cristalls we find a lot of H_2O-molecules and ions, bound onto the protein-cristall. Furthermore there exists quite a big amount of mobile H_2O-molecules. So we could assume, that molecules in a proteincristall are in a quasi-natural environment.

2.2 Natural Structures (Repetitive Units)

The secondary structure of a protein or biopolymer is build out of the primary sequence, using H-bonds connections inbetween polypeptide chain members **CO** and **NH**. It makes only sense to call a part of the secondary structure structural element, if it is easy to recognize and represented in several proteins. Following those criteria you can find three structural elements:

- Helices
- β-Sheets
- Turns

The secondary structures are of particular interest to learn more about the mechanism of protein folding. Using a plausible folding-model secondary structures are developed, which serve folding of the remaining protein [30] [1].
Secondary structures also are important for theory and prediction of protein-structures, because conventional conceptions are build upon understanding tertiary structures by using “forming-rules” of secondary structures.

In the following we will discuss some structural elements of proteins. In chapter 3.3 our abstract models as well as different types of displaying structural elements will be presented.

2.2.1 Helices

The prediction of α-Helices as essential structural element in proteins made by Linus Pauling and Robert Corey [31] turned out to be a milestone in the understanding of biopolymeres. It is a right-handed helix (figure 2) of the polypeptide-chain. Each Amino acid residue in a α-helix forms a h-bond with its own CO-group and the fourth next NH-group:
$k \rightarrow k+4; k = 1, 2, 3, \ldots$. By creating a α-helix the polypeptide chain transforms into a more compact and more stabile form.

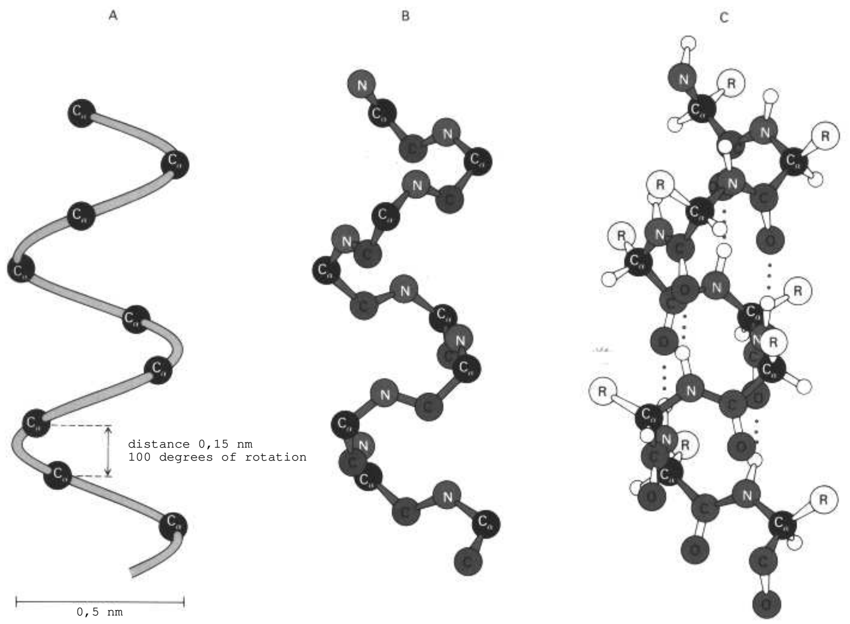


Figure 2: α Helix: A; only α-C-atoms(C_α). B; C_α+ N + C of the backbone. C; total helix

Helical structures can be defined in several different ways. One of them is by announcement of the dihedral angles ϕ, ψ and ω (figure 3). In nearly all structures ω is approximately 180°.

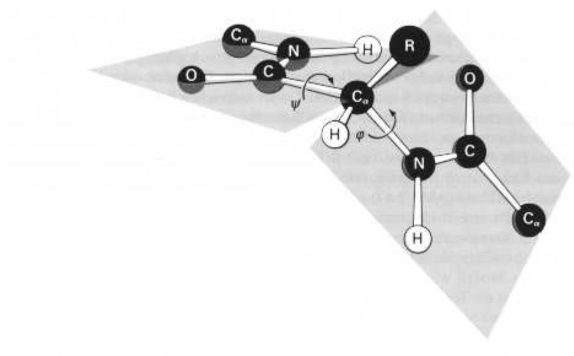


Figure 3: Definition of ϕ, ψ

Another characterization builds upon the number of amino acids n completing one full turn. In addition the number m, which counts the atoms forming a ring, by creating a CO...HN H-bond, has to be given (figure 4).

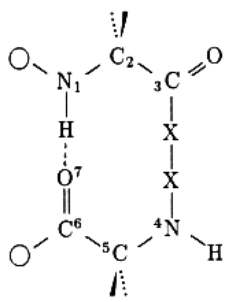


Figure 4: H-bonds in a helix

The continuation of the polypeptide-chain is indicated by the two circles. "X" symbolizes amino acids inbetween the H-bond. The helix will be fully described in the following manner: n_m The 2.2_7-helix has got the smallest ring, here there exists no "X".
Still there is an important parameter: h it is a rate for the polypeptide contraction in the translational direction.The parameter h is measured in units of Ångstrøms
Questions about stability you answer through (ϕ, ψ)-potential-fields, which have great similarity to Ramachandran-plots (figure 5[38]).

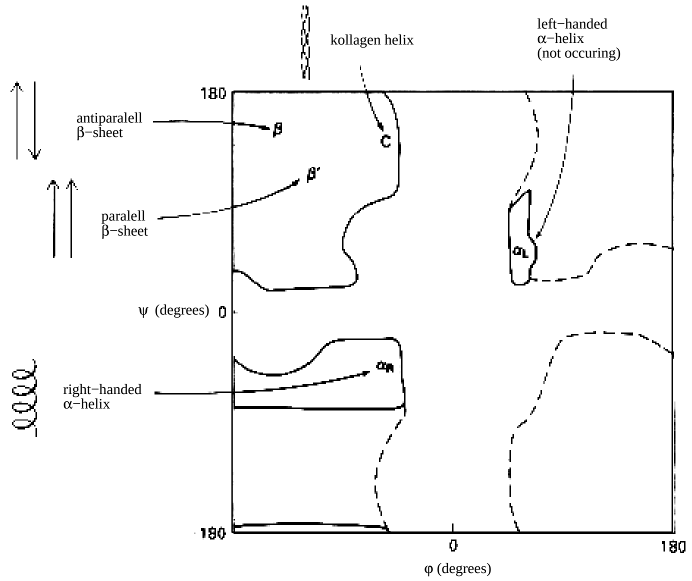


Figure 5: Ramachandran plot; poly-L-*ala*

In addition to the right hand sided helices there are left hand sided helices: $\psi_{lh} = -\psi_{rh}$ and $\phi_{lh} = -\phi_{rh}$. These helices are of course *no* mirror images, because all amino acid residues,except *gly*, are *chiral* and only the L-forms are represented.

2.2.2 β-Sheets

In the second important form of secondary structure elements the CO...HN H-bonds are made between two different parts of the polypeptide chain. In consideration of the orientation you will get a *parallel* or *anti-parallel* β-sheet. In general they are named β-structures, because the crystalline form of poly-L-*ala* forms an antiparalell sheet. The molecular properties are shown in figure 6. We have to notice, that there is a *non* local problem in comparison to the α-helix .
The side chains are alternatively orientated to both sides. There is only one exception: L-*pro*, because there is no hydrogen bond to the nitrogen and in addition it cannot rotate to the required angles (ψ, ϕ).

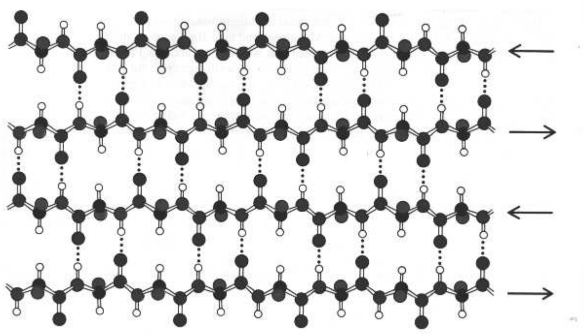


Figure 6: Antiparallel β-sheet

2.2.3 Turns

Globular proteins have mostly of spherical shape, so they can't form longer α helices or β sheets then their diameter. The polypeptide chains have to change their directions with help of turns.

- hairpin- or β- turns
  Within β-sheets very "sharp" turns often occur. They exist out off four amino acid-residues. Between CO of the first and NH of the fourth amino acid there is a H-bond. Hairpins imply following restrictions in the occurance of certain residues in special positions of the different types:

  I: all residues are allowed on positions 1-4 except *pro* in position 3,
  I': positions 2 and 3 must be *gly*,
  II: position 3 must be *gly*,
  II': position 2 must be *gly*,
  III: is part of a 3.0_{10}-helix (no more restrictions),
  III': positions 2 and 3 must be *gly*,
  VI: must have *pro* in position 3 and a cis-peptide bond in front.

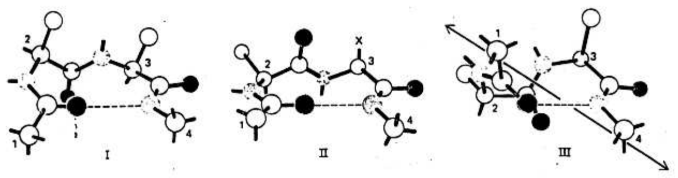


Figure 7: Most frequent hairpins. X marks the position where an H-atom is required. Type III shows a strong similarity to a 3.0_{10}-helix (axes marked)

- Ω-turns
  There still is a doubt whether it is correct or not to classify Ω-turns as a secondary structure element. They are turns with the length of approximately 6-16 residues, which combine other secondary structure elements and have an end-to-end distances of 3.7-10 Å. In addition the residue-distribution is different comparing to α-helix and β-sheets. In particular *gly* and *pro* occur very often.

3 Self Avoiding Walks

3.1 Characterization of SAWs

3.1.1 Enumeration of SAWs

Let $SAW(\mathcal{L}, n)$ denote the set of SAWs of length n on the lattice $\mathcal{L}$ starting at the origin. Let $c_n = |SAW(\mathcal{L}, n)|$ be the number of SAWs with n nodes starting at the origin of some lattice $\mathcal{L}$. By $c_n(x)$ we denote the number of SAWs starting at the origin and ending in $x \in \mathcal{L}$.

The mean-square end-to-end distance is defined by

$$\bar{\omega}_n = \frac{1}{c_n} \sum_{x \in \mathcal{L}} \|x\|^2 c_n(x), \tag{2}$$

where $\|x\|$ is the distance of x from the origin.

dim	Lattice Type	μ	ref.	γ	ref.	ν	ref.
2	Square	2.63820	[2][25][12]	1.34275	[25]	0.750	[25][12][36]
	Triangular	4.15076	[25][13][26]	1.343	[25]	0.590	[25][47]
	Hexangular	1.84777	[25]	1.345	[25]	*	
3	SC	4.68391	[25]	1.161	[25]	0.592	[25]
	BCC	*	[39]	*		*	
	FCC	10.0364	[25]	1.162	[25]	0.592	[25]

Table 1: Asymptotic enumeration of SAWs.

It is generally believed that

$$\begin{aligned} c_n &\sim n^{\gamma-1}\, n^{\mu} \\ \bar{\omega}_n &\sim n^{2\nu} \end{aligned} \tag{3}$$

Obviously the most interesting constant is μ, the effective coordination number of the lattice. So we have $\mu \leq \deg[\mathcal{L}] - 1$, the degree of the lattice. The generating function

$$\chi(\beta) = \sum_n c_n\, \beta^n \tag{4}$$

is believed to be of the form

$$\chi(\beta) = A(\beta) + B(\beta)(1-\mu\beta)^{-(\gamma+1)}\left[1+\sum_{i=1}^{K-1} C_i(\beta)(1-\mu\beta)^{-\Delta_i}\right] \quad (5)$$

where A, B, C_i are analytic and $K \geq 1$.

3.1.2 Radius of Gyration and its Relatives

1. Radius of Gyration

   The radius of gyration is defined as the mean value of the quadratic distance between the coordinates of each monomer and the centre of mass of the chain.

   $$s^2 = (n+1)^{-1}\sum_0^n s_i^2 \quad (6)$$

   s_i is the distance of the $point_i$ of the chain. n are the number of chain-molecules.

2. End to End Distance

   The vector that combines the two ends is defined as follows:

   $$r = \sum_{i=1}^{n} I_i \quad (7)$$

   in which I_i are the binding vectors of each segment.

   You can get another interesting property according to a theorem by Lagrange:

   $$s^2 = (n+1)^{-2}\sum_{0\leq i<j\leq n} r_{ij}^2 \quad (8)$$

3. Number of Contacts The number of contacts of a SAW is

   $$N_{\natural}(\xi) = \sum_{i<j} a_{ij} \quad (9)$$

a_{ij} refers to the contact matrix (see chapter 3.3.1). By $N_\natural^{max}(n)$ we denote the maximum number of contacts in a SAW of length n. The *compactness* of a SAW ξ of length n is defined as

$$Q(\xi) = N_\natural(\xi)/N_\natural^{max}(n) \tag{10}$$

The numbers $N_\natural^{max}(n)$ depend on the lattice.

