Exact, Constraint-Based Structure Prediction in Simple Protein Models

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat
der Fakultät für Mathematik und Informatik
der Friedrich-Schiller Universität Jena

von Diplom-Informatiker (Univ.) Sebastian Will
geboren am 13. April 1974 in München

Gutachter

1. Prof. Dr. Rolf Backofen
2. Prof. Dr. Peter Stadler

Tag der letzten Prüfung des Rigorosums: 31. März 2005

Tag der öffentlichen Verteidigung: 27. April 2005

Exact, Constraint-Based Structure Prediction in Simple Protein Models

Sebastian Will

16th January 2005

Zusammenfassung

Die Arbeit untersucht die exakte Vorhersage der Struktur von Proteinen in dreidimensionalen, abstrakten Proteinmodellen; insbesondere wird ein exakter Ansatz zur Strukturvorhersage in den HP-Modellen [LD89] des kubischen und kubisch-flächenzentrierten Gitters entwickelt und diskutiert. Im Gegensatz zu heuristischen Methoden, liefert unser exaktes Verfahren beweisbar korrekte Strukturen. HP-Modelle (**H**ydrophob, **P**olar) repräsentieren die Rückgratkonformation eines Proteins durch Gitterpunkte und berücksichtigen ausschließlich die hydrophobe Wechselwirkung als treibende Kraft bei der Ausbildung der Proteinstruktur. Wesentlich für die erfolgreiche Umsetzung des vorgestellten Verfahrens ist die Verwendung von constraint-basierten Techniken. Im Zentrum steht die Berechnung und Anwendung hydrophober Kerne für die Strukturvorhersage.
Proteinstrukturvorhersage, d.h. das Problem aus einer gegebenen Sequenz deren native Struktur – die Struktur mit minimaler Energie – vorauszusagen[1], gilt als eines der bedeutendsten ungelösten Probleme der Bioinformatik. Proteinstrukturvorhersage erscheint aus zweierlei Gründen als Heiliger Gral der Bioinformatik:

- Die Hauptmotivation für Strukturvorhersage lässt sich überspitzt als

  $$\text{Struktur} = \text{Funktion}$$

  formulieren. Gemeint ist damit, dass erst die Struktur eines Proteins direkte Schlüsse auf dessen Funktion erlaubt. Die Aufklärung der Funktion biologischer Makromoleküle muss als Hauptaufgabe der molekularbiologischen Forschung angesehen werden. Die indirekte Aufklärung der Funktion bisher unbekannter Proteine durch Homologien zu bereits bekannten Molekülen stösst dabei zunehmend an ihre Grenzen; dies ist vor allem eine Konsequenz aus dem nächsten Punkt.

[1] Es wird hier, wie in diesem Kontext üblich, vereinfachend angenommen, dass keine anderen Faktoren als die Minimierung der freien Energie die native Struktur einer Sequenz bestimmen.

- Unser derzeitiges Wissen über biologische Makromoleküle, insbesondere Proteine, ist stark auf das Wissen über die Sequenzen, d.h. die lineare Abfolge der Bausteine dieser Heteropolymere fokussiert. Dieses Wissen resultiert aus den immer besser beherrschten Techniken zur DNA-Sequenzierung. Die drei-dimensionalen Strukturen dieser Moleküle sind jedoch in sehr viel weniger Fällen bekannt. Experimentelle Methoden zur Strukturaufklärung, d.h. Röntgenstrukturanalyse und NMR, sind sehr zeitaufwendig und stoßen darüberhinaus vielfach an Grenzen ihrer Anwendbarkeit.

Unser bisher geringes Vermögen zur Strukturvorhersage aus der Sequenz lässt sich im Wesentlichen auf zwei Aspekte zurückführen.

- Schon in stark vereinfachte Varianten wurde Proteinstrukturvorhersage als ein NP-vollständiges Problem nachgewiesen [BL98, CGP+98]. Damit liegt der Schluss nahe, dass das Vorhersageproblem für reale Proteine ebenfalls nicht effizient exakt gelöst werden kann.

- Dem gegenüber steht das als Levinthal-Paradox [Lev69] bekannte Phänomen, dass natürliche Proteine trotz der astronomischen Menge möglicher Strukturen in verhältnismässig kurzer Zeit in ihre Struktur falten. Vermutlich steht also unser ungenügendes Wissen über den Proteinfaltungsprozess, d.h. den Vorgang der Ausprägung der Proteinstruktur, sowie die allgemeine Beziehung zwischen Sequenzen und Strukturen einer effizienten Strukturvorhersage im Wege. Zum einen ist also denkbar, dass durch zusätzliches Wissen über den Faltungsprozess schnelle Verfahren zur Vorhersage gefunden werden können. Zum anderen könnten die Sequenzen natürlich vorkommender Proteine einer Klasse von Sequenzen angehören, deren Strukturen einfacher vorhergesagt werden können, als die von beliebigen Sequenzen. In letzterem Fall werden Verfahren benötigt, die speziell auf diese Klasse zugeschnitten sind. Constraint-basierte Methoden bilden eine adäquate Antwort auf derart strukturierte NP-vollständige Probleme.

Beitrag der Arbeit Die Arbeit trägt in mehrerlei Hinsicht zur Lösung des Proteinstrukturvorhersageproblems bei. Zum einen stellt die Arbeit eine exakte und vollständige Lösung des Problems für zwei vereinfachte Modelle vor. Im kubischen Gitter muss sich das Verfahren an CHCC [YD95] messen lassen, der bisher einzigen konkurrierenden exakten Strukturvorhersagemethode für das HP-Modell. Unser neues Verfahren arbeitet deutlich schneller als CHCC.

CHCC erwies sich darüberhinaus als unvollständig. Für das flächenzentriert-kubische Gitter stellen wir das erste exakte Verfahren überhaupt vor.
Die unterstützten Proteinmodelle, insbesondere das HP-Modell des flächenzentriert-kubischen Gitters, sind deutlich realistischer als zuvor rechnerisch handhabbare Modelle. Vielfach wurden bisher für Untersuchungen entweder zwei-dimensionale Proteinmodelle verwendet oder unrealistische Restriktionen eingeführt, um den rechnerischen Aufwand zu begrenzen. Vorhersage im flächenzentriert-kubischen Gitter ist darüberhinaus ein deutlicher Fortschritt über das kubische Gitter, denn wie [PL95] zeigte, nähert das flächenzentriert-kubische Gitter reale Proteinkonformationen an mit einer Abweichung (coordinate root mean square deviation) von nur 1.78 Å im Vergleich zu 2.48 Å des kubischen Gitters. Durch diese gute Approximation bietet sich Vorhersage im flächenzentriert-kubischen Gitter an als erster Schritt in hierarchischen Ansätzen zur Strukturvorhersage realer Proteine.
Durch das vorgestellte Verfahren wird es erstmals möglich, umfangreiche Untersuchungen mit realistischen Modellen durchzuführen. Die Arbeit zeigt diesen Weg anhand von zwei Beispielanwendungen auf. Durch die Anwendung vereinfachter Modelle lässt sich Einsicht in die Sequenz-Struktur Beziehung und den Faltungsprozess gewinnen.
Schließlich demonstriert die Arbeit den erfolgreichen Einsatz constraint-basierter Methoden für die Strukturvorhersage und weist damit einen möglichen Weg für weitergehende algorithmische Arbeit in diesem Gebiet.