3.1.3 Flory Model

As mentioned before Paul J. Flory [16] developed an effective method for computing the exponent ν . Here we give a brief description of this method: For simplicity, we ignore all multiplicative constants. Fix N and consider a linear polymer with $N+1$ monomers, represented by an N-step walk $\omega = (\omega(0), \ldots, \omega(N))$ in $\mathbf{Z}^d$ (d=dimension; not necessarily self -avoiding). Let L be the radius of gyration of ω, or any *effective radius* of the walk. Then ω consists of $N+1$ monomers (sites) through a box of volume L^d. Assuming uniformity, this gives a density of

$$\rho = \frac{N}{L^d} \tag{11}$$

monomers per unit volume. The repulsive energy per unit volume depends on the number of pairs of monomers per unit volume, which we approximate by ρ^2. This is a "mean field" approximation: it uses the assumption of uniformity very heavily, ignoring the strong correlations in the locations of consecutive monomers along the polymer. If we accept this approximation, then the local repulsive energy of the polymer is given by

$$E_{rep} = L^d \rho^2 = \frac{N^2}{L^d}. \tag{12}$$

Naturally the repulsive energy is lower for highly extended chains, i.e., large values of L.
Now consider the free energy F of the polymer of radius L, in the absence of the repulsion. This is given (up to constants) by (-1) times the entropy [1] and

[1]In thermodynamics we have $F = U - TS$, where U is internal energy, T is the temperature, and S is entropy. Here U depends on the number of monomers but not in L, so for our purposes it is constant and hence we ignore it.

the entropy in turn is just the logarithm of the number of walks of radius L. Without repulsion, this can be found from the Gaussian behavior of the ideal chain, as follows. Taking L now to denote the end-to-end distance and fixing $\omega(0) = 0$, we have

$$Pr\{\omega(N) = x\} \approx N^{-\frac{d}{2}} \exp(\frac{-|x^2|}{N}) \tag{13}$$

for every $x \in \mathbf{Z}^d$, and hence

$$Pr\{|\omega(N)| = L\} \approx \frac{L^{d-1}}{N^{\frac{d}{2}}} \exp(\frac{-L^2}{N}). \tag{14}$$

The total number of N-step walks is $(2d)^N$ in the nearest neighbor case, so the free energy is

$$\begin{aligned} F &= -\log[(2d)^N Pr\{|\omega(N)| = L\}] \\ &= -(d-1)\log L + \frac{L^2}{N} + \text{terms independent of } L. \end{aligned} \tag{15}$$

The term F may also be viewed as an "elastic energy" term, which prevents L from getting too large. The total energy of the polymer is now given by the sum of the two energy terms (12) and (15):

$$E_{rep} + F = \frac{N^2}{L^d} + \frac{L^2}{N} - (d-1)\log L + K, \tag{16}$$

where K is independent of L. Now put $L = N^\nu$. Then the total energy (16) becomes

$$E_{rep} + F = N^{2-d\nu} + N^{2\nu-1} - \nu(d-1)\log N + K. \tag{17}$$

The value of ν that minimizes the energy (17) may be found by first equating the first two powers of N: solving $2 - d\nu = 2\nu - 1$ gives

$$\nu = \frac{3}{d+2}. \tag{18}$$

Substituting this back into (17), the first two terms become $N^{(4-d)/(d+2)}$and these are the dominant terms if and only if $d < 4$. Therefore this argument predicts that (18) gives the correct value of ν whenever $d < 4$.

When $d = 4$, this argument also predicts $\nu = 3/(4+2) = 1/2$ since this is the only value for which the first two terms of (17) bounded. Pushing this argument further suggests that we should take the largest value in this interval so as to minimize the $-\nu(d-1)\log N$ term in (17), obtaining $\nu = 1/2$ for $d > 4$. This answer makes sense: since ν equals 1/2 in the ideal case, the addition of a repulsive energy term should not decrease ν below 1/2, and so we conclude that $\nu = 1/2$ whenever $d > 4$.
To summarize, the above argument makes the following predictions for ν:

$$\nu_{Flory} = \begin{cases} 1 & if \quad d = 1 \\ 3/4 & if \quad d = 2 \\ 3/5 & if \quad d = 3 \\ 1/2 & if \quad d \geq 4. \end{cases} \tag{19}$$

These predictions are known as the *Flory values* for ν. They are known to be correct for $d = 1$ and $d \geq 5$, and they are believed to be correct for $d = 2$ and $d = 4$ as well. The Flory value for $d = 3$ is generally believed to be slightly too large: numerical and field theory calculations indicate that the actual value is probably close to 0.59. The success of Flory's argument is all the more remarkable when one realizes that it benefits greatly from the cancelation of two errors: both E_{rep} and F are greatly overestimated [11].
Last but not least, we shall recast the Flory argument in a more probabilistic language. In (14) we calculated the probability that an N-step random walk ω (starting at the origin) has $|\omega(N)| = L$. We shall write $L = N^\nu$ and choose ν to maximize this probability. As above, we assume that the $N+1$ sites of ω are spread uniformly through a box of volume L^d. Given that $\omega(0), \ldots, \omega(k-1)$ are all distinct, the probability that $\omega(k)$ does not coincide with any one of the previous k sites is approximately $1 - kL^{-d}$ (this is the mean field approximation). Hence the probability that ω is self-avoiding given that $|\omega(N)| = L$ is approximately

$$\prod_{k=1}^{N}(1 - kL^{-d}) \approx \exp\left(-\sum_{k=1}^{N} kL^{-d}\right) \approx \exp(-N^2/L^d). \tag{20}$$

Multiplying (20) by (14) yields

$$Pr\{\omega \text{ is self avoiding and} |\omega(N) = L\} \approx \frac{L^{d-1}}{N^{d/2}} \exp[-\frac{L^2}{N} - \frac{N^2}{L^d}]. \tag{21}$$

To find the most likely value of L, we maximize the above probability for fixed N. Since the logarithm of this is just the negative of the total energy (16), we are again led to the Flory exponents.

3.2 Representation of SAWs

We consider here regular[2] lattices together with a set of moves at each lattice point with the following properties: Let $y = x + m$ denote the lattice point obtained by attaching the move m to site x, then there is a move m^* such that $x = y + m^*$, i.e. $m^* = -m$ is a move as well. Furthermore we require that for all lattice points x, y and all moves m, m' there is a symmetry operation ψ of lattice such that $y = \psi(x)$ and $y + m' = \psi(x + m)$.

A walk on a lattice is completely described by its initial point, 0, and the ordered list of the N moves. Let us denote by $\mathcal{A} = \{a_1, \ldots, a_M\}$ the set of all possible moves in absolute coordinates. Not all of these M moves can be realized at each lattice point. As an example consider the hexagonal lattice, `HEX`. There are 3 possible moves at each lattice site, out of a total of 6 absolute directions. We define the set $\mathcal{R} = \{R_1, \ldots, R_m\}$ of "relative" directions as the subset of $\mathcal{A}$ consisting of all moves allowed at the reference site 0.

The representation of a walk in terms of moves in absolute coordinates is not always convenient. Instead we adopt a local point of view by rotating the coordinate frame after each move R_k in such a way that $-R_k$ is assigned the backwards direction B in the rotated coordinate system. In other words, for each relative move R there is a rotation $\tilde{R}$ of the coordinate system fulfilling $\tilde{R}(R) = -B$. Note that $\tilde{R}$ is not always uniquely defined. As a consequence of the symmetry requirements above the allowed moves for the next step expressed in the rotated coordinate system are again the relative moves $\mathcal{R}$. The coordinates of the point y_k obtained as the k-th step of the walk are given by $y_k = y_{k-1} + a_k$. The absolute direction a_k of the k-th step is uniquely defined by the relative direction R_k of this step, *and* the absolute direction a_{k-1} of the previous step. Instead of defining the rotations $\tilde{R}_k$ explicitly, we can directly consider the mapping

$$\tau : \mathcal{A} \times \mathcal{R} \to \mathcal{A}, \ (a, R) \mapsto \tilde{R}^{-1} a$$

that — given the lattice — determines the absolute direction of a step given its relative direction and the absolute direction of the previous step. Once we have derived the mapping τ we have a unique encoding of each walk as a

[2] We call a lattice regular if for any two lattice points x and y there is a symmetry operation σ of the lattice such that $x = \sigma(y)$ and if for any two pairs of neighbors (u, v) and (x, y) there is a symmetry operation τ of the lattice such that $u = \tau(x)$ and $v = \tau(y)$.

string of length N over the alphabet of the m relative directions. For SAWs we only need $z = m - 1$ relative directions, since the backwards direction B never occurs. As a by-product, point mutation of these strings are pivot moves [36].
The number of neighbors n and possible moves z are enlisted in the depicted tabular for all the lattices:

lattice	dim	n	z
HEX		6	2
SQ	2	4	3
TRI		6	5
KM		4	7
SC		6	5
TET		8	3
BCC	3	8	7
FCC		12	11
TDKM		6	23

Here we explicitly give the mapping τ for the lattices : First we will describe the moveset of the 2D lattices SQ and TRI/HEX:

	f	r	l	b
F	f	r	l	b
R	r	b	f	l
L	l	f	b	r
B	b	l	r	f

	f	r	l	d	u	b
F	f	r	l	d	u	b
R	r	d	f	b	l	u
L	l	f	i	r	b	d
D	d	b	r	u	f	l
U	u	l	b	f	d	r
B	b	u	d	l	r	f

The HEX lattice can be treated as a subset of the TRI lattice, only "B,L,R moves" are relevant.

The definition of the directions for this, and all other lattices discussed here, can be found in the following figures. The mapping τ is invertible in the following sense: given the absolute directions a_{k-1} and a_k we can retrieve R_k. It is not possible in general, however, to calculate a_{k-1} given only a_k and R_k.

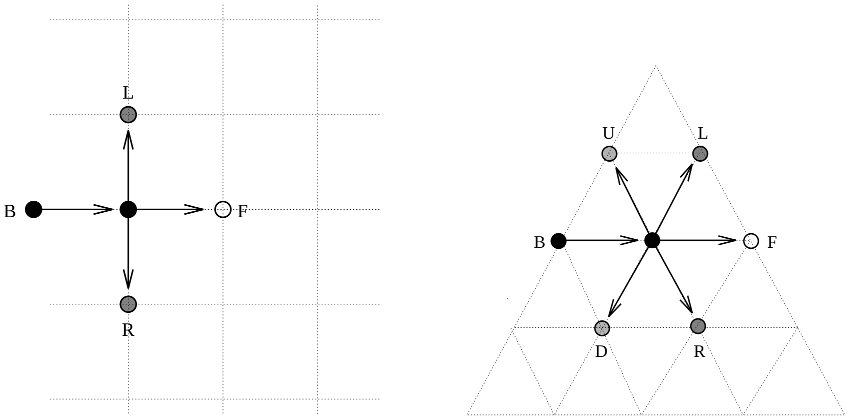


Figure 8: Moveset of SQuare TRIangular and HEXangular lattice

A very special lattice is the KM (*Knight's Moves*) lattice. as on KM the moves do not coincide with the neighbors and moves may in fact cross each other as long as they don't end in the same lattice point. In this sense in not really a 2D lattice.

	b	f	l	r	m	q	k	s
B	f	b	r	l	q	m	s	k
F	b	f	l	r	m	q	k	s
L	r	l	b	f	k	s	q	m
R	l	r	f	b	s	k	m	q
M	q	m	k	s	l	r	b	f
Q	m	q	s	k	r	l	f	b
K	s	k	m	q	f	b	l	r
S	k	s	q	m	b	f	r	l

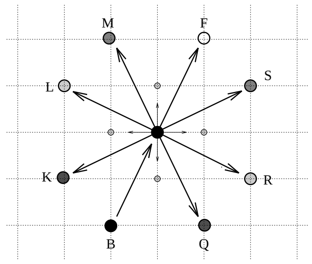


Figure 9: Moveset Knight's Moves

Tables for **3D Lattices:**

In three dimensions there is some arbitrariness in the definition of the relative directions. Note, however, that

$$x * f = f * x = x \tag{22}$$

i.e. the forward step is used as a the group identity. Furthermore b reverses the direction of the walk, hence its action is also uniquely determined. In a simple cubic lattice the moves R, L, U and D are equivalent with respect to the previous step.

	b	f	l	r	u	d
B	f	b	r	l	d	u
F	b	f	l	r	u	d
L	r	l	u	d	f	b
R	l	r	d	u	b	f
U	d	u	f	b	l	r
D	u	d	b	f	r	l

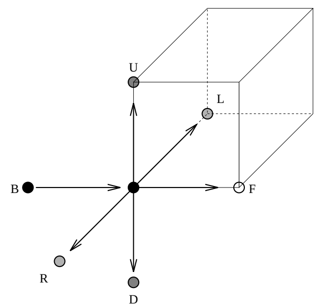


Figure 10: Moveset of Simple Cubic lattice

For a **Body-Centered Cubic** lattice we have:

	b	f	p	x	q	y	r	z
B	f	b	x	p	y	q	z	r
F	b	f	p	x	q	y	r	z
P	x	p	q	y	z	r	f	b
X	p	x	y	q	r	z	b	f
Q	y	q	z	r	f	b	x	p
Y	q	y	r	z	b	f	p	x
R	z	r	f	b	x	p	y	q
Z	r	z	b	f	p	x	q	y

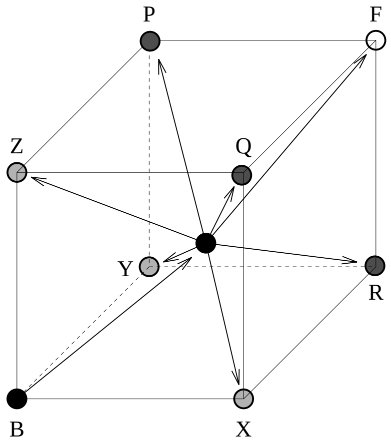


Figure 11: Moveset of Body Centered Cubic lattice

The **Diamond or TETragonal lattice** can be treated as a subset of the BCC lattice

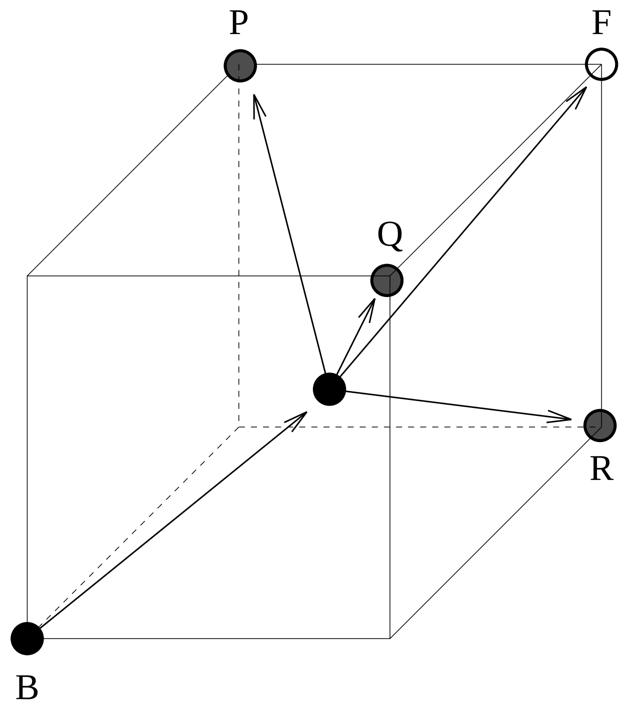


Figure 12: Moveset of TETragonal lattice

For a **Face-Centered Cubic** lattice we have:

	b	f	r	l	w	p	x	q	y	m	z	n
B	f	b	l	r	p	w	q	x	m	y	n	z
F	b	f	r	l	w	p	x	q	y	m	z	n
R	l	r	b	f	z	n	y	m	p	w	q	x
L	r	l	f	b	n	z	m	y	w	p	x	q
W	p	w	x	q	z	n	m	y	l	r	f	b
P	w	p	q	x	n	z	y	m	r	l	b	f
X	q	x	p	w	f	b	l	r	n	z	y	m
Q	x	q	w	p	b	f	r	l	z	n	m	y
Y	m	y	n	z	r	l	f	b	x	q	p	w
M	y	m	z	n	l	r	b	f	q	x	w	p
Z	n	z	y	m	q	x	w	p	f	b	r	l
N	z	n	m	y	x	q	p	w	b	f	l	r

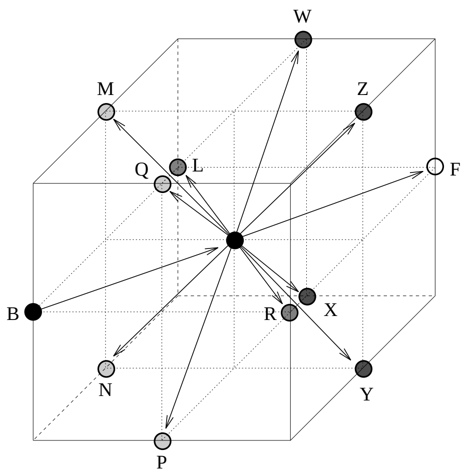


Figure 13: Moveset of Face Centered Cubic lattice

3D Knight's Moves are an interesting idea, but not very promising, because the cpu-consumption is enormous. There are 24 different moves: Have fun with the table !

	b	f	r	l	s	u	t	v	w	m	x	e	y	a	z	c	h	p	i	m	j	n	g	o
B	f	b	l	r	u	s	v	t	d	w	e	x	a	y	c	z	p	h	m	i	n	j	o	g
F	b	f	r	l	s	u	t	v	w	d	x	e	y	a	z	c	h	p	i	m	j	n	g	o
R	l	r	b	f	v	t	u	s	z	c	y	a	x	e	w	d	i	m	j	n	g	o	h	p
L	r	l	f	b	t	v	s	u	c	z	a	y	e	x	d	w	m	i	n	j	o	g	p	h
S	u	s	t	v	l	r	b	f	x	e	w	d	z	c	y	a	g	o	h	p	m	i	j	n
U	s	u	v	t	r	l	f	b	e	x	d	w	c	z	a	y	o	g	p	h	i	m	n	j
T	v	t	u	s	b	f	r	l	y	a	z	c	w	d	x	e	j	n	g	o	h	p	m	i
V	t	v	s	u	f	b	l	r	a	y	c	z	d	w	e	x	n	j	o	g	p	h	i	m
W	d	w	e	x	a	y	c	z	p	h	m	i	n	j	o	g	f	b	l	r	u	s	v	t
D	w	d	x	e	y	a	z	c	h	p	i	m	j	n	g	o	b	f	r	l	s	u	t	v
X	c	e	y	a	x	e	w	d	i	m	j	n	g	o	h	p	l	r	b	f	v	t	u	s
E	e	c	a	y	e	x	d	w	m	i	n	j	o	g	p	h	r	l	f	b	t	v	s	u
Y	x	e	w	d	z	c	y	a	g	o	h	p	m	i	j	n	u	s	t	v	l	r	b	f
A	e	x	d	w	c	z	a	y	o	g	p	h	i	m	n	j	s	u	v	t	r	l	f	b
Z	y	a	z	c	w	d	x	e	j	n	g	o	h	p	m	i	v	t	u	s	b	f	r	l
C	a	y	c	z	d	w	e	x	n	j	o	g	p	h	i	m	t	v	s	u	f	b	l	r
H	p	h	m	i	n	j	o	g	f	b	l	r	u	s	v	t	d	w	e	x	a	y	c	z
P	h	p	i	m	j	n	g	o	b	f	r	l	s	u	t	v	w	d	x	e	y	a	z	c
I	i	m	j	n	g	o	h	p	l	r	b	f	v	t	u	s	z	c	y	a	x	e	w	d
M	m	i	n	j	o	g	p	h	r	l	f	b	t	v	s	u	c	z	a	y	e	x	d	w
J	g	o	h	p	m	i	j	n	u	s	t	v	l	r	b	f	x	e	w	d	z	c	y	a
N	o	g	p	h	i	m	n	j	s	u	v	t	r	l	f	b	e	x	d	w	c	z	a	y
G	j	n	g	o	h	p	m	i	v	t	u	s	b	f	r	l	y	a	z	c	w	d	x	e
O	n	j	o	g	p	h	i	m	t	v	s	u	f	b	l	r	a	y	c	z	d	w	e	x

3.3 Contact Structures and Secondary Structure Elements

Structures, as generated by SAWs or folding algorithms are strings of length N-1 in the notion of relational moves. They can be converted into Contact Matrices `CM`, Bond Matrices `BM` and Distance Matrices `DM`. `BM`s and `CM`s can be converted into Contact Lists `CL`s and `BM`s into trees. Two matrices, representing two (different) structures in the same notion can be compared and distances between them measured. A detailed description of the structure distance measures is given in [3].

Further measures with respect to the nature of neighboring residues will be provided. For the comparison of `CL`s we need a table, showing the edit costs for each edit operation. Given a set of edit operations and edit costs, the edit distance is given by the minimum sum of costs along an edit path converting one structure into the other.

3.3.1 The Contact Matrix

For a SAW ξ on a lattice $\mathcal{L}$ let $A(\xi)$ be the adjacency matrix of the *non-bonding* contacts, i.e.,

$$a_{ij} = \begin{cases} 1 & \text{if} \quad (\xi_i, \xi_j) \in E(\mathcal{L}) \wedge |i-j| > 1 \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

We will call A the *contact matrix* of ξ. Obviously A is symmetric. Note that $A(\xi) = A(\xi')$ does not imply $\xi = \xi'$ in general. In fact, all SAWs without contacts have the same contact matrix $A = 0$.

The notion of a contact matrix can be generalized to p-shell matrices:

$$a_{ij}^{[p]} = \begin{cases} 1 & \text{if} \quad d_{\mathcal{L}}(\xi_i, \xi_j) = p \wedge |i-j| > 1 \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

where $d_{\mathcal{L}}$ is the graphical distance on $\mathcal{L}$. Of course, $A = A^{[1]}$ is again the contact matrix.

3.3.2 The Distance Matrix

The Distance Matrix is symmetric and contains the distances between two residues, where the nearest neighbors in the lattice are set to 1 by convention $d_{ij} = \|\triangle(x_i, x_j)\|$. Another description:

$$D_{ij} = p \Leftrightarrow a_{ij}^{[p]} = 1. \tag{25}$$

3.3.3 Definition of Secondary Structures on Lattices

Within structural elements chain molecules have multiple topological neighbors $(i_1 j_1)(i_2 j_2), \ldots$. The most elementary building block "of secondary structures" in globular proteins, i.e., helices, parallel and anti parallel sheets, is a set of two such contact pairs [7].

A definition, based on the analogy with real proteins has been given by Dill.

The analogon to an α helix in the case of a 2d SQ-lattice and a 3d SC-lattice was given in [6] and is depicted in some examples in Appendix B. It is essentially based on repetive units that can be found in the contact matrix which

in turn is based on the notion of nearest neighbors.
A general definition works a s follows (t is the minimal turnsize, s the step-size):
For an anti parallel-beta-sheet ($APBS$) we have:

$$c_{ij} = c_{i+s,j-s} = ... = c_{i+ps,j-ps} = 1 \tag{26}$$

when $j > i$ upstream in the chain, p the length of (=number of contacts in) the APBS.
For a parallel-beta-sheet (PBS) we have:

$$c_{ij} = c_{i+s,j+s} = ... = c_{i+ps,j+ps} = 1 \tag{27}$$

with the same conditions as above and the constraint that $j > i + p$ for the bend back. In both cases $p > 1$, so that 3 contacts are required as the minimum to define a unit.
In the case of the α-helix analogy, however, the definition is rather ambiguous. Various patterns that could possibly fit the definition are depicted in fig 32. Characteristic measures of elements in natural proteins and LP are summarized in table 3.3.3. The definition is essentially based on the size of a turn, for SC $t = 3$ (which is the minimal number to be added to a position on a given lattice to yield a turn).

$$c_{ij} = c_{i+s,j+s} = ... = c_{i+ps,j+ps} = 1 \tag{28}$$

with $j = i + turnsize$ (see Appendix A) and s the height of the helix. α-helices therefore appear as repetitive patterns with some of the contacts, corresponding to the turns, close to the diagonal.

$$\begin{aligned} c_{ij} = &\quad c_{i+u,j+u} = ... = c_{i+pu,j+pu} = 1 \\ c_{kl} = &\quad c_{k+u,l+u} = ... = c_{k+pu,l+pu} = 1 \\ c_{mn} = &\quad .... \end{aligned} \tag{29}$$

so that a subset where $i, j, k, l < i + u, j + u, k + u, l + u < ... < i + pu, j + pu$ and at least $i \neq k$ or $j \neq l$, where u is the length of the repetitive structure and $p \geq 2$ the number of units required to define a repetitive structure .

Contact Maps contain entries in an upper (or lower) triangular matrix if two residues are topological neighbors, i.e. they are adjacent on the lattice but not along the sequence.

lattice	turnsize	steps
HEX	5	2
SQ	3	1
TRI	3	1
KM	3	1
TET	3	2
SC	3	1
BCC	3	1
FCC	3	1
TDKM	3	1

Table 2: Lattice constants for recognition of structural elements

α-helical conformations appear as bands parallel and near the diagonal
parallel arrangements are parallel but far from the diagonal
antiparallel arrangements (α-helices, β-sheets) of residues appear as bands perpendicular to the diagonal.
According to Dill secondary structures are considered necessary to gain a high compactness as regular structures can be packed more densely [6]. Secondary structures reflect the geometry of the underlying lattice, the best compromise cited in [21] was the TDKM lattice.

In Appendix A a description of the program is given. In Appendix B some samples of contact maps and structures are depicted.

3.4 Grown SAWs and Self Trapping

As mentioned before gSAWs can be generated by a random walk on the lattice subject to the constraint that already occupied sites are inaccessible. Such a walk will get trapped whenever there are no unoccupied neighboring sites available.

3.4.1 Length of Self-Trapped SAWs

The length ℓ of a gSAW is the number N of steps accepted until the walk is trapped. It is known that gSAWs exhibit a characteristic length distribution with an exponential tail on the 2D square lattice `SQ`, and that they are remarkably short and localized on average [27]. In table 3 we compare the average walk lengths and the variances for a number of different lattices in both two and three dimensions.

lattice	d	z	$\langle\ell\rangle$	σ	a	γ	samples
HEX	2	2	70.9±0.2	50	35.2	1.01	60000
SQ	2	3	71.7±0.2	50	34.9	1.05	60000
TRI	2	5	77.9±0.7	54	37.4	1.08	60000
KM	2	7	3212±10	2400	1790	0.80	60000
TET	3	3	381±1.3	340	303	0.26	60000
SC	3	5	3997±38	3800	3610	0.11	10000
BCC	3	7	22447±296	21200	20000	0.12	5000
FCC	3	11	34963±1024	32000	29200	0.19	1000
TDKM	3	23	>2.000.000				

Table 3: Length ℓ of gSAWs

The average walk length for 3D Knight-moves (TDKM) is more than $2e^6$. We could find some trapped walks before, but here we reached the limits of our computer-resources.

It is surprising how little the three genuine $2D$ lattices HEX, SQ, and TRI differ in $a = \sigma_l^2/\langle\ell\rangle$ despite their different connectivities. The knights move lattice KM cannot be compared with them, as on KM the moves do not coincide with the lattices and moves may in fact cross each other as long as they do not end in the same lattice point. In this sense KM as mentioned before is not really a 2D lattice. For 3D lattices there is strong dependence of a on the connectivity z of the lattice; not surprisingly, the average length $\langle\ell\rangle$ increases with the connectivity z of the lattice.
Using $\gamma = (\langle\ell\rangle/\sigma)^2 - 1$ we find a significant difference in the shape of the distribution function ϕ between 2D and 3D lattices. While the genuine 2D lattices are consistent with $\gamma = 1$, and the KM lattices yields $\gamma \approx 0.8$, we find much smaller values in for the 3D lattices: $0.1 \leq \gamma \leq 0.26$, see table 2.

The following figures describe the distributions of the walk length in the 2D lattices.

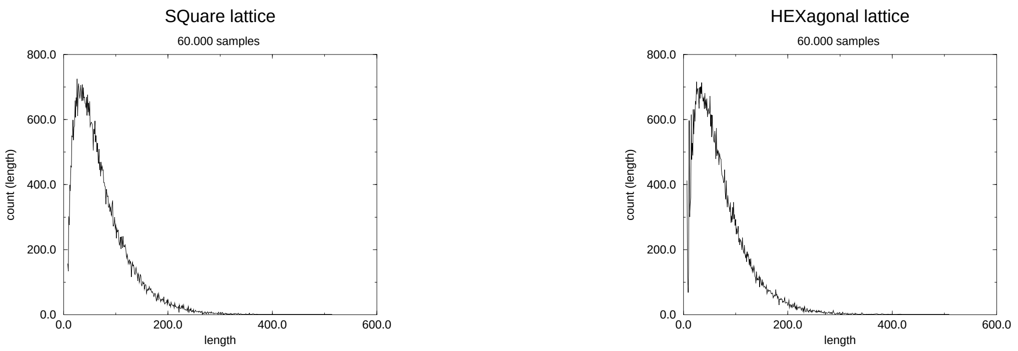


Figure 14: gSAWs;Distribution of walk length SQ / HEX

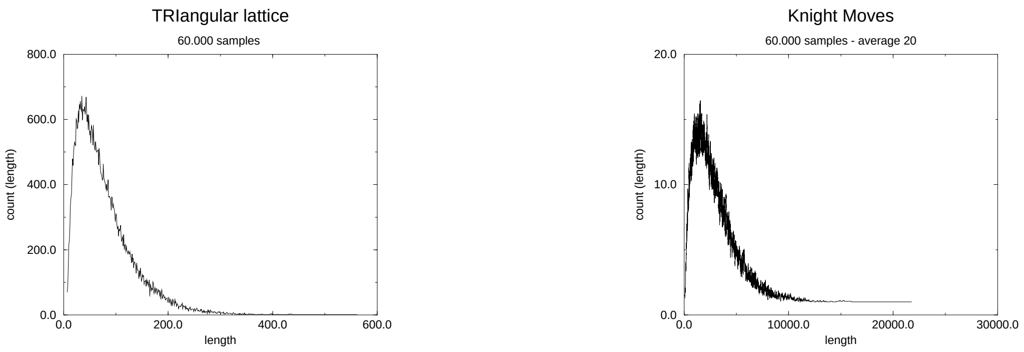


Figure 15: gSAW; Distribution of walk length. TRI + averaging of 20 (KM)

Within *3D lattices* you don't get so smooth curves, cause the spectrum of different walk lengths is much broader, as you can see, by looking at $\langle \ell \rangle$.