Aufbau der Arbeit In den Kapiteln 1,2 und 3 werden eine ausführliche Einleitung und ein Überblick über das Verfahren zur Strukturvorhersage gegeben. Außerdem werden grundlegende Konzepte beschrieben, die im Überblick und vor allem in den folgenden Kapiteln benötigt werden. Kapitel 4, 5 und 6 gehen jeweils im Detail auf ein Teilproblem des Verfahrens ein. Die Teilprobleme sind Berechnung von Energie-Abschätzungen, Konstruktion kompakter hydrophober Kerne und Abbildung von Sequenzen auf hydrophobe Kerne. Das Kapitel 7 präsentiert schließlich Ergebnisse und Anwendungen der Strukturvorhersage, wobei Anwendungsbeispiele zur Erforschung protein-artiger Sequenzen der Modelle gegeben werden, sowie daraus resultierende neue Ergebnisse.

Danksagung

Zuallererst gilt mein Dank meinem Doktorvater Rolf Backofen. Er hat meine akademische Entwicklung schon seit langer Zeit begleitet und entscheidend zu meinem Weiterkommen beigetragen. Rolf verdanke ich, mein Interesse an Proteinmodellen und Constraint-Programmierung geweckt zu haben, eine gute, erfolgreiche Zusammenarbeit, aus der zahlreiche gemeinsame Veröffentlichungen entstanden, weiterhin seine Arbeiten zu constraintbasierter Strukturvorhersage, auf denen ich aufbauen konnte, und nicht zuletzt seine Ideen und Anmerkungen zu meiner Dissertation.

Dankbar bin ich Peter Stadler, dass er sich bereitfand, meine Dissertation zu begutachten, für sein Interesse an meiner Arbeit und seine hilfreichen Kommentare.

Erich Bornberg-Bauer danke ich besonders für seinen Ideen und Anregungen zu Anwendungen vereinfachter Proteinmodelle. Mein Besuch seines Lehrstuhls in Münster war für mich gewinnbringend und motivierend.

Neben Rolf Backofen habe ich meinen guten Start in die Bioinformatik Peter Clote zu verdanken. Beide hielten zusammen meine erste Bioinformatik-Vorlesung. Die Zusammenarbeit mit beiden als studentische Hilfskraft am Lehrstuhl für theoretische Informatik hat mich entscheidend geprägt.

Außerdem danke ich allen mit denen ich wissenschaftlich zusammenarbeiten durfte, insbesondere meinen Mitautoren. Außer den schon genannten Rolf Backofen, Erich Bornberg-Bauer und Peter Clote, zählen dazu Slim Abdennadher, Agostino Dovier und Alessandro Dal Palù und Matthias Saft. Alessandro verdanke ich eine intensive und erfreuliche Zusammenarbeit während seines zweimonatigen Aufenthaltes in Jena.

Dem Graduiertenkolleg für Logik in der Informatik (GKLI) schulde ich Dank für die Finanzierung der ersten zwei Jahre meiner Promotionszeit in München.

Aus meiner Zeit an der Münchener Universität möchte ich Slim Abdennadher, unseren Techniker Max Jakob, Jan Johannsen, Ralf Matthes, und unsere Sekretärin und meine Gitarrenschülerin Irmgard Mignani dankend erwähnen.

Mein besonderer Dank gilt meinen derzeitigen Kollegen am Lehrstuhl für Bioinformatik Anke Busch (auch für das Korrekturlesen von Teilen der Arbeit),

Michael Hiller, Frank Mäurer, Kathrin Müller, Rainer Pudimat und meinem Zimmergenossen Sven Siebert für ideelle und konkrete Unterstützung, Gespräche und Diskussionen. Weiterhin danke ich Martin Mann für unsere Zusammenarbeit, die unterhaltsame Abwechslung, die seine Besuche in den Büroalltag bringen, und unser gemeinsames Gitarrenspiel.
Nicht zuletzt danke ich herzlich meinen Eltern Christine und Hartmut und meinen Geschwistern Anna und Roland, die mir immer einen sicheren Rückhalt geben, die mir in allen Lebenslagen zur Seite stehen und denen ich weit mehr zu verdanken habe als ich hier in Worte fassen kann.

Jena, den 13.01.2005 Sebastian Will

Contents

Chapter 1

Introduction

In this thesis, we discuss protein structure prediction in two three-dimensional, abstract protein models. Structure prediction can promote the exploration of proteins. Such research is crucial for our understanding of life, since proteins, the nano-machines of life, perform and control the essential functions in living organisms. In contrast to *heuristic* methods, the introduced approach is *exact*, i.e. it yields provably correct predictions.
Protein structure prediction is the task to predict the native structure for a given amino acid sequence. It is commonly regarded the holy grail of bioinformatics for twofold reason:

- The main motivation for protein structure prediction is — slightly oversimplifying — formulated as

  $$\text{structure} = \text{function}.$$

  This means that only the structure of a protein gives direct hints to its function. Elucidating the function of proteins has to be considered a main goal of molecular biological research. Indirect exploration of the function of an unknown protein via homology to known proteins is increasingly limited; mainly since knowledge about sequences is growing much more rapidly than knowledge about protein function (cf. second reason).

- Our current knowledge about biological macromolecules, especially proteins, is strongly focused on sequences, i.e. the linear chain of monomers in the hetero-polymers. Mainly, this knowledge results from our increasing understanding of DNA-sequencing methods. The three-dimensional structures of these molecules are known to an much less extent. Experimental techniques for solving protein structures — like X-ray

crystallography and NMR — are still very time consuming and not applicable in all cases.

Our limited ability to predict native structures from the sequence information alone can be ascribed to at least the following reasons.

- Already strongly simplified variants of protein structure prediction problem were shown to be NP-complete [BL98, CGP+98, HI97b]. This suggests that there is no efficient and exact solution for the structure prediction problem of real proteins.
- This is contrasted by the phenomenon that is well known as Levinthal paradox [Lev69]. This paradox says that many proteins fold quickly into their native structure, whereas the structure space is of astronomical size. Presumably, our insufficient knowledge about protein folding and the relationship of sequence and structure hinders the development of efficient protein structure prediction. On the one hand, there is a good chance that increased knowledge about protein folding can be used for fast structure prediction. On the other hand, sequences of natural proteins could belong to a class, where structures are much easier to predict than for the majority of sequences. In the latter case, one needs approaches that are tailored for sequences of this class. Constraint-based methods are especially suited for NP-complete problems with such a structure.