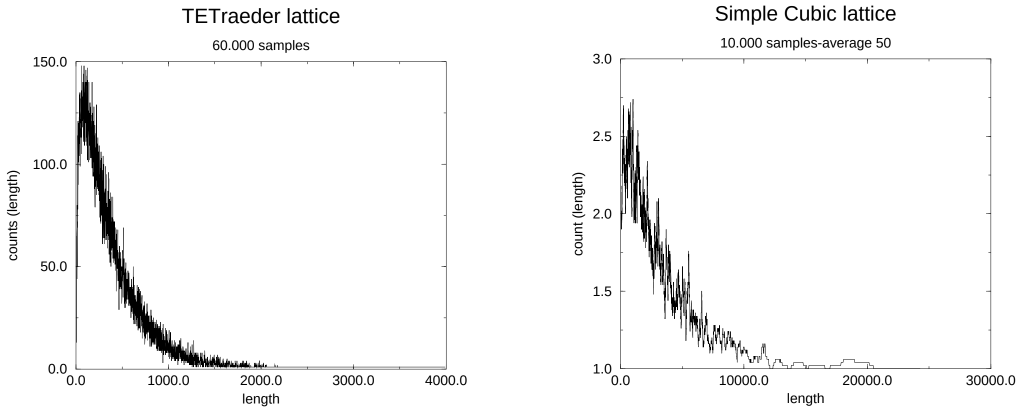


Figure 16: gSAW; Distribution of walk length. TET + averaging of 20 (SC)

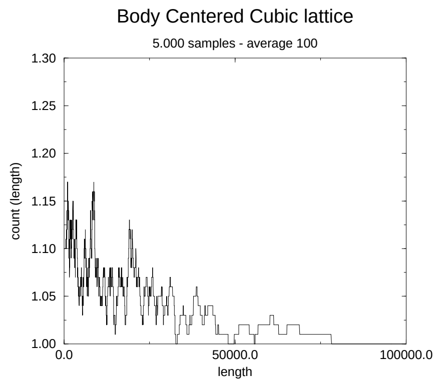


Figure 17: gSAW; Distribution of walk length.Averaging of 50 (BCC)

For the other lattices (FCC/TDKM) the same experiment was done, but here the sample size is to low to get accurate curves.

3.4.2 Fitting of 2D lattices

Hemmer [27] proposed following fit for the distribution of walk length in the SQuare lattice:

$$\phi(\ell) \approx (\ell - c_0)^\gamma \exp(-\ell/a) \tag{30}$$

we found this formula to fit reasonable well on the SQuare lattice as well, but then we found a better one:

$$\phi(\ell) = c_0 \ell^\gamma \exp(-\ell/a) \approx [a^{\gamma+1}\Gamma(\gamma+1)]^{-1}\, \ell^\gamma \exp(-\ell/a). \tag{31}$$

Consequently we obtain $\langle \ell \rangle \approx (\gamma+1)a$ and $\sigma_\ell \approx \sqrt{\gamma+1}a$ [22]. In table 3.4.1 the values for σ a and γ are included.
In the following graphs we proudly present the results:

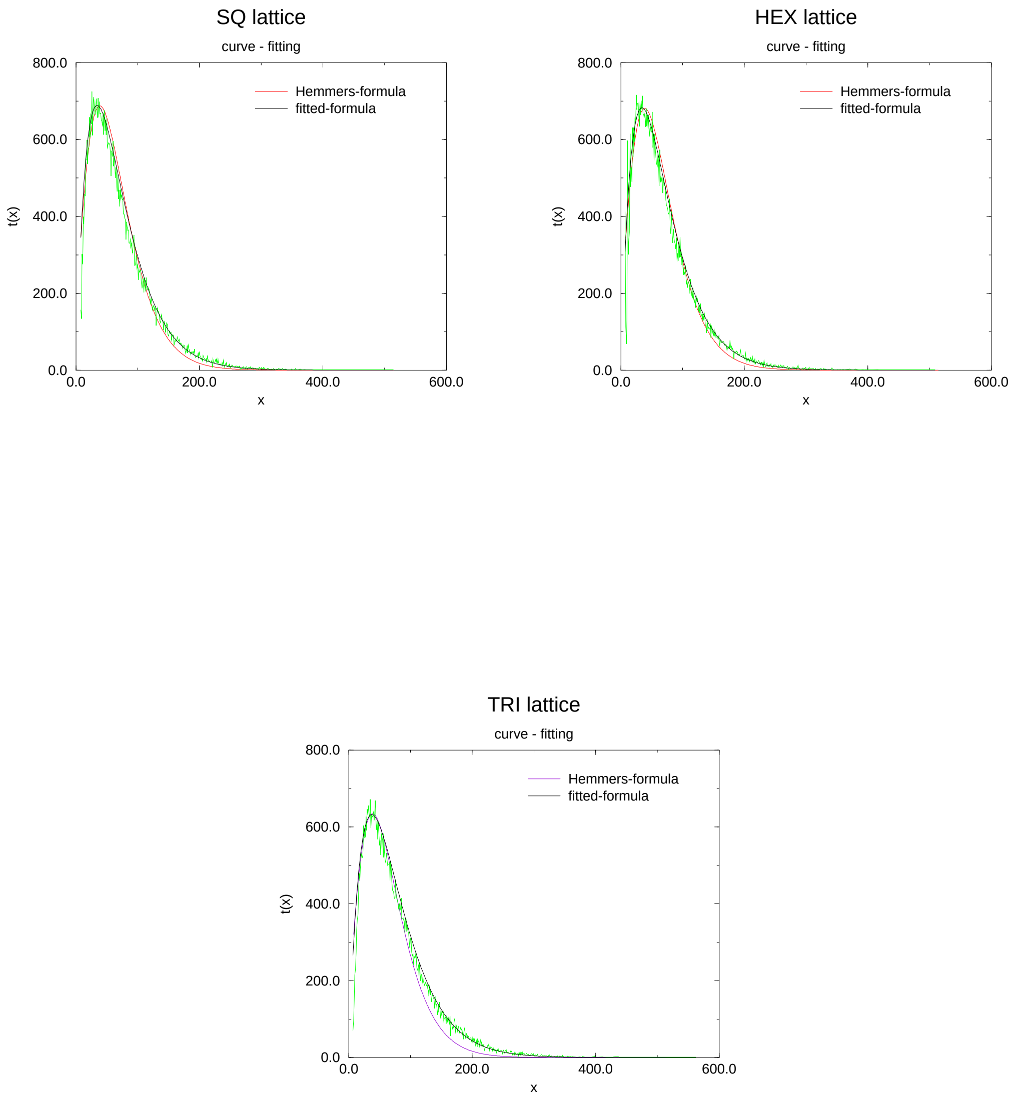


Figure 18: gSAW; Curve fitting-distribution of walk length SQ HEX TRI

3.5 Minimal Length gSAWs

The minimum length ℓ_{min} of a gSAW also depends strongly on the lattice. For most lattices we have obtained the exact values of the length ℓ_{min} of the shortest gSAWs by exhaustive search. Only for `FCC` the search space is too large so that we have to be content with upper bounds on ℓ_{min}. Examples for minimum length SAWs are given in table 4 and are depicted in the following figures.

Lattice	l_{min}	Example
TRI	6	FDDDDR
SQ	7	FRFRFRR
HEX	9	LDDDLDDDD
KM	15	FQLQLQLQLQLQLQK
TET	8	PQPPQRRP
SC	11	FLRLFDFLLRU
BCC	15	FXXYFXXZFZFYFXR
FCC	36	FZRFLZXLRNQFWXYNRYNNQYQMFZLXWQFRZQYW

Table 4: Minimum Length ℓ_{min} of gSAWs ©

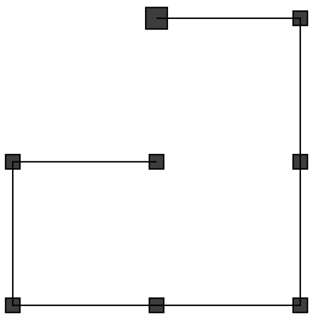


Figure 19: SQ lattice

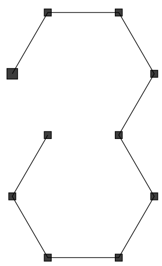


Figure 20: HEX lattice

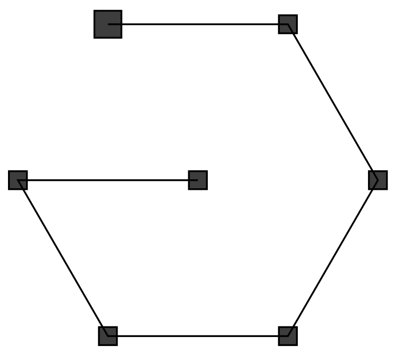


Figure 21: TRI lattice

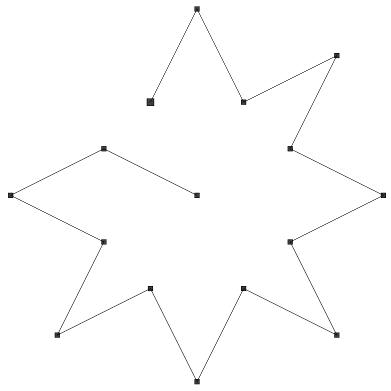


Figure 22: KM lattice

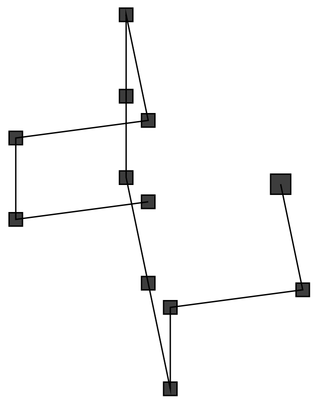


Figure 23: SC lattice

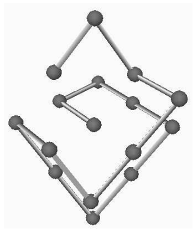


Figure 24: Minimum walk-length BCC

3.5.1 Distribution of Relative Directions

The relative moves $\mathcal{R}$ can be subdivided into symmetry classes with respect to the set of all symmetry operations of the lattice that leave both 0 and the backwards direction B invariant. For instance, in a square lattice there are 3 symmetry classes. One consists of the backward direction B only, the second class consists of the forward direction $F = -B$, and third class consists of the directions left L and right $R = -L$. The backward direction B forms a class by itself in all lattices, which never occurs in self-avoiding walks, of course. Table 6 clearly shows that the relative directions do not occur with equal

Lattice	rel. dir†	Frequency gSAW		Frequency SAW‡	
HEX	R L	1.0000		1.0000	
SQ	F	0.3694	(6)	0.4007	(5)
	R L	0.3153	(4)	0.2997	(3)
TRI	F	0.2343	(7)	0.2643	(8)
	R L	0.2088	(2)	0.2215	(3)
	U D	0.1739	(2)	0.1464	(2)
KM	F	0.1470	(2)	0.1888	(4)
	R L	0.1420	(1)	0.1350	(3)
	M Q K S	0.1429	(1)	0.1353	(3)
TET	P Q R	1.0000		1.0000	
SC	F	0.2085	(3)	0.2396	(7)
	R L U D	0.1977	(3)	0.1901	(5)
BCC	F	0.1494	(4)	0.1870	(7)
	P Q R	0.1450	(3)	0.1388	(6)
	X Y Z	0.1385	(3)	0.1322	(6)
FCC	F	0.0979	(3)	0.1707	(7)
	M N P Q	0.0888	(2)	0.0863	(4)
	R L	0.0905	(2)	0.0815	(4)
	W X Y Z	0.0917	(2)	0.0803	(4)

Table 5: †:The classes of relative direction s are arranged in the order of increasing angle from the forward direction. ‡ Data for free SAWs have been produced by the pivot algorithm. Data correspond to SAW of length approximately $\langle \ell \rangle$, although the frequencies of relative moves exhibit only very small variations with chain length.

frequencies in SAWs. There is a strong bias towards "forward" steps, while the directions closer to "backwards" are strongly disfavored. This effect is not surprising, since "backward" steps lead to more restricted choices for the next moves, and hence are more likely to lead to trapping. It is interesting to note, however, that the bias towards "forward" moves is much stronger in free SAWs than in gSAWs.

3.5.2 Exhaustive Search

Exhaustive search for SAWs was performed, using a recursiv programmed utility. We obtained the same data as Guttmann ([24][23][14]. In addition we made calculations for our other lattices. It is obvious, that the number of SAWs depends on the moveset and dimension of the lattice (fig. 25).

The numerical results are enlisted in the following table and compared to the known results:

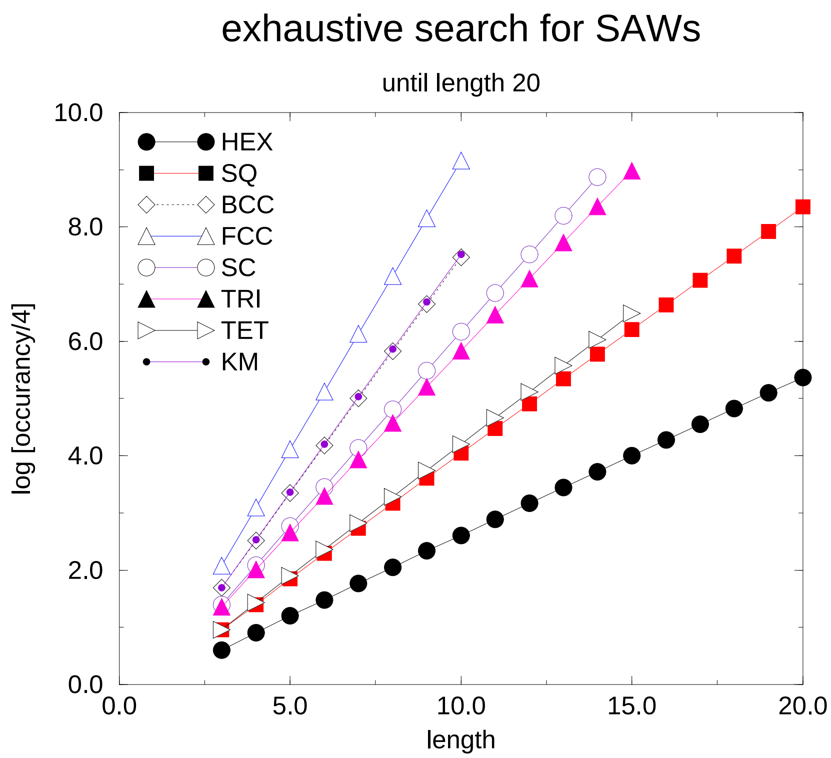


Figure 25: Exhaustive search for SAWs

n	SQ	HEX	SC	TET
0	1	1	1	1
1	4	2	6	4
2	12	4	30	12
3	36	8	150	36
4	100	16	726	108
5	284	32	3 534	316
6	780	60	16 926	916
7	2 172	116	81 390	2 664
8	5 916	224	387 966	7 696
9	16 268	432	1 853 886	22 188
10	44 100	812	8 809 878	63 728
11	120 292	1 552	41 934 150	183 240
12	324 932	2 944	198 842 742	525 104
13	881 500	5 592	943 974 510	1 505 236
14	2 374 444	10 520	4 468 911 678	4 305 164
15	6 416 596	19 928	21 175 146 054	12 319 304
16	17 245 332	37 512	100 121 875 974	
17	46 466 676	70 800		
18	124 658 732	132 900		
19	335 116 620	250 336		
20	897 697 164	469 536		

Table 6: Exact enumerations of SAWs for 2d and 3d lattices, the results go exactly conform with the literature

3.5.3 Structural Elements in Random SAWs

Using our secondary structure parser we looked at random SAWs and those SAWs after an adaptive walk (chain-length 70; cost function = minimum gyration described in a following chapter).

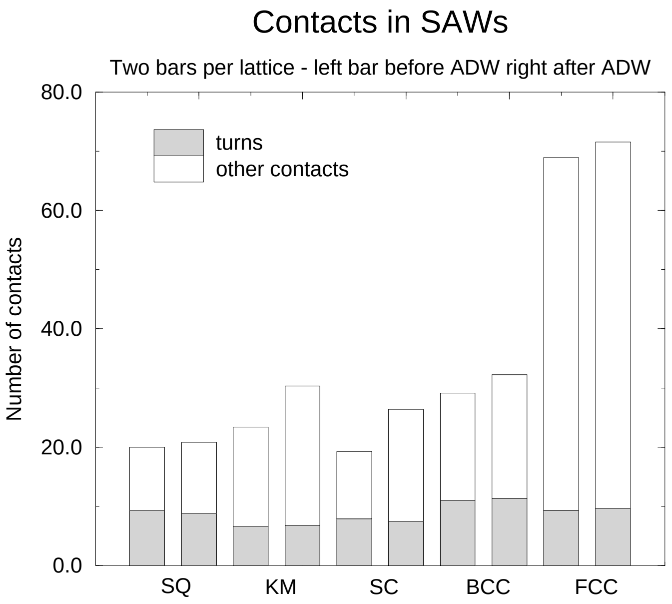


Figure 26: total number of contacts.

The values in figure 3.5.3 represent the amount of point positions that are part of a secondary structure. For example: 1 helices → there is on average one helical structure in each secondary structure. You can see, that the more complex lattices loose secondary structure elements, while performing an adaptive walk. This tendency is not being observed within the SQ lattice and the SC lattice as. The less complex lattices as SQ and SC follow Dills

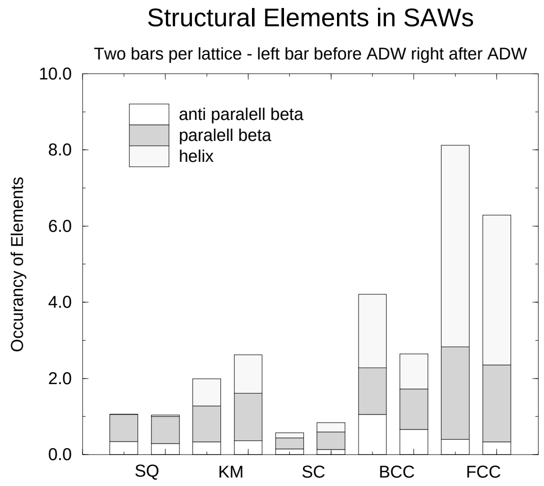


Figure 27: occurancy of secondary structure elements ©

[6] theory, in which he finds exactly this phenomena. The more complex lattices as you can see in the histogram loose secondary structure elements here there is a contradiction to Dill, which will be studied in the near future. The increase of the number of contacts is a logical conclusion for the ADW.

4 SAWs as a Model for Heteropolymers

4.1 Hetero Polymers suitable for Folding ?

Investigating the folding and the structures of Lattice Polymers serves well as a simplified model, that can easily be handled by computational means. Lattice models like the one introduced in this work, basicly derived from Dill's model, are a generalized and simplified representation of Polymers. Those models regard the excluded volume effects, the hydrophobicity as a generic folding potential and structure formation from energy potentials.
Advantages of the model are: easy computability with integers, as there are discrete values for coordinates and angles, simple energy potentials, great clearness for observing and defining structures. The conformational space therefore, is drastically reduced. Compactness and energy states can be exactly defined and visualized, solvent effects can be included implicitly, energy surfaces may be smoothened. Regularities of structural elements, often falling together with the definition of a secondary structure, can be easily defined and identified on regular lattices.
If these models are used for "realistic representations" of polymers, namely proteins, there are, however, numerous shortcomings arising from the crude resolution: The symbol-set is drastically restricted (usually 2 instead of 20); residues are of equal size and mass; different volumes of sidechains are not considered, 1 discrete angle of n*90 (n=1,2,3) on the cubic lattice model instead of 2 dihedral angles with any value in the peptide - bond, bond lengths are restricted to unity, degeneracies of ground state structures are likely. Cross links (= long range interactions, f.i. disulfide bonds) are usually omitted. They restrict the remaining number of possible conformations and thereby the entropy and stabilize the folded state (glutamate/alanine - ester bonds). However an implementation of this constraint is easy to realize. Further problems of previous applications of the LP - concept are: Dill's approach yields the minimum free energy state, which is not necessarily the one that occurs in natural protein folding pathways for any given random sequence. ([8] reports that, for natural sequences on a lattice the best solution is within 2% of all possible conformations that fit in a volume, that is equivalent to the space as restricted by natural observations when using the FCC (face centered cubic lattice) lattice for simulation, lower states are artifacts due to the insufficiencies of the model)

The model is, however, perfect, to investigate the influence of various parameters on structure formation and, due to the computability, to date the only model that can be used for extensive statistical investigations. Furthermore it depends on the rules for the formation of intramolecular bonds, stabilizing the structure, what a model like this serves for. In the case of most RNA folding models, each monomer binds at most to one other and the alphabet is essentially composed of a set of pairwise matching monomers. In case of an energy rules based on Dill's concept, the model is essentially based on the assumption that hydrophobic forces are the major driving force for structure formation. (This point however seems to be undecided, recent simulations on lattices claim the hydrophobic forces to work against the formation of a compact-state).

4.1.1 Hetero Polymers

Nearly all biopolymeres are *heteropolymers*, they are build upon a combination of monomers out of a given number of classes κ.
You can build out of κ classes

$$N_\kappa(\nu) = \kappa^\nu \tag{32}$$

different polymers with the length ν.
Even in a binary alphabet ($\kappa = 2$) the amount of different protein-molecules is outrages and exceeds soon Avogadro number. We use a binary alphabet for our purposes, in which the different types off amino acids are classified as hydrophobic monomers (denoted by H), which are oil-like and interact unfavorably with water, and polar or charged monomers (denoted by P), which interact favorably with water. An example of an H monomer is the amino acid leucine; an example of a P monomer is serine. In native conformations of globular proteins, the H monomers tend to be buried inside the core of the globule, implying that proteins are driven to compactness by the hydrophobic interactions. P monomers tend to reside on the surface of the globule, although exceptions are common.

We have to be aware that our knowledge about biopolymeres, their structure and properties only represent a small notch of the possibilities in vivo. We can only get a closer picture, if some properties would be redundant.
Therefore it is of extreme importance to get an order into the tremendous

amount of different biopolymeres, by introducing measures of relatedness: *sequence-space or shape-space*[37].

4.1.2 The Folding Problem

Most interest is focused on determining a stable minimum, preferable the global minimum which in general is supposed to fall together with the native state.
Even a simplified model like this has been proven to be NP complete very recently [20, 44]. This fact simply states, that the computational effort for a method to find the global minimum in any case increases exponentially with the size of the problem.
NP completeness however only denotes a worst - case behavior and, depending on the problem it might well be that a useful solution, possibly even the global minimum can be found within reasonable computational time. (The application of branch and bound algorithms to a folding pathway based procedure might prove helpful.)
Very often however the goal is to understand *how* the folding process works. Levinthal's paradox is one off the most striking finding of molecular dynamics properties: to find the lowest energy state of a protein, which is usually considered to be the native state, it would take a simple protein astronomic time scales by simple trial and error of all possible configurations of all single bindings [33]. Actually Proteins do not fold by a random search. In fact the ground state is found within a relatively short time, i.e. seconds or less.
Levinthal proposed the existence of a directed folding pathway. Several assumptions to circumvent the difficulties expressed by the formulation of Levinthal's paradox have been formulated recently [48]. Zwanzig gave a reasonable explanation very shortly, showing how a small bias can have a dramatic influence on search speed [48]. A practical method for simulation is to reduce the conformational space by excluding all but certain directions for bonds by a reduction of the space to lattice representation.
An ansatz like this could offer a way out of this dilemma and, although we do not present a "realistic model" of the folding process (association and dissociation constants are not considered), we can simulate the formation of a compact state, close to the global minimum, that could possibly correspond to the so called Molten Globe state.

The primary reason for our simulations however is, to gain insight into the sequence space / shape space relation shown with the example of simplified polymer models. For this reason it is essential to have a folding algorithm that circumvents the pitfalls of the NP completeness in finding the global minimum and enables extensive statistical investigations.

4.1.3 Energy Functions

Dependency of potentials on bond angles, lengths, electrostatic etc. are neglected. A general form of the energy function can then be formulated as follows ([28][41][4]) :

$$E(A, \mathcal{A}) = \sum_{i}^{n} \sum_{j>i+1}^{n} U(s_i, s_j); |i-j|; d_{ij}) \tag{33}$$

so that contributions are considered for topological neighbors only. E is the listing of pairing energies listed in a matrix, depending on the nature of the residues s_i, s_j only. f is a bond-dependent function, as different types of bond may have different distance dependency.
In particular, Dill's potential:

$$U(s_i, s_j); |i-j|; d_{ij}) = \epsilon[s_i, s_j]\delta(d_{ij}, 1) \tag{34}$$

considered the simplest case only for c instead of DM entries d and only one pairing force, stabilizing the overall structure. In our implementation contributions can be considered up to a certain cutoff distance: $U = 0$ if $d_{ij} > cutoff$.
An interesting approach is Crippen's empiric potential considering 4 different classes of amino acids and distances along the chain [9].

4.1.4 Structures

Great attention has to be paid to the characterization of structures and distances inbetween [3].
In a very general model like this one, the analogy of secondary structure notion to real proteins should not be overestimated. Each occupied bead represents a whole residue, different sizes of side chains and sterical hindrance are not considered. From that point of view, a chain of, say 10 amino acids

length could possibly correspond to a real sequence that is much longer. One should therefore focus rather on the essence of a secondary structure: the order in space, basicly recognizable as repitivity. For that reason we can also investigate the occurance of any repitititive unit within an overall shape. It is therefore not so important to find structural elements that are a geometric analogy, but to distinguish between regularities that are intrinsic properties of the model.
As soon we are able to characterize the comparison and distances measures between structures we are able to apply the methods to explore the RNA Shape Space and to investigate the mapping process from Sequence to Shape Space as this has been done very recently on the Shape Space of RNA secondary Structures [19, 40, 29]. There the mapping turned out to be of major importance compared to the nature of the folding process. Several features like correlation lengths and the existence of neutral paths appeared to be robust, thereby kinetic folding to yield a similar result as with other algos [43].
It is however not always necessary to operate with folded states. A Density Of States algorithm (DOS) algo of Stolorz [42] f.i. approximates the distribution of low energy states without weighting them, accuracy however again increases computational effort exponentially.

4.2 A First Glimpse on Landscapes

The notion of a landscape was introduced in the early thirties by Sewall Wright [46] in order to describe evolution as an adaptive walk on a fitness landscape. Nowadays landscapes appear in so different fields as in the physics of spin glasses, in the computer science of problems of combinatorial complexity, in evolution, in neural networks, in gene regulatory networks, in the maturation of immune response and in the biophysics of macromolecules.

A geographical landscape is described by the height h over all vectors $\vec{x} = (x, y)$ lying in the XY-plane. Here a landscape more generally stands for a scalar function $F(\vec{x})$, which assigns to all points $\vec{x} = (x_1, x_2, \ldots, x_n)$ of a n-dimensional space a real value. The total of all $\vec{x}$ is called the configuration space. In order to describe a landscape by its statistical properties, we need a metric in configuration space. In a geographical landscape the configuration space is two-dimensional and the components x, y of the vector $\vec{x}$ are continuous. The natural metric is the Euclidean metric. We do not restrict the components of $\vec{x}$ to be continuous, in fact in all considered examples the x_i are discrete.

We can define a landscape as a triple (X, d, f) where X is a finite set, $d : X \times X \rightarrow R_0^+$ is a metric and $f : X \rightarrow V$ is a function, which maps into a vector space with scalar product. This definition restricts the configuration space X to be discrete, because X has to be finite. For all landscapes discussed here, the vector space V is R . f is often called the cost function. This expression originates from the study of fitness landscapes, where f evaluates the fitness of a species. Landscapes have to be seen in the context of an optimization process, which maximizes or minimizes $F(\vec{x})$.