Contribution

The thesis contributes in several aspects to the solution of the protein structure problem. On the one hand, the work introduces an exact and complete solution to the problem for two simplified, three-dimensional models. For the cubic lattice, we have to compare our approach to CHCC, which is the only second exact structure prediction method for HP-models. Our new approach is significantly faster than CHCC. Furthermore, CHCC turned out to be incomplete, which will be discussed in more detail. For the face-centered cubic lattice (FCC), we present the first exact method ever.
Both models, especially the HP-model of the FCC lattice, are much more realistic than all models that could be handled computationally before. In many cases, studies used two-dimensional lattice models or introduced unrealistic restrictions, just for computational tractability. Furthermore, prediction in the FCC lattice is a major advance over the cubic lattice, as argued in more

detail later. By the good approximation of off-lattice structures in the FCC, prediction in the FCC suggests itself as a first step to structure prediction of real proteins in hierarchical approaches.
Due to our approach, it is possible to perform large scale studies using realistic protein models for the first time. Studies with simplified models can provide insight into the relationship of sequence and structure and the protein folding process. By giving two example applications, we demonstrate how our approach can be applied for such research.
Finally, the thesis demonstrates the successful use of constraint-based methods for structure prediction. By this, it shows a possible way for further algorithmical research in this area.

Overview

In Chapters 1 and 2, we give a detailed introduction and overview of protein structure prediction and our approach. In Chapter 3, we describe fundamental concepts, which are used in the overview and in particular in the following three chapters. Chapters 4, 5, and 6 each describe in detail a sub-problem of our approach, where Chapter 3 contains all the common foundations for the three chapters. The three sub-problems are computation of energy-bounds, construction of compact cores, and mapping of sequences to hydrophobic cores. Finally, Chapter 7 presents results and applications of structure prediction. In particular, we present applications for exploring protein-like sequences and give new results.

1.1 Proteins

Viewed on the molecular level, proteins are chains that are composed of building blocks, which share a common structure. More specifically, a protein is a linear polymer formed by connecting monomers, which are called amino acids. An amino acid is a molecule of the form shown in Figure 1.1. All amino acids share the same general structure and differ only in the chemical group R. The central carbon atom is called the *α-carbon* (short Cα), the left group NH_2 is called the *amino group*, and the right group COOH is called *carboxy group*. In living organisms, there occur 20 different amino acids[1], which have

[1]Commonly, only 20 different amino acids occur in organisms, whereas many more amino acids can be synthesized. Recently, a 21st amino acid that occurs in organisms was discovered.

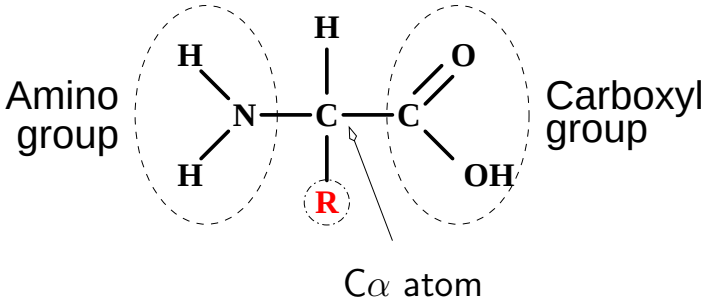


Figure 1.1: General structure of an amino acid.

different chemical properties. For example, residues can be hydrophobic or hydrophilic, small or large, charged or uncharged.
In a protein, the amino acids are linearly arranged thereby forming a chain. The order of amino acids in this chain is called *sequence* of the protein and is specific for each protein. One observes that by simple combinatorics, this allows for a huge variety of different proteins. For example, there are $20^{100} \approx 10^{130}$ different sequences of length 100.
In order to form a chain, two successive amino acids are connected via a *peptide bond*, where the carboxy group of the first amino acid reacts with the amino group of the second one. The result of connecting two amino acids is a molecule as shown in Figure 1.2.

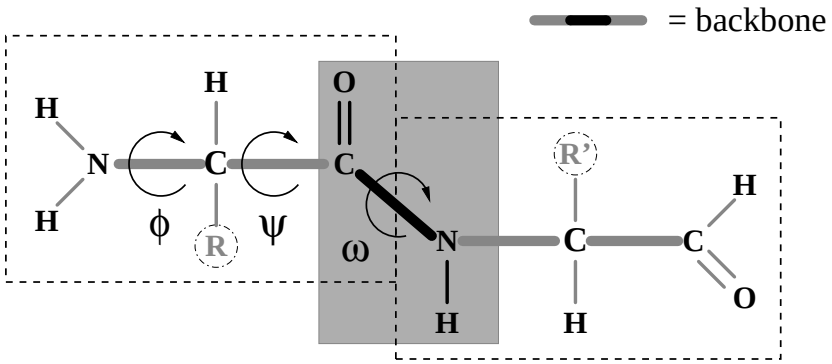


Figure 1.2: Two amino acids connected by a peptide bond.

The peptide bond itself, which is indicated with a grey rectangle in Figure 1.2, is usually planar, which means that there is no free rotation around this

bond.[2] There is more flexibility for rotation around the N-Cα-bond (called the ϕ-angle) and around the Cα–C bond (called the ψ-angle). But even there, the allowed values of combinations of ϕ and ψ angles are restricted to small regions in natural proteins.
Using this freedom of rotation, the protein can form a huge variety of different three-dimensional structures. Due to thermodynamics, some of them are energetically more favorable than others, i.e. these conformations have lower energy and therefore are more stable than the other ones. As of yet, the details of this energy function are not completely clear and a matter of intensive research. In the context of structure prediction, it is commonly assumed that natural protein have a distinguished conformation that has minimal energy and is uniquely determined by the sequence of amino acids. For this reason, one speaks of the *native structure* of a protein denoting this distinguished conformation. The term *protein folding* denotes the conformational search of the protein, which culminates in finding the native conformation. This term is distinguished from the term *protein structure prediction*, which denotes the computation of the native structure from the sequence. In this thesis, we deal with structure prediction and are not concerned with the protein folding process.
Finally, the native structure and sequence determine the function of the protein, due to the general mode of operation of proteins.

1.2 Structure Prediction

Computational structure prediction is especially valuable, since experimental structure determination — X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy — is still difficult and time consuming. In consequence, current biochemical methods have difficulties to keep pace with the rapid growth of the number of known protein sequences.
Therefore, protein structure prediction is one of the most important problems of computational biology, which is however still unsolved in general. We specify this problem in the following way. Given a protein by its sequence of amino acids, what is its native structure? Since, as already discussed, the native structure is the structure with minimal energy, protein structure prediction is reasonably modeled as an optimization problem. The NP-completeness of

[2] There are two conformations for the peptide bond, namely *trans* (corresponding to a rotation angle of 180°), and *cis* (corresponding to a rotation angle of 0°). The *cis* conformation is rare and usually occurs only in combination with the amino acid Proline.

this problem has been proven for many different formal, in general simplified protein models including lattice and off-lattice models. [BL98, CGP+98]
These results strongly suggest that the protein folding problem for real proteins is NP-hard. Therefore, it is unlikely that a general, efficient algorithm for solving this problem can be given. In fact, one is not able to answer this question definitively, since too little is known about the general principles of protein folding.
Knowing these general principles not only would certainly improve our capabilities to predict the structure of a protein, but it is also of paramount importance for rational drug design, where one faces the difficulty to design proteins that have a unique and stable native structure.