Landscapes in our context are mappings from the space of genotypes (=sequences) into a space of real numbers that are assigned to some phenotypic (=structural) features, e.g. the minimum free energy, rate constants of structure formation or an arbitrarily chosen fitness value.
A suitable method to characterize landscapes is the autocorrelation function

$$\rho(h) = 1 - \frac{\langle D^2(f(x), f(y)) \rangle_{h(x,y)=h}}{\langle D^2(f(x), f(y)) \rangle_{random}} \tag{35}$$

where for example

$$D(f(x), f(y)) = \Delta G(x) - \Delta G(y)$$

is the difference in free energies for two sequences x and y. As analytical solutions are not available for most landscapes, we use large statistical ensembles of computationally folded RNA molecules to compute $\rho(h)$. This expression can be viewed as a measure of the average similarity of energies or structures etc. as a function of the Hamming distance h of the underlying sequences.
A useful measure for the ruggedness of a landscape is the correlation length $l = \rho(h) = 1/e$. It is in the order of the average distance between two local optima and therefore characteristical for the complexity of an optimization problem. In the case of RNA we found energy correlation lengths that are longer than structure correlation lengths, both are very short compared to the diameter of the sequence space [19, 18]. A generalization of the theory of landscapes allows to directly address the properties of the folding: Folding can also be viewed as a mapping from one abstract metric space of combinatorial complexity, the sequence space to another metric space, the shape space: a *Complex Combinatory Map (CCM)*. This will work as we can define

a metric distance D to compare structures.
Instead of dealing with CMMs explicitly we can also use large ensembles of random structures (structures, folded from random sequences). We compute structure distances as well as the corresponding Hamming distances of their underlying sequences. We then generate *structure density surfaces*, expressing the joint probability for two sequences of constant chain length of having a certain Hamming distance and a certain distance between their structures.

4.2.1 Adaptive Walks

Additional information about the structure of a landscape can be attained by per forming uphill (or downhill) walks on the landscape. These walks are restricted to increasing (or decreasing) values of the cost function and always end in local optima, so they provide information on the distribution of local optima. One of them is the ADaptive Walk (ADW):
Adaptive walks start at a randomly chosen configuration, choose one neighboring configuration are random and accept it, if it has a better value of the cost function. As long as a better solution exists, it is adopted and the neighborhood search is repeated from the new solution. The walk stops if no neighbor has to offer a better value of the cost function, which means that a local optimum is reached.

We performed several ADWs of lattice polymers. We used the number of contacts and in addition the radius of gyration as cost functions.

4.2.2 ADW Using Pivot Moves

Following tables are using the pivot algorithm. In table 7 we see a summary of results, using the radius of gyration as cost function. Table 8 shows data for ADWs with respect to the number of contacts as cost function. You clearly can recognize that there is an expected relation between increasing number of contacts and decreasing radius of gyration.

4.2.3 ADW Using Snake Movement

Table 9 represents data of ADWs using a snake movement to get from one state to another. Here we are using number of contacts as cost function again. It is remarkable how little effect the snake movement has on a walk.

SAWs: adaptive walks with pivot moves towards minimum radius of gyration											
Dim.	Grid	length	adapt. walk		no. of contacts		radius of gyration		end to end distance		sample
			local optima	walk length	random	minimum	random	minimum	random	minimum	
2	HEX	30	0.154	2.83	4.28	4.73	11.88	5.11	10.9	5.8	1000
		70	0.154	3.83	9.75	9.57	36.76	16.39	18.8	10.1	1000
	SQ	30	0.066	5.2	10.15	11.63	6.4	2.86	8.4	3.6	1000
		70	0.036	8.45	24.41	24.64	20.88	8.71	15.4	5.4	711
	TRI	30	0.066	6.37	23.78	27.08	8.34	2.78	9.0	3.9	1000
		70	0.051	9.22	57.98	59.44	25.98	8.51	15.8	6.2	1000
	KM	30	0.886	1.19	10.0	22.0	18.5	4.8	13.0	3.9	1000
		70	0.269	12.3	31.4	39.9	42.8	16.7	19.3	7.9	1000
		100	0.326	13.8	48.5	57.0	66.6	27.6	24.6	10.3	129
3	SC	30	0.002	13.76	8.83	16.07	3.66	0.80	6.8	2.4	1000
		70	0.002	22.88	22.83	30.97	9.19	2.41	10.4	3.2	839
		100	0.027	26.7	34.3	40.8	14.1	2.81	3.29	0.91	75
	BCC	30	0.095	10.88	16.43	21.93	3.06	0.85	6.1	2.5	1000
		70	0.241	13.95	47.0	46.2	7.5	2.6	9.1	4.2	1000
		100	0.28	15.29	70.6	64.8	11.4	4.2	11.2	5.2	514
	FCC	30	0.204	10.37	35.5	42.1	11.8	4.0	11.4	5.2	1000
		70	0.378	12.35	100.97	95.91	27.1	11.8	17.2	9.3	230
		100	0.404	14.38	152.45	140.23	41.69	17.49	20.6	10.7	156
	TET	30	0.187	5.41	9.6	10.3	4.1	1.5	6.7	3.6	1000
		70	0.36	5.84	30.0	25.9	9.7	4.9	10.2	6.4	1000
		100	0.395	6.3	45.9	37.6	14.6	7.9	12.4	8.0	1000

Table 7: ADW with pivot moves towards minimum radius of gyration ©

SAWs: adaptive walks with pivot moves towards maximum number of contacts											
Dim.	Grid	length	adapt. walk		no. of contacts		radius of gyration		end to end distance		sample
			local optima	walk length	random	minimum	random	minimum	random	minimum	
2	HEX	30	0.28	1.8	4.2	6.5	8.4	5.0	9.6	6.1	500
		70	0.25	2.9	9.7	13.8	26.1	15.4	16.9	10.8	500
		100	0.27	3.5	13.7	18.9	43.9	26.4	18.9	13.7	500
	SQ	30	0.25	3.3	9.7	15.0	6.9	3.5	8.8	4.9	500
		70	0.21	5.6	24.5	34.5	21.0	11.1	15.2	8.5	500
		100	0.23	6.7	35.3	47.5	33.6	18.1	19.2	10.7	500
	TRI	30	0.22	5.1	23.5	34.3	5.8	3.0	8.0	4.3	500
		70	0.22	8.8	59.0	79.2	18.7	9.9	14.0	7.7	500
		100	0.23	11.4	84.7	112.1	30.5	14.9	17.9	9.2	500
	KM	30	0.08	7.0	10.0	26.5	18.7	6.8	12.8	5.1	500
		70	0.28	7.2	29.8	51.0	47.1	22.7	10.4	8.3	500
		100	0.37	6.7	47.1	71.0	66.9	37.4	24.2	14.7	500
3	SC	30	0.07	7.3	8.7	21.4	3.7	1.2	6.8	2.8	500
		70	0.08	11.8	23.9	49.1	8.1	3.0	10.2	4.2	500
		100	0.07	15.2	34.5	68.7	13.6	4.0	12.8	4.6	500
	BCC	30	0.30	5.1	16.8	28.4	3.0	1.3	6.0	3.3	500
		70	0.55	4.7	47.5	61.5	7.4	4.9	9.0	6.4	500
		100	0.59	5.3	70.5	86.5	11.2	7.4	11.3	8.1	500
	FCC	30	0.41	5.1	24.8	51.0	11.7	6.4	11.2	7.0	500
		70	0.67	4.0	98.9	116.1	30.2	21.2	18.4	13.7	500
		100	0.69	4.3	149.6	168.9	41.1	30.9	20.9	16.7	500
	TET	30	0.43	2.0	9.6	13.9	4.1	2.8	6.8	5.2	500
		70	0.59	1.9	30.1	35.8	10.4	8.4	10.6	9.0	500
		100	0.71	1.3	44.1	49.1	14.8	13.3	12.2	11.3	500

Table 8: ADW with pivot moves towards maximum number of contacts ©

SAWs: adaptive walks with snake moves towards maximum number of contacts												
Dim.	Grid	length	No of neutral moves	adapt. walk		no. of contacts		radius of gyration		end to end distance		sample
				local optima	walk length	random	minimum	random	minimum	random	minimum	
2	HEX	30	1		0.81	3.5	3.6	8.0	7.9	9.5	9.2	500
		30	5		2.98	3.3	3.9	8.4	8.0	9.5	9.1	500
		30	10		4.36	3.6	4.5	8.0	7.4	9.2	8.4	500
		30	100		5.34	3.3	4.3	8.2	7.5	9.5	8.4	500
		70	1		0.79	8.5	8.7	26.0	25.9	16.8	16.8	500
		70	5		3.06	8.5	9.1	26.0	25.7	16.8	16.3	500
		70	10		4.01	8.5	9.4	25.8	25.2	16.3	15.8	500
		100	1		0.76	12.6	12.7	42.7	42.7	21.5	21.4	500
		100	5		2.84	12.7	13.3	42.8	42.7	21.3	21.1	500
		100	10		4.1	12.8	13.7	43.3	42.1	21.4	21.0	500
	SQ	30	1		0.84	8.9	9.2	6.8	6.7	8.8	8.6	500
		30	5		2.87	8.8	9.8	6.8	6.4	8.9	8.3	500
		30	10		3.89	8.8	10.2	6.7	6.1	8.7	8.0	500
		30	100		5.08	8.7	10.5	6.8	6.1	8.9	7.8	500
		70	1		0.87	23.0	23.3	21.7	21.6	15.2	15.1	500
		70	5		2.61	23.0	24.0	21.8	21.6	15.3	15.1	500
		70	10		3.67	22.9	24.4	21.7	21.3	15.4	14.9	500
		100	1		0.88	34.8	35.2	33.8	33.7	19.2	19.1	500
		100	5		2.60	34.7	35.8	34.2	33.7	19.2	18.8	500
		100	10		3.10	34.7	36.0	34.2	33.6	19.3	18.9	500
	TRI	30	1		1.32	23.0	24.1	5.9	5.8	8.0	7.8	500
		30	5		3.97	23.0	25.9	6.0	5.3	8.0	7.3	500
		30	10		5.57	22.3	26.4	5.9	5.3	8.1	7.1	500
		30	100		6.19	22.6	27.1	6.1	5.1	8.2	6.8	500
		70	1		1.22	58.6	59.8	18.4	18.3	13.9	13.8	500
		70	5		3.63	58.5	61.6	18.6	18.2	13.9	13.4	500
		70	10		4.79	58.2	62.3	18.8	18.3	14.0	13.5	500
		100	1		1.23	85.3	86.4	29.0	28.9	17.7	17.6	500
		100	5		3.88	85.2	88.6	29.3	28.6	17.8	17.3	500
		100	10		4.46	85.3	89.2	29.4	28.8	17.6	17.1	500
	KM	30	1		0.61	9.1	9.6	18.8	18.5	12.9	12.7	500
		30	5		0.79	9.3	10.0	17.6	16.9	12.7	12.2	500
		30	10		0.90	9.4	10.2	18.7	17.7	13.3	12.7	500
		30	100		0.90	9.4	10.2	18.7	17.7	13.3	12.7	500
		70	1		0.25	30.6	30.7	46.8	46.7	20.5	20.4	500
		70	5		0.38	29.8	30.0	45.1	44.6	20.2	19.9	500
		100	1		0.23	47.9	48.0	63.2	63.1	23.2	23.1	500
		100	5		0.25	48.5	48.6	64.1	64.2	23.6	23.6	500

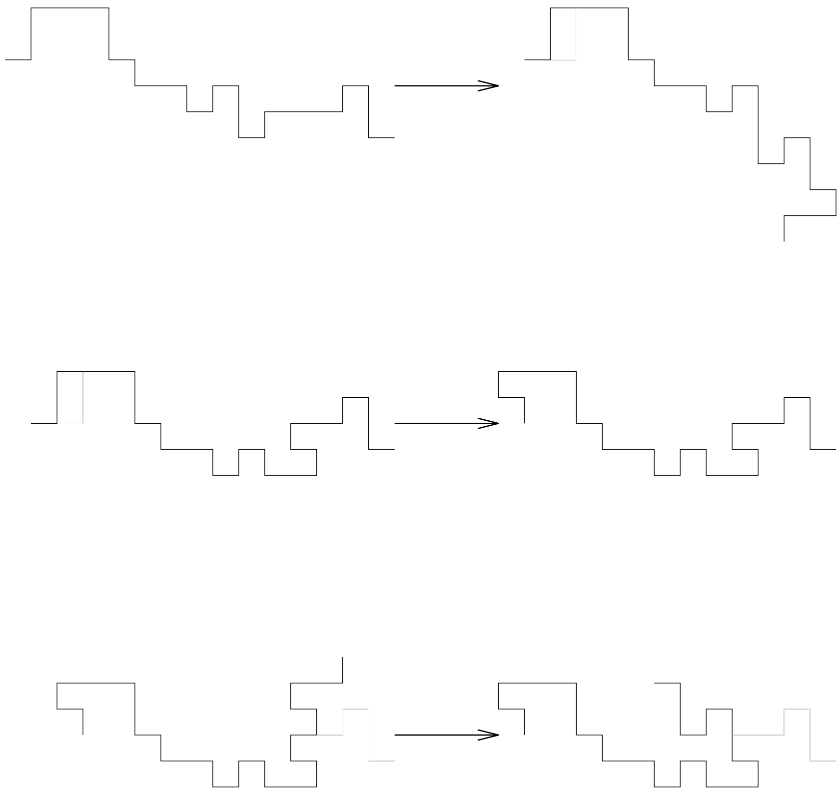
Table 9: ADW with snake movement for 2D lattices ©