1.3 Simplified Models of Proteins

Mainly, we will discuss the most important sub-class of simplified protein models, although parts of this section apply to simplified models in general. The models in this class are called lattice models. The simplifications that are commonly used in this class of models are:

- each monomer is modeled by only one point,
- the positions of the monomers are restricted to lattice positions,
- all monomers have equal size,
- all bonds are of equal length, and
- a simplified energy function is used.

For the aim of structure prediction, gaining insight into the relationship between sequence and structure of proteins is of utmost importance. *Simplified protein models*, also known as low-resolution or coarse-grained protein models, were proposed to study this relationship. Furthermore, they can be used to tackle structure prediction directly.
For both areas of application, the following consideration motivates the use of simplified models. Since simplified models describe only some aspects of the structure and energy of a protein, it is often easier to compute optimal structures for the proteins of the model than for real proteins. The same applies for solving related problems like the design of sequences that fold to a given structure (sequence design problem).

For the application area of direct structure prediction, simplified models have been successfully used by several groups in the international contest "Critical Assessment of Techniques for Protein Structure Prediction" (CASP, see the meeting review of CASP3 [KL99]). There, the models are used in hierarchical approaches for protein folding [XHLS00] (see also Figure 1.3). In general, these approaches use simplified models in a filter step as follows. Given a protein sequence, first a set of good structures in the simplified model is generated (e.g. 10 000 structures). In subsequent steps, these candidate structures are fine-tuned using more computationally involved methods. Usually, these methods incorporate biological knowledge and simulation of protein folding on full atomic detail (i.e. molecular dynamics simulation).

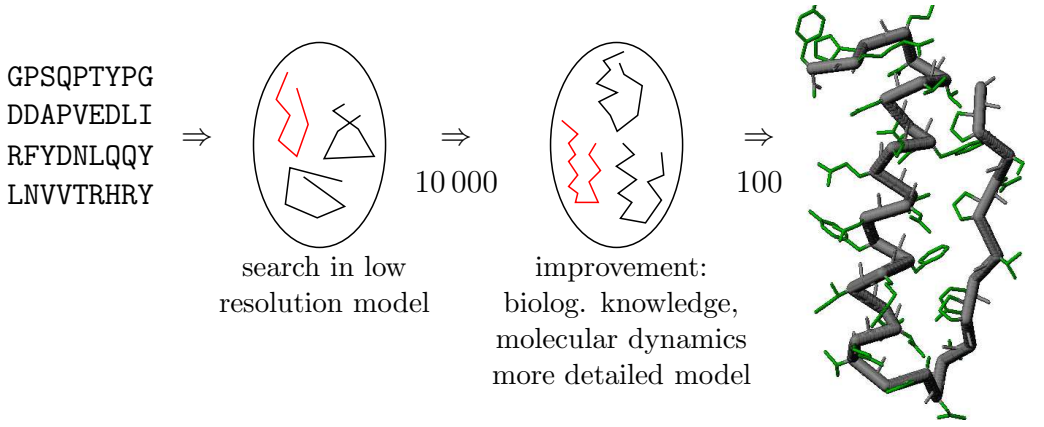


Figure 1.3: Hierarchical approach to protein structure prediction

Apart from their use in structure prediction and regarding the relationship of sequence and structure, lattice models have become a major tool for investigating general properties of protein folding. They constitute a genotype (protein sequence) to phenotype (protein conformation) mapping that can be handled using computational methods. An interesting application coming out of this is their use to investigate evolutionary processes. An example is [BBC99], where the arrangement of sequences in neutral nets is shown. In these neutral nets, a prototype sequence is connected to related sequences, which fold to the same structure, however with increasing hamming distance in a less stable way. The assumed universality of this principle feeds the hypothesis of super-funnels in the sequence space with direct implications to protein design. Another exciting question in this context is, whether one can switch between two different neutral nets using only a small number of amino acid substitutions. If this is the case, then this suggests a way for producing

the diversity of protein conformations that is found in nature by evolution. In the literature, many different lattice models (i.e., lattices and energy functions) have been used (see related work). Of course, the question arises which lattices and energy functions have to be preferred. There are two (somewhat conflicting) aspects that have to be evaluated when choosing a model:

- the accuracy of the lattice in approximating real protein conformations and the ability of the energy function to discriminate native from non-native conformations.
- the availability and quality of search algorithm for finding minimal (or low) energy conformations.

While the first aspect is well-investigated in the literature (e.g., see [PL95, DBY+95]), the second aspect is underrepresented. By and large, there are mainly two different heuristic search approaches used in the literature. The first approach is an ad hoc restriction of the search space to compact or quasi-compact conformations. A good example is [SSK94], where the search space is restricted to conformations forming a $n \times n \times n$-cube. The main drawback here is that the restriction to a compact conformation is not biologically motivated for a complete amino acid sequence (as done in these approaches), but only for the hydrophobic amino acids. In consequence, the restriction either has to be relaxed, and then leads to an inefficient algorithm, or is chosen too strong and then may exclude minimal conformations. The second approach is to use stochastic sampling as performed by Monte Carlo methods with or without simulated annealing or genetic algorithms. Here, the degree of optimality for the best conformations and the quality of the sampling cannot be determined by state of the art methods.[3]

1.4 Constraint-based Structure Prediction

The thesis discusses protein structure prediction in an important class of lattice models, which is known as the class of HP-models. As our main contribution, we introduce a constraint-based approach that outperforms all existing approaches in HP-model lattice protein folding.
Originally, the term HP-model has been introduced by Lau and Dill in [LD89] to denote a two-dimensional square lattice model with an energy function,

[3]Despite there are mathematical treatments of Monte Carlo methods with simulated annealing, the partition function of the ensemble (which is needed for a precise statement) is in general unknown.

a)

	H	P
H	-1	0
P	0	0

b)

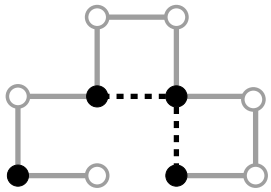


Figure 1.4: Energy matrix and sample conformation for the HP-model

which is simplified as much as possible. In this model, the 20 letter alphabet of amino acids is reduced to a two letter alphabet, consisting of H and P. The symbol H represents *hydrophobic* amino acids, whereas P represents *polar* (*hydrophilic*) amino acids. The energy function for the HP-model is given by the matrix of Figure 1.4a. It simply states that the energy contribution of a contact between two monomers is -1 if both are H-monomers, and 0 otherwise. Two monomers form a *contact* in some specific conformation if the euclidian distance of their positions is 1 and they are not connected via a bond.[4] A conformation with *minimal energy* (also called *optimal conformation*) is just a conformation with the maximal number of contacts between H-monomers. Just recently, NP-completeness of the structure prediction problem has been shown even for the HP-model [BL98, CGP$^+$98].
A sample conformation for the sequence PHPHPPHPPH in the two-dimensional square lattice with energy -2 is shown in Figure 1.4b. The white beads represent P, the black ones H monomers. The two HH-contacts are indicated via dashed lines.
In particular, we outperform prior approaches for structure prediction in three important aspects, namely in terms of

- flexibility,
- completeness, and
- efficiency.