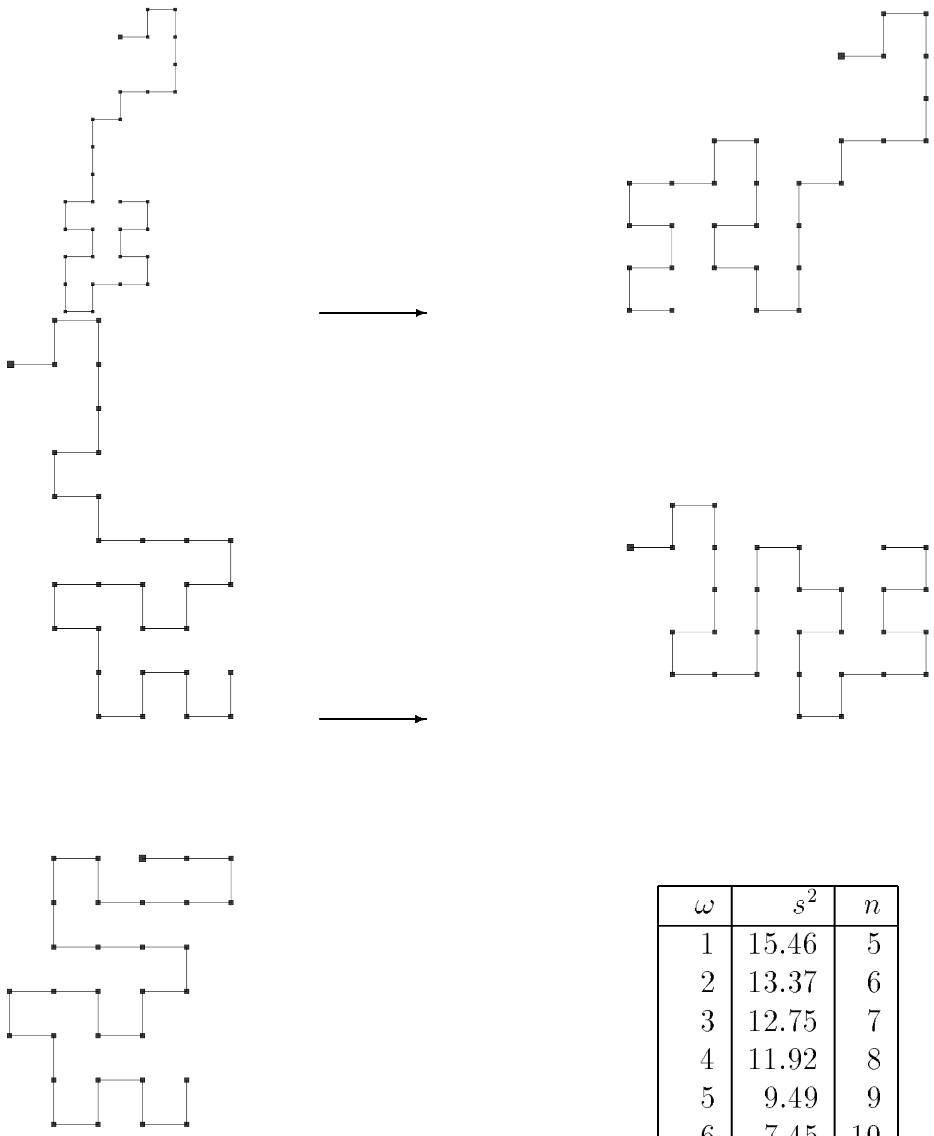
SAWs: adaptive walks with snake moves towards maximum number of contacts												
Dim.	Grid	length	No of neutral moves	adapt. walk		no. of contacts		radius of gyration		end to end distance		sample
				local optima	walk length	random	minimum	random	minimum	random	minimum	
3	SC	30	100		10.02	7.9	10.9	3.7	3.0	6.7	5.8	500
		30	1000		10.02	7.9	10.9	3.7	3.0	6.7	5.8	500
		70	1		1.08	22.3	22.7	8.8	8.7	10.1	10.1	500
		70	5		4.37	22.4	24.0	8.7	8.6	10.1	9.9	500
		70	10		6.76	21.8	24.2	9.5	9.0	10.6	10.1	500
		100	1		1.15	33.1	33.7	13.8	13.7	12.5	12.4	500
		100	5		4.48	33.0	34.7	13.4	13.2	12.3	12.2	500
		100	10		6.76	32.8	35.3	13.1	12.9	12.4	12.1	500
	BCC	30	100		0.95	15.2	16.2	3.0	2.9	5.9	5.7	500
		70	1		0.32	46.1	46.3	7.4	7.5	9.1	9.1	500
		70	10		0.38	44.6	45.0	7.8	7.8	9.5	9.5	500
		100	1		0.26	71.9	72.1	10.4	10.4	10.9	10.9	500
		100	5		0.26	69.8	70.0	10.7	10.7	11.0	11.0	500
		100	10		0.26	69.8	70.0	10.7	10.7	11.0	11.0	500
	FCC	30	100		0.4	34.1	34.7	11.3	11.2	11.9	11.8	500
		70	10		0.16	99.4	99.5	29.4	29.5	18.0	18.0	500
		100	1		0.12	150.0	150.1	40.5	40.5	20.9	20.9	500
		100	5		0.14	148.9	149.1	41.1	41.0	21.0	20.9	500
		100	10		0.14	148.9	149.1	41.1	41.0	21.0	20.9	500
	TET	30	1		0.2	9.0	9.6	4.0	3.8	6.8	6.7	500
		30	5		0.2	9.5	10.1	3.7	3.4	6.4	6.2	500
		30	10		0.2	9.5	10.1	3.7	3.4	6.4	6.2	500
		70	1		0.04	29.5	29.7	9.7	9.6	10.3	10.3	500
		70	5		0.05	29.7	30.0	9.7	9.6	10.2	10.1	500
		70	10		0.05	29.7	30.0	9.7	9.6	10.2	10.1	500
		100	1		0.04	45.4	45.7	14.1	14.0	12.3	12.3	500
		100	10		0.04	45.4	45.8	14.1	14.0	12.3	12.3	500

Table 10: ADW with snake movement for 3D lattices ©

4.2.4 Selected Graphs of ADWs

The following graphs show an ADW towards maximum number of contacts (pivot moves). The table shows the walklength ω, number of contacts n and radius of gyration s^2 of the depicted ADW:





ω	s^2	n
1	15.46	5
2	13.37	6
3	12.75	7
4	11.92	8
5	9.49	9
6	7.45	10
7	7.32	12
8	4.71	13
9	4.42	14
10	2.35	17
11	2.20	19

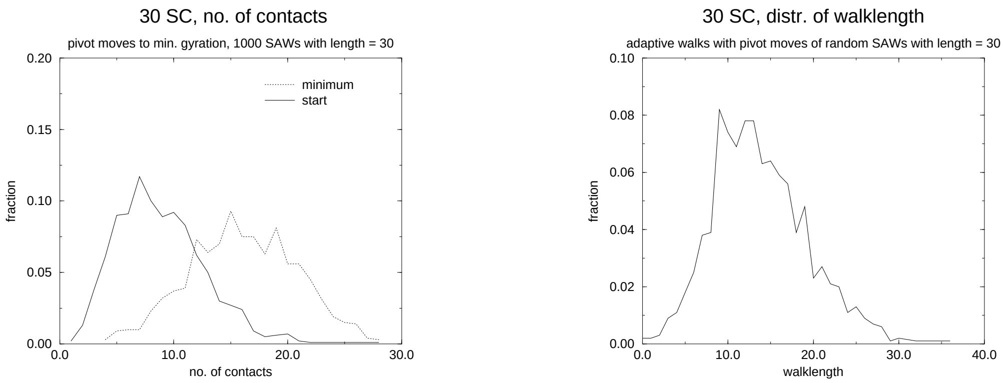


Figure 28: ADW towards min radius of gyration SC 30

In this figures you clearly can see, that the number of contacts increase during an ADW. The walklength is quite Gaussian distributed and the value for it is approximately about 12.

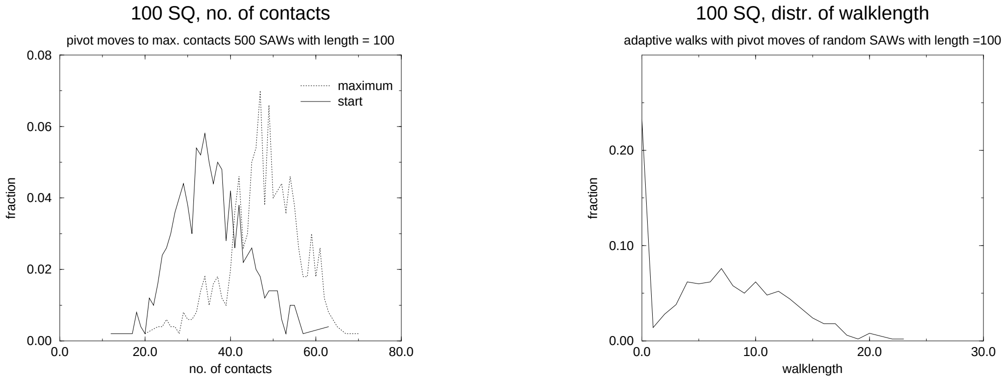


Figure 29: ADW towards max contacts SQ 100

Here it is also clear that the number of contacts rise during and ADW. The walklength is also quite even distributed.

5 Conclusion and Outlook

5.1 Conclusion

In the present work we achieved the following data:

1. The length ℓ of a gSAW is calculated and we yield an interesting propertiy of SAWs.

2. The shape of the distribution function of ℓ is strong dependent of the connectivity within 3D lattices. 2D lattices don't vary a lot. Only the KM lattice shows a different form, this is due to the fact, that it is some sort of a 2-3D lattice.

3. The minimum length ℓ_{min} of gSAWs has been computed by exhaustive search, only for the FCC lattice it was impossible and we have to be content with upper bounds on ℓ_{min}.

4. The distribution of relative-directions within gSAWs and SAWs after a certain amount of pivot moves show, that they do not occur with equal frequencies in SAWs. There is a strong bias towards "forward" steps. This effect is not surprising, since "forward" steps yield in unrestricted choices for the next move.

5. Structural elements during an ADW were searched and it is obvious, that more complex lattices loose secondary structure elements, while performing an adaptive walk. This tendency is not being observed within the 2D lattices.

6. Simple ADWs were calculated using pivot moves, there is an expected relation between increasing number of contacts and decreasing radius of gyration. Snake movements didn't yield the promised advantage to get to higher values of contacts.

7. Exhaustive search for gSAWs have been performed, the obtained data confirmed that the number of SAWs strongly depend on the dimension and on the moveset of the lattice.

5.2 Outlook

We developed a toolkit for lattice polymers, to determine properties of SAWs. After having calculated a lot of properties we would like to use this approach to model foldings of polymers. First steps were taken and we are going to continue into this direction.

A Program Description

A.1 Summary

The toolkit is designed to compute and compare structures of Lattice Polymers (LPs). The package works independendently from the chosen lattice and can be easily extended. Various methods to characterize and compare structures as well as to detect secondary structure elements in the sense of Dill's LP analogy to natural proteins are introduced [5]. At this point I would like to thank Erich Bornberg-Bauer and Peter F. Stadler especially for their programming.

A.2 Working on Lattices

A.2.1 Lattice Definitions and Handling of SAWs

header **lattice_types.h**

Contains the definition of the structure to be used for lattice - indepentend SAWs (examples for SQ in parenthesis):

```
struct Lattice {
 int     dim;      /* dimension of lattice                      (2) */
 int     nn;       /* the maximum number of neighbours          (4) */
 int     nm;       /* the maximum number of possible moves  (4) */
 char   *names;    /* move - names of absolute moves     (BFLR) */
 char   *relnames; /* move - names of relative moves      (FLR) */
 int   **Move;     /* possible moves in lattice coord.          */
 int   **Neighbor; /* possible neighbours, coordinates          */
 int   **Table;    /* transformation table rel./abs. moves      */
 float **lv;       /* Lattice Vectors                           */
 int     turns;    /* minimum steps to nearest neighbour    (3) */
 int     steps;    /* step                                  (1) */
 float   wlen;     /* average wlen, before self-traping  (71.7) */
};
```

module **lattice_types.c**

Global Definitions:

- `char Lattice_type[6]`

To choose one out of the predefined lattices, assign the name of desired lattice to this variable. Possible choices must coincide with one of the possibilities as listed in `struct Lattice` .

- `struct Lattice L`

The global structure, carrying all necessary variables during computation. the proper values must be assigned by using the procedure `define_lattices (...)` .

- `PRIVATE struct Lattice TRI, HEX, SQ, TET, SC, BCC, FCC, KM and TDKM`

The structures of type `struct Lattice` that can hold the explicit values for all variables for the available lattices TRIgonal, HEXagonal, SQare, TETraedic, Simple Cubic, Face Centered Cubic, Body Centered Cubic, Knight Moves and Three Dimensional Knight Moves respectively. This is the only data type to be extended for incorporating new lattices into the package. Some (larger) arrays with fixed size and static definitions such as movesets and Neighbors have to be introduced in the header of the file.

Function Calls:

- `void define_lattices(char *Lattice_type)`

Has to be set for initializing the proper parameter set before using any other procedures of the package. First all non-static variables for the chosen lattice are assigned, then the contents of the structure are copied to the global

- `struct Lattice L` which is then the only lattice structure to be used from there. This is the only procedure to be altered for incorporating new lattices into the package, valid choices are those specified at `struct Lattice`.

- `void free_lattices()`

Frees the space in memory after the usage of the package.

module **SAW_utils.c**:

Variables:

- `pindx` A global array, containing an index-field to address matrices.

Function Calls:

- `int isSAW(char *structure)`

Expects a structure in rel.moves notation and returns 0 if structure is no SAW according to the specified lattice, a 1 otherwise.

- `int isSAW_last(char *structure, int **x, int i)`

Same as isSAW(), but checks correctness of the last move only. x must already contain the coordinates in positive integers.

- `void pivotSAW(char* structure)`

Applies a random pivot move to structure, returns only SAW - structures. A pivot move can always be found due to the ergodicity.

- `void pivot(char* structure)`

makes a random pivot move to structure, without respect if the structure is a SAW afterwards.

- `char *trap_SAW(int len,int maxlen, double *end_dist)`

Returns a SAW that continues to grow until it was trapped or `maxlen` reached. `len` is required to initialize fields and should be of higher value then maxlen. This function also calculates the end to end distance and returns it as a pointer(`end_dist`).

- `int isSAW_last_old(char *structure)`

Same as is `SAW_last()`, requires no x values however

- `int isSAW_avl(char *structure)`

Same as isSAW() but with usage of AVL tree.

- `int isSAW_last_avl(char *struc,int **x,int pos,int last)`

Same as `isSAW_last()` but with usage of an AVL tree package.

- `void init_indx(int length)`

Initializes the indexfield `int* pindx` for usage in Distance and Contact Matrix for a structure of length `len`. The exact adressing works as follows: its a 2 dimensional matrix index normally you would address a 2 dim matrix XX[1][2] with pindx: XX[pindx[1]+2].

- `char *Contact_Matrix (char *structure)`

Returns a char containing the Contact Matrix, using pindx. Each residue with a contact, according to the proper criterion, is denoted with a `1`, each non-contact residue with a `0`.

- `int CM_counts (char *CM)`

Returns the number of contacts from the Contact Matrix CM.