Concerning flexibility, our method is the only one that works for two different important three-dimensional lattices. Here, we note again that originally the HP-model was defined for the two-dimensional square lattice. However, the extension to other lattices is straightforward. For example, a HP-model using the face-centered cubic lattice is investigated in [ABD$^+$97].

[4]Note that this second condition can be dropped without changing the optimization problem, since it adds only a constant number of contacts. Actually, we will do this later.

The treated lattices are the cubic lattice and the face-centered cubic lattice. The cubic lattice is the most intensively studied three-dimensional lattice. However, the ability of this lattice to approximate real protein conformations is poor. Furthermore, as e.g. [ABD+97] pointed out, there is a parity problem in the cubic lattice. This means that two monomers with chain positions of the same parity cannot form a contact. Nevertheless, we support the cubic lattice due to its wide-spread use in literature and for comparability of our method to other structure prediction approaches.

The face-centered cubic lattice (FCC) overcomes the discussed drawbacks of the cubic lattice. It lacks the parity problem and models real protein conformations with good quality (see [PL95], where it was shown, that the FCC lattice can model protein conformations with coordinate root mean square deviation of 1.78 Å, whereas the cubic lattice achieves a deviation of only 2.84 Å). Recently, [BJB02a, BJB02b] have shown that neighborship of amino acids in proteins closely resembles a distorted FCC lattice and that the FCC is best suited for modeling proteins. This is an immediate effect of hydrophobic packing. Just recently, it was shown that the FCC is the lattice allowing the densest packing of identical spheres, approximately 400 years after the original conjecture by Kepler [Slo98, Cip98] (see Figure 1.5 for a description of the FCC lattice).

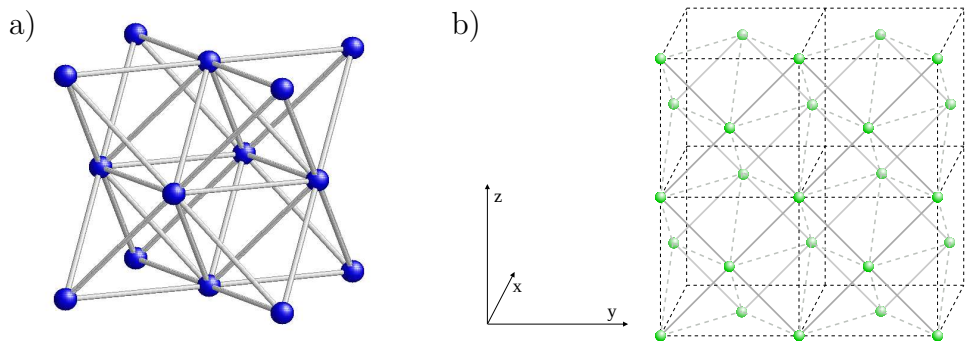


Figure 1.5: a) The unit cell of the FCC. b) A cut-out of two layers of the face-centered cubic lattice (FCC). The layers can be seen as two square lattices, which are shifted such that every position in the first layer has contacts to 4 positions in the second layer and vice versa (shown as dashed lines). Since the FCC lattice is continued by stacking layers of the square lattice in this way, every position of the FCC has twelve neighbors (four within the same layer, four in the previous layer, and four in the next layer).

Concerning completeness, our approach finds optimal structures that can not be computed by all other comparable approaches. The fact that the HP-model is degenerated[5] allows for a direct comparison of completeness. The degeneracy (i.e. the number of optimal structures) of a sequence can be computed by enumerating all optimal conformations. Only for the cubic lattice, there is one other method [YD95] that claims to completely enumerate optimal conformations in a large class of conformations *and* to prove their optimality. In [YD95], Yue and Dill give a lower bound for the number of such conformations for some sequences, by enumerating as many structures as their algorithm can find. For these sequences we can significantly improve their lower bound, which shows that the CHCC algorithm is incomplete. Note that an incomplete algorithm can not only miss optimal conformations, but even fail to determine the optimal energy for structures of a sequence.
Concerning efficiency, we have successfully applied our algorithm to sequences up to length 200 in the face-centered cubic lattice (FCC). For several sequences of length 200, we found a minimal energy conformation *and* proved its optimality. For the FCC, there existed only heuristic algorithms up to now (for an example of a genetic algorithm for arbitrary Bravais lattices see [BWC00]). Usually, these algorithms are applied to sequences of length of at most 80 (where they usually find only a low but *not* minimal energy conformation). Since the search space for conformations in the cubic lattice grows with approximately 4.5^n (where n is the length of the sequence), this implies that our method handles a search space that is at least by the factor 4.5^{120} higher than the search space handled by other methods for the face-centered cubic lattice.[6]

1.5 Related Work

Here, we distinguish two aspects in the relationship to prior work. On the one hand, we discuss work that investigates biological macromolecules, i.e. mainly proteins and also RNA, by using computationally tractable models. This work studies structure, thermodynamical stability, folding kinetics and evolution of proteins and RNA. On the other hand, we review algorithmical

[5] In a degenerated model there are usually many optimal conformations for one sequence, instead of only one (native) conformation as it is assumed for real proteins.

[6] The number 4.5^n has been estimated for the cubic lattice [MS96]; for the FCC, we are not aware of a good estimation of the number of conformations. However, due to the increased degrees of freedom in the FCC lattice this number is certainly higher than the number of conformations in the cubic lattice.

work on structure prediction in lattice protein models.

For the first aspect, there is a large number of working groups that use lattice protein models. The kinetics of protein folding is investigated by e.g. [SD03, SR01, GSA+98, UM96, DSK96, AGS95, SSK94]. All groups perform Monte-Carlo optimization in order to simulate the folding process. One of their main observations is that sequences can be distinguished in fast and slow folding ones. For example, [SSK94] claims that fast folders are characterized by a large energy gap between the optimal conformation and the next best one. Others [AGS95] claim that folding speed is increased if there are many more non-local than local contacts in the ground state. In contrast, [UM96] observe in a larger study that sequences with many strong local interactions fold quickly and also argue against the energy gap hypothesis. [GSA+98] discuss how folding time depends on the temperature.

A prerequisite for such work is to know the optimal structures of sequences. Therefore, the authors choose protein models on the square lattice or cubic lattice, where structures can be predicted by complete enumeration. There, complete enumeration is only tractable due to the artificial restriction to compact conformations, e.g. on a $3 \times 3 \times 3$ cube for polymers of length 27 in three dimensions or on a 5×5 square for polymers of length 25 in two dimensions. Most of this work uses energy functions based on pairwise potentials of monomers in unit distance. [SD03] is an example for the use of Go models[7] in such studies, which circumvents the need for structure prediction by the use of an artificial energy function that is tailored for the target structure.

Other work studies protein evolution by the use of simple protein models; examples are [WBBC04, CWBBC02, TG00a, TG00b, AGS97, GG96, BBC99, GG97]. Some of these papers use similar models with a restriction to compact conformations as described for the papers on folding kinetics. For example, [GG97] asks why some structures are more common than others, which is investigated by using a cubic lattice model with the restriction to compact conformations on a $3 \times 3 \times 3$ cube. Others [WBBC04, BBC99, CWBBC02] use sequences of length 18 in two-dimensional square lattice models. Besides the HP-model, [WBBC04] investigates five other models with a binary code. Their main finding is the occurrence of neutral nets and a super-funnel topology of the sequence space in a variety of protein models, which suggests a more general principle. Notably, the use of the HP-model and a limited length allows an unrestricted enumeration of sequences and their structures.

[7] A Go model uses an energy function that favors the contacts that occur in the target structure.

Several papers [IT02, LTW02, HL96] especially address protein design, also known as inverse protein folding, which is of great importance for evolutionary questions. [IT02] enumerates designing sequences of length 25 in the two-dimensional HP-model. [SSG+00] investigates designable structures and tries to explain that formation of secondary structure and collapse to a globule happens rapidly at the same time. [LTW02] compares HP-models to models with an empirical potential, using the Miyazawa-Jernigan matrix [MJ96] and finds a good agreement in the designable structures.

A review of simplified protein models and their application is e.g. given in [DBY+95] and [CBB02]. In particular, these reviews advocate the use of strong simplifications as found in the HP-model.

Concerning the algorithmical aspects of the research above, most of this work avoids more sophisticated algorithms for protein structure prediction by using protein models, where complete enumeration is still possible. Therefore, one has to introduce strong restrictions as the limitation to rather short chains (e.g., length 18 in [WBBC04] and others) or the rather artificial restriction to compact conformations (e.g. on a 6×6 square and $3 \times 3 \times 3$ cube in [LTW02] and others). However, there is research [IT02] that pushes the length limitation to its limits by completely enumerating the designing sequences of length 25 in the square lattice HP-model. Enumerating structures of lattice proteins is similar to counting self-avoiding walks. [MJH+00] counts such walks in the cubic lattice up to length 26. However, the applied method is not transferable to the enumeration of protein structure, since there we need to compute the energies of conformations.

Structure prediction for RNA was available much earlier than for proteins. More precisely, in the case of RNA there are efficient algorithms for predicting the secondary structure of minimal free energy from the sequence information [ZS81]. For RNA, even complete sub-optimal folding can be done efficiently [WFHS99]. The availability of an efficient sequence-structure map for RNA motivated studies on RNA evolution like [SFSH94], RNA folding kinetics [FFHS00], and general work on fitness landscapes, e.g. [FHSW02], which develops a theory of barrier trees in degenerate fitness landscapes and applies it to RNA and spin glass landscapes.

For our second major aspect, namely the algorithmical work on protein structure prediction, we distinguish heuristic and exact approaches. Here, heuristic approaches are those which can not guarantee the optimality of the predicted optimal structure. In contrast, exact methods are able to prove that computed structures are optimal or at least within a certain energy ratio of the optimum.

Work on heuristic methods for protein structure prediction ranges from Monte-Carlo simulated annealing (e.g. [Gra04, LW01, BFG+98]) and genetic algorithms (e.g. [JCSM03, UM93b, UM93a]) to methods like hydrophobic zipper [DFC93] and the chain growth algorithm [BB97].
A heuristic approach of predicting structures using a FCC lattice-model and also constraint-technology is the work of [DBF02]. There, secondary structure annotations, i.e. which amino-acids form α-helices and β-sheets, are employed to restrict the search space. Despite some similarities to our approach (due to lattice and technology), the aim of their work is to find good, but not necessarily optimal solutions in a protein model with empirical potential. However, the method is slow and only applicable to short proteins, although their method cannot guarantee the quality of found solutions.
In general, heuristic methods are unsatisfying for many applications of simple protein models, including most of the ones that are mentioned above. In this respect, the first improvement was the introduction of an exact algorithm for finding minimal energy conformations in the cubic HP-Model [YD93, YD95]. The algorithm is called CHCC for "**C**onstraint **H**ydrophobic **C**ore **C**onstruction". Note that albeit its name, the approach does not use constraint-based methods. In comparison to our work, CHCC is slow in predicting optimal structures. Furthermore, we found for several of available example sequences that CHCC does not completely enumerate all optimal structures. Reasons for the incompleteness of CHCC will be discussed later on in Chapter 5. Finally, the CHCC method works only for the cubic lattice, but not for the more complex face-centered cubic lattice.
The second improvement is the appearance of efficient approximation algorithms [Heu03, ABD+97, HI96] for different lattice models. [HI96] guarantees an approximation within 3/8 of the optimum in the square and cubic lattice HP-models. [ABD+97] develop constant factor approximations for the HP-models of the two-dimensional triangular lattice and the three-dimensional face-centered cubic lattice. [Heu03] even includes side chains in its modified cubic lattice protein model. However, despite indisputable merits, the approximation ratio of these algorithms is still too weak for practical use.

Chapter 2

Fundamental Concepts

This chapter describes the basic preliminaries and fundamental concepts of the thesis. Here, we endeavor to give only those definitions and to explain only those concepts that apply to more than a single chapter of the thesis. Thus, we minimize redundancy and in the same time avoid unnecessary scattering of definitions. In consequence, the Chapters 3,4,5, and 6 are formally independent and each can be read on its own.
We note that it is unavoidable that some terms, like lattice and lattice protein models, that were already mentioned in the introductory chapter, reoccur and are redefined formally.
First, we will review lattices and lattice proteins. Second, we give some background on constraint technology, in particular constraint satisfaction and constraint optimization.

2.1 Lattices

The discussed models of proteins rely on a discretization of the two or three-dimensional space by geometrical structures that are known as lattices.

Definition 2.1.1 (Lattice)
A lattice *is a set L of* lattice vectors *(also called* lattice points*) such that*

$$\vec{0} \in L \tag{2.1}$$

$$\vec{u}, \vec{v} \in L \text{ implies } \vec{u} + \vec{v}, \vec{u} - \vec{v} \in L, \tag{2.2}$$

where $+$ and $-$ denote vector addition and subtraction and $\vec{0}$ is the zero vector.

By this definition, a lattice forms an additive group with operator $+$, its inverse operator $-$, and neutral element $\vec{0}$. Note that the term lattice conflicts with the term for ordered sets of lattice theory, which is a completely different concept. Sometimes, the term *point lattice* is used in place of our term lattice to avoid confusion.
For a lattice L, there exist n vectors $\vec{v}_1, \ldots, \vec{v}_n$ such that the lattice consists of all the integral linear combinations of these vectors, i.e.