- `float *Distance_Matrix (char *structure)`

Expects the structure according to the current lattice settings in relative moves notation and returns an array containing the triangular matrix with all pair distances. Values can be accessed as usual with the `pindx` index field.

- `float gyr_radius(float *DM,int len)`

Returns the radius of gyration from a Distance Matrix that has been generated from a structure with length `len`.

A.2.2 Handling of Data Structures

For handling complex tasks we programmed a hash utility cause of memory and time savings. So just use them for big stuff.

module **SAW_hash.c**:

- `int hash_comp(void *x, void *y)`

Compares two specific sets of coordinates returns 0 if they match.

- `int isSAW_hash(char* structure)`

Expects a structure in relative moves notation and returns 0 if structure is a SAW according to the specified lattice.

- `int isSAW_last_hash(char *struc,int **x,int pos,int last)`

Same as `isSAW_hash()` but checks correctness of the last move only. x must already contain the coordinates in positive integers. pos must be the current position and last is a criteria for the maximal length, it should always be bigger than pos. If last is exceeded the function returns a 2.

- `char *trap_SAW_hash(int len,int maxlen, double *end_dist)`

Same as `trap_SAW()`, just using hash-functions.

- `void pivotSAW_hash(char* structure)`

Applies a random pivot move to structure with use of hash-functions. Returns only SAWs.

module **hash_util.c**

- `int hash_f (int *x);`

Here the hash-function is defined. In our case its a function of the following type: $f(x, y, z, \ldots) = ax * by * cz \ldots$. In which the constants $a, b, c, \ldots$ are primes.

- `void  define_paras();`

before using any hash-functions you have to make this function call. Here the constants of the hash-functions are defined.

- `int lookup_hash (void *x);`

This function evaluates whether the int pointer x is already part of the hash table (returns 1) or not (returns 0).

- `int write_hash (void *x);`

Here you can make entrances into the hash-table, if it the function returns a 0, otherwise a 1.

- `void delete_hash (void *x);`

Deletes a specific entry out of the hash-table.

- `void kill_hash();`

Releases memory uptaken by hash-functions. Should be used as the last function, to avoid memory leakage.

- `void initialize_hash();`

There is no purpose yet, but it could be used for new functions and parameters.

For medium problems (talking about length approx: 60-10000) the use of AVL trees is convenient and fast:
module **avl.c**
Here are all the functions you need for the use of an avl tree.
header **avl.h**
Different function parameters.
module **avlaccess.c**
Specific functions for the use of avl-trees within our SAW package.
header **SAW_avl.h**
Definition of parameters.

A.2.3 Handling and Comparison of Structures

Structures, as generated by SAWs or folding algorithms are strings of length `strlen(sequence)-1` in the notion of relational moves. They can be converted into Contact Matrices `CM`, Bond Matrices `BM` and Distance Matrices `DM`. `BM`s and `CM`s can be converted into Contact Lists `CL`s and `BM`s into trees. 2 matrices, representing 2 (different) structures in the same notion can be compared and distances between them measured. A detailed description of the structure distance measures is given in [3].
Further measures with respect to the nature of neighboring residues will be provided.

For the comparison of CLs we need a table, showing the edit costs for each edit operation. Given a set of edit operations and edit costs, the edit distance is given by the minimum sum of costs along an edit path converting one structure into the other.
header *CL_cost.h*:
Provides the edit table for the comparison of two CLs.

module **SecStr.c**
Here structures can be analyzed to find number and size of secondary structure elements as defined in [?]. Values are stored in a structure `SecStr` and can be retrieved from there.

- `SecStr *parse_CM(char *ContactMatrix)`

Takes a Contact Matrix `CM` and converts it from a `0/1` notion for no contacts /contacts into a `H/B/P/T/C/0` (for corresponding residues being member of a Helix, antiparallelBetasheet, Parallelbetasheet, Turn, otherContact, n0contact respectively) notion.
header **SecStr.h**

```
typedef struct  blah{
  int  nt;       /* number of turns                            */
  int  npb;      /* number of parallel beta sheets             */
  int  nab;      /* number of antiparallel beta sheets         */
  int  nh;       /* number of helices                          */
  int  others;   /* number of all other contacts               */
  int  *turns;   /* list of turns by first base                */
  int  *pbeta_i; /* begin1 of parallel beta sheets             */
  int  *pbeta_j; /* begin2 of parallel beta sheets             */
  int  *pbeta_l; /* lenght of parallel beta sheets             */
  int  *abeta_i; /* begin1 of anti-parallel beta sheets        */
  int  *abeta_j; /* begin2 of anti-parallel beta sheets        */
  int  *abeta_l; /* length of anti-parallel beta sheets        */
  int  *helix_b; /* begin of helix                             */
  int  *helix_e; /* end of helix                               */
  int  *helix_m; /* winding height of helix                    */
}SecStr;
```

- `int ident_contacts (char *CM1, char *CM2)`

Counts the number of identical contacts (= nearest neighbors) from the two Contact Matrices `CM1` and `CM2` that must have been generated from two structures of same length before.

- `int **CM_to_CL(char *CM)`

Transfers a Contact Matrix `CM` into a two dimensional array, the Contact List `CL` which is returned, each residue a list of contacts is assigned to. It needs the Contact Matrix `char *CM` as input, and returns a matrix, which has to be freed afterwards. The `pindx` field must be already initialized.

A.2.4 Graphik - Utilities

module **SAW_graf_utils.c**
Provides several methods to illustrate structures in postscript or `pdb` format.

- `void SAW_PS(char *sequence, char *structure, char *fname, char *ALPHAbet);`

Generates a file named `fname` in postscript format, depicting the structure with balls representing each residue in the node point and a stick for each bond. Lattice and Alphabet must have been specified before usage.

- `void pdb_out (char *struc, char *seq, char *pdb_fn, char *alphabet, char *para_fn)`

Generates a file `pdb_fn`. A different colour is used for each different secondary structure element.

- `void seq_pdb_out(char *struc, char *seq, char *scont,char *pdb_fn,char *para_fn)`

Same as `pdb_out()` , but uses different colours for different residues in the sequence.

module **SAW_dots.c**

- `void LP_CM_dot(char *CM, char *string, char *ps_fn)`
Needs a Contact Matrix `CM` that has been transformed into a `H/B/A/T/C/O` notion by the use of `parse_CM` and produces a file in postscript format that shows the structure in a coloured dot-plot matrix where each filled square denotes a neighbor as well as a legend, illustrating the symbols for secondary structure elements.

- `void LP_CM_DM_dot(char *CM, char *DM, char *string, char *ps_fn)`
Similar to `LP_CM_dot()`, but shows both, the `CM` and `DM` , where the size of the squares is related the distance of the two corresponding residues.

Figure 30: Sample structure SQ

Figure 31: Sample contact-matrix SQ

B Contact-matrix Samples and More

Figure 32: Sample structure SC

Figure 33: Sample contact-matrix SC

References

[1] R. Baldwin. How does protein folding get started. *TIBS*, pages 291–294, 1989.

[2] A. Beretti and A. Sokal. New monte carlo method for the self-avoiding walk. *J. of Stat. Phys.*, 40:483–531, 1985.

[3] E. G. Bornberg-Bauer. Structures of lattice polymers - phd thesis. *in preparation*, 1995.

[4] G. Casari and M. J. Sippl. Structure-derived hydrophobic potentials — hydrophobic potenti als derived from x-ray structures of globular proteins is able to indentify native folds. *J.Mol.Biol.*, 224:725–732, 1992.

[5] H. S. Chan and K. A. Dill. Intrachain loops in polymers: Effects of excluded volume. *J. Chem. Phys.*, 90:492–509, Jan. 1989.

[6] H. S. Chan and K. A. Dill. The effects of internal constraints on the configurations of chain molecules. *J. Chem. Phys.*, 92,5:3118–3135, Mar. 1990.

[7] H. S. Chan and K. A. Dill. Origins of structure in globular proteins. *Proc. Natl. Acad. Sci. USA*, 87:6388–6392, 1990.

[8] D. G. Covell and R. L. Jernigan. Conformations of folded proteins in restricted spaces. *Biochemistry*, 29:3287–3294, 1990.

[9] G. M. Crippen. Prediction of folding from amino acid sequence over discrete conformation spaces. *Biochemistry*, 30:4232–4237, 1991.

[10] G. P. D. Amit and L. Peliti. Asymptotic behaviour of the treu self-avoiding walk. *Phys. Rev. B*, 27:1635–1645, 1983.

[11] P. de Gennes. Scaling concepts in polymer physics. *Cornell University Press, Ithaca*, 1979.

[12] Enting and Guttmann. Self-avoiding polygons on the square, l and manhatten lattice. *J. Phys. A.*, 18:1007, 1985.

[13] A. J. G. et al. Connective constant of the self-avoiding walk on the triangular lattice. *J. Phys. A*, 19:2591, 1986.

[14] B. M. et al. An extension of the two dimensional self avoiding walk series on the square lattice. *12¡ J. Phys A: Math.*

[15] F. Familiy and M. Daoud. Experimental realization of true self avoiding walks. *Phys. Rev. B*, 29:1506–1507, 1984.

[16] P. Flory. The configuration of a real polymer chain. *Journal of Chemical Physics*, 17:303–310, 1949.

[17] P. Flory. Principles of polymer chemistry. *Cornell Univ. Press, Ithaca, NY*, 1971.

[18] W. Fontana, D. A. M. Konings, P. F. Stadler, and P. Schuster. Statistics of RNA secondary structures. *Biopolymers*, 33:1389–1404, 1993.

[19] W. Fontana, P. F. Stadler, E. G. Bornberg-Bauer, T. Griesmacher, I. L. Hofacker, M. Tacker, P. Tarazona, E. D. Weinberger, and P. Schuster. RNA folding and combinatory landscapes. *Phys. Rev. E*, 47(3):2083 – 2099, March 1993.

[20] A. S. Fraenkel. Complexity of protein folding. *Bull. Math. Biol.*, 55:1199 – 1210, 1993.

[21] A. Godzik, A. Kolinski, and J. Skolick. Lattice representations of globular proteins: How good are they? *J. Comp. Chemistry*, 14:1194 – 1202, 1993.

[22] I. Gradehteyn and I. Ryzhik. Table of integrals, series and products. *Academic Press, New York, 4th edition*, 1980.

[23] A. Guttmann. On the critical behaviour of self-avoiding walks. *J. Phys A: Math*, 22:2807–2813, 1989.

[24] A. Guttmann and J. Wang. The extension of self-avoiding random walk series in two dimensions. *J. Phys. A: Math*, 24:3107–3109, 1991.

[25] A. J. Guttmann. On the critical behaviour of self-avoiding walks. *J. Phys. A*, 20:1839 – 1854, Jul. 1986.

[26] A. J. Guttmann. The high-temperature susceptibility end spin-spin correlation function of the three-dimensional ising model. *J. Phys. A*, 20:1855, 1987.

[27] S. Hemmer and P. Hemmer. An an average self-avoiding walk on the square lattice lasts 71 steps. *J.Chem.Phys*, 81:584–585, 1984.

[28] M. Hendlich, P. Lackner, S. Weitckus, H. es Floeckner, R. Froschauer, K. Gottsbacher, G. Casari, and M. J. Sippl. Identification of native protein folds amongst a large number of incorrect models — the calculation of low energy conformations from potentials of mean force. *J.Mol.Biol.*, 216:167–180, 1990.

[29] M. A. Huynen, D. A. M. Konings, and P. n Hogeweg. Multiple coding and the evolutionary properties of rna secondary structure. *J. theor. Biol.*, 165:251 – 267, 1993.

[30] e. a. K.R. Shoemaker, R. Fairman. The c-peptide helix from nuclease a considered as autonomous folding unit. *Cold Spring Harbor Symp. Quant. Biol.*, LII:391–398, 1987.

[31] R. C. L. Pauling and H. Branson. *Proc. Natl. Acad. Sci. USA*, 37:205,251,729, 1951.

[32] G. Lawler. Intersection of random walks. *Birkhäuser, Boston*, 1991.

[33] C. Levinthal. Are there pathways for protein folding? *J. Chem. Phys.*, 65:44 – 45, 1968.

[34] N. Madras and G. Slade. The self-avoiding walk. 1993.

[35] N. Madras and A. D. Sokal. Nonergodicity of local, length-conserving monte carlo algorithms for the self-avoiding walk. *J. Statist. Phys.*, 47(3/4):573–595, 1987.

[36] N. Madras and A. D. Sokal. The pivot algorithm: A highly efficient monte carlo method for the self-avoiding walk. *J. Statist. Phys.*, 50(1/2):109–186, 1988.

[37] J. Maynard-Smith. Natural selection and the concept of a protein space. *Nature*, 225:563–564, 1970.

[38] G. Ramachandran and V. Sasisekharan. *Adv. Protein Chem.*, 23:283–437, 1968.

[39] D. C. Rapaport. On three-dimensional self-avoiding walks. *J. Phys. A*, 18:113–126, 1985.

[40] P. Schuster, W. Fontana, P. F. Stadler, and I. L. Hofacker. From sequences to shapes and back: A case study in rna secondary structures. *Proc.Roy.Soc.(London)B*, 255:279–284, 1994.

[41] M. J. Sippl. Boltzmann's principle, knowledge-based mean fields and protein folding. an approach to the computational determination of protein structures. *J.Computer-aided molec.design*, 7:473–501, 1993.

[42] P. Stolorz. Recursive approaches to heterogenous systems: Statistical physics of lattice proteins. *preprint*, 1993.

[43] M. Tacker, P. F. Stadler, E. G. Bornberg-Bauer, I. L. Hofacker, and P. Schuster. Robust properties of RNA secondary structure folding algorithms. *Santa Fe Preprint*, 1994. in preparation.

[44] R. Unger and J. Moult. Finding the lowest free energy conformation of a protein is an np-hard problem: Proof and implications. *Bull. Math. Biol.*, 55:1183 – 1198, 1993.

[45] S. Whittington. Statistical mechanics of polymer solutions and polymer adsorption. *Adv. Chem. Phys.*, 1982.

[46] S. Wright. Evolution in mendelian populations. *Genetics*, 16:97–159, 1931.

[47] J. Zinn-Justin and J. L. Guilliou. Critical exponents from field theory. *Phys. Rev. B*, 21:3976–3998, 1980.

[48] R. Zwanzig, A. Szabo, and B. Bagchi. Levinthal's paradox. *Proc. Natl. Acad. Sci. USA*, 89:20–22, 1992.