$$L = \{\sum k_i \vec{v}_i \mid k_1, \ldots, k_n \in \mathbb{Z}\}. \tag{2.3}$$

If n is minimal with the property (2.3), then $\vec{v}_1, \ldots, \vec{v}_n$ is a *basis of* L and n is the *dimension of* L. Inversely, for vectors $\vec{v}_1, \ldots, \vec{v}_n$, where Eq. (2.3) holds, L is called *generated by the vectors* $\vec{v}_1, \ldots, \vec{v}_n$. This allows that a lattice L is generated by a set of vectors that is larger than its dimension. By definition, each basis of a lattice L also generates L.
We define a neighborship relation between points of a lattice L by fixing a set of *neighbor vectors* NV for L. Two lattice points $\vec{p}$ and $\vec{p}'$ are called *neighbors* of each other if and only if $\vec{p} - \vec{p}' \in NV$.
The three-dimensional lattices that are used for protein models are naturally embedded into Euclidian space $\mathbb{R}^3$.[1] In this embedding, their lattice points are represented as

$$\vec{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}.$$

Then, a natural choice for the set of neighbor vectors NV is the set of non-zero vectors $\vec{p}$ which have minimal Euclidian length

$$\sqrt{p_x^2 + p_y^2 + p_z^2}$$

among all lattice vectors. We call these vectors *minimal vectors*. If we do not explicitely define otherwise, we choose the minimal vectors as neighbor vectors. In this case, the number of minimal vectors equals the number of neighbors of a single lattice point. This number is an important property of a lattice, it is called *coordination number* or in the context of sphere packings *kissing number*.
It is useful to identify regular sub-sets of a lattice. For a three-dimensional lattice L, we define the *layer* $a_x x + a_y y + a_z z = b$ *of* L as the set of points

$$\left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in L \mid a_x x + a_y y + a_z z = b \right\},$$

[1] It goes without saying that the same applies to two-dimensional lattices analogously.

where $a_x, a_y, a_z, b \in \mathbb{R}$. For a layer, we require that this sub-set of the lattice is non-empty. Special layers $\xi = c$, where $\xi \in \{x, y, z\}$ and $c \in \mathbb{R}$ are denoted ξ-layers.

2.1.1 The Cubic Lattice

The *cubic lattice* is probably the most prominent and the simplest three-dimensional lattice. It is defined as the set of lattice points $\mathbb{Z}^3$ and generated by its basis

$$\{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}\}.$$

The minimal length of a non-zero vector in the cubic lattice is 1 and the set of minimal vectors is given by

$$\{\begin{pmatrix}\pm1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\\pm1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\\pm1\end{pmatrix}\}.$$

Thus, the coordination number of the cubic lattice is six, i.e. each point has six neighbors.
Figure 2.1 shows a cutout of the cubic lattice, where neighbored points are connected. The connections form angles of 90° and multiples thereof.

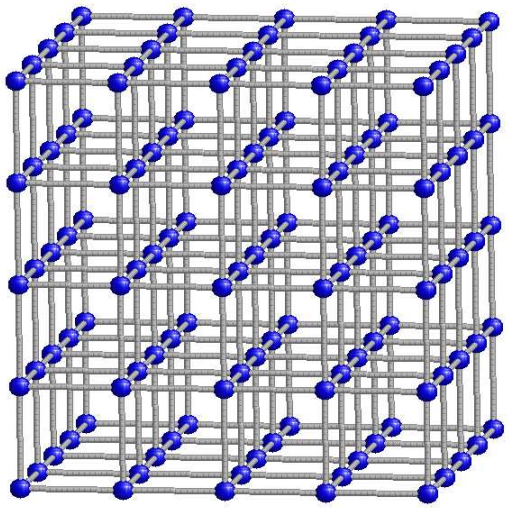


Figure 2.1: The cubic lattice, where connections are drawn according to the neighbor relation.

The cubic lattice can be partitioned into layers that form square lattices. For each $\xi \in \{x, y, z\}$ and $c \in \mathbb{Z}$, there is a layer $\xi = c$ of the cubic lattice. Each

point of a layer $\xi = c$ has four neighbors in the same layer, one neighbor in the layer $\xi = c - 1$, and one in the layer $\xi = c + 1$.
An interesting property of the cubic lattice is that its points are naturally partitioned into two disjoint classes by the neighbor relation; namely, into points with even sum

$$\mathbb{Z}^3_{|\text{even}} = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{Z}^3 \mid x + y + z \text{ is even} \right\}$$

and points with odd sum

$$\mathbb{Z}^3_{|\text{odd}} = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{Z}^3 \mid x + y + z \text{ is odd} \right\}.$$

Every point in one of the two classes has only neighbors in the other class, since adding any neighbor vector to a point changes the parity of the sum of its coordinates.
This property of the cubic lattice was used in approximate and exact structure prediction before [HI96, Bac98b]. The property is also known as the *parity problem* of the cubic lattice, since it artificially restricts possible contacts in the cubic lattice.

2.1.2 The Face-Centered Cubic Lattice

Since Kepler's famous conjecture four centuries ago, the *face-centered cubic lattice* (FCC) was believed to be a densest packing of spheres in three dimensions. The conjecture became a theorem only recently, when it was proven by Thomas C. Hales in 1998 (cf. [Cip98]).[2]
The FCC is defined as the set of points

$$D_3 = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{Z}^3 \mid \ x + y + z \text{ is even} \right\}.$$

The minimal distance between two lattice points, i.e. the length of the minimal vectors, is $\sqrt{2}$. There are twelve minimal vectors in the FCC, namely the vectors of the form

$$\begin{pmatrix} \pm 1 \\ \pm 1 \\ 0 \end{pmatrix}, \ \begin{pmatrix} \pm 1 \\ 0 \\ \pm 1 \end{pmatrix}, \text{ or } \ \begin{pmatrix} 0 \\ \pm 1 \\ \pm 1 \end{pmatrix}.$$

The Figures 2.2a and b show the unit cell of the face-centered cubic lattice. It is constructed by placing points at the corners of a cube and at the centers of its faces; this construction principle, which explains the origin of the name

[2] Remarkably, already in 1831 Gauss showed that the FCC lattice is the densest *lattice* packing of spheres.

of the lattice "face-centered cubic", is best seen in Figure 2.2a. Figure 2.2b shows the same lattice points but connects neighbors that have minimal distance $\sqrt{2}$. Figure 2.3 shows a larger cutout of the face-centered cubic lattice.

If we slice the lattice into layers that are parallel to the faces of the cube, the lattice points in each layer form a square lattice. For each $\xi \in \{x, y, z\}$ and $c \in \mathbb{Z}$, there is such a layer $\xi = c$ of the FCC lattice. A point in such a layer $\xi = c$ has four neighbors in the same layer, four neighbors in the layer $\xi = c - 1$, and four neighbors in $\xi = c + 1$ (cf. cubic lattice).

There is a different way to partition the lattice points into layers such that the points in each layer form a hexagonal lattice (also known as triangular lattice), where the connections form angles of 60°.[3] There are four classes of such layers, namely for each even $c \in \mathbb{N}$, there is a layer $x + y + z = c$, a layer $-x + y + z = c$, a layer $x - y + z = c$, and a layer $x + y - z = c$. A point of such a layer has six neighbors in the same layer, three neighbors in the preceeding layer, and three neighbors in the succeeding layer of the same class.[4]

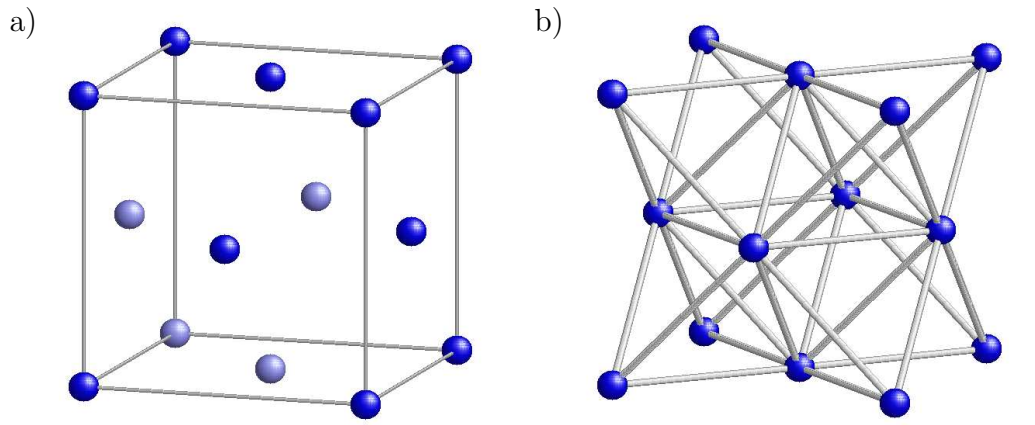


Figure 2.2: Unit cell of the face-centered cubic lattice. a) cube with lattice points at corners and centers of faces. b) edges between neighbors.

[3] It is well known that there are two equivalent representations of the FCC, namely the D_3 and the A_3 representation. The A_3 view of the FCC points out its relation to the hexagonal lattice.

[4] For example, the preceeding (resp. succeeding) layer of layer $x+y+z = c$ is $x+y+z = c - 2$ (resp. $x + y + z = c + 2$).

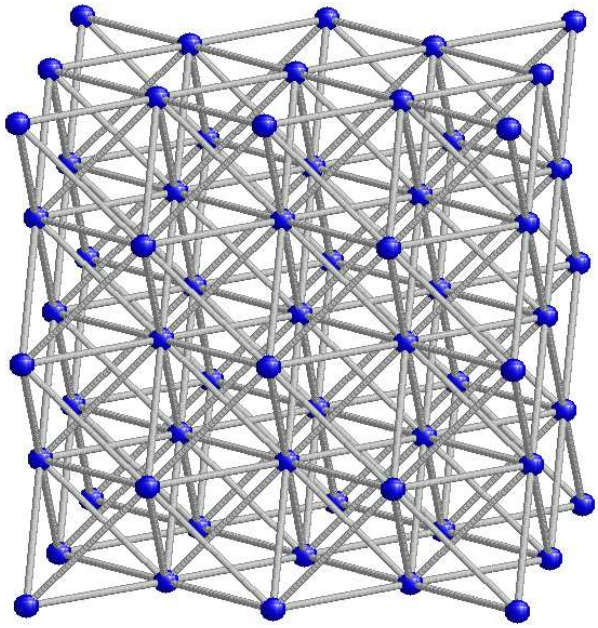


Figure 2.3: The face-centered cubic lattice.

2.1.3 Finite Sets of Lattice Points

Here, we discuss finite sub-sets $P \subset L$ of a lattice L, in the following called *finite point sets*, or more shortly *point sets*. We recall that $|P|$ denotes the cardinality of P.
First, we discuss some properties that are induced by the neighborship relation. Pairs of points $\vec{p}$ and $\vec{p}'$ in P that are neighbors form a *contact*. The *number of contacts* in P is defined as

$$\text{contacts}(P) = |\{\{\vec{p}, \vec{p}'\} \mid \vec{p}, \vec{p}' \in P \text{ are neighbors}\}|.$$

A point set of size n that has maximally many contacts among all point sets of size n shall be called *optimal point set of size n*.
According to the neighbor relation, a point set P is either connected or consists of several connected components. P is called *connected* if and only if every two points $\vec{p}$ and $\vec{p}'$ in P are related by the transitive closure of the neighbor relation, i.e. for every $\vec{p}$ and $\vec{p}'$, there exist points

$$\vec{p} = \vec{p}_1, \ldots, \vec{p}_n = \vec{p}'$$

such that for $1 \leq i < n$, the points $\vec{p}_i$ and $\vec{p}_{i+1}$ are neighbors. Note that all optimal point sets are connected.

A further property of finite point sets is whether they contain cavities (see Figure 2.4). A *cavity in a point set* P is a k-tuple of points $(\vec{p}_1, \ldots, \vec{p}_k)$ such that

$$\exists \vec{v} \in NV \; \forall 1 \leq j < k : ((\vec{p}_{j+1} - \vec{p}_j) = \vec{v}),$$

$$\{\vec{p}_1, \vec{p}_k\} \in P,$$

and

$$\forall 1 < j < k : \vec{p}_j \notin P.$$

Point sets without cavities are called *cavity-free*.

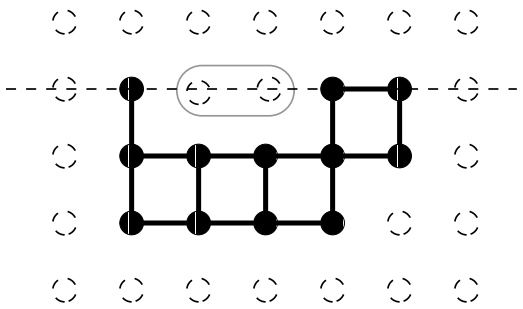


Figure 2.4: The figure shows a layer of the cubic (or FCC) lattice, which has one cavity (grey oval).

Furthermore, we define the surface of a set of points P. A *surface pair* of P is a pair of neighbors $(\vec{p}, \vec{p}')$, where $\vec{p} \in P$, and $\vec{p}' \notin P$. The *surface of* P, written surface(P), is the number of surface pairs of P, i.e.

$$\text{surface}(P) = |\{(\vec{p}, \vec{p}') \text{ surface pair of } P\}|.$$

There is a notable relation between the surface and the number of contacts in a point set P in a lattice, namely

$$|NV| \cdot |P| = 2 \cdot \text{contacts}(P) + \text{surface}(P). \tag{2.4}$$

This equation holds since each of the $|NV|$ many neighbors of the $|P|$ many points in $p \in P$ forms either a contact or a surface pair with p. Counting in this way, contacts occur twice and surface pairs once.
The definition of layers and ξ-layers of lattices can be extended to layers and ξ-layers of point sets straightforwardly by intersection with the point set. The layers of a finite point set P are called *finite layers* of P.

We will utilize ways to characterize a connected finite point set P in either the cubic lattice or the FCC lattice. Therefore, we partition a point set into its (non-empty) ξ-layers, for $\xi \in \{x, y, z\}$. Since P is connected, there is a consecutive sequence of such (non-empty) layers, i.e. layer $\xi = c + 1, \ldots$, layer $\xi = c + k$ for some $c, k \in \mathbb{Z}$ (cf. properties of cubic lattice and FCC lattice). We denote these layers of P by $P_1, \ldots, P_k$. $P_1, \ldots, P_k$ is called the *ξ-layer decomposition* of the point set P.
We call the sequence $|P_1|, \ldots, |P_k|$ the *number sequence of P in dimension ξ*. The *size of a number sequence* $(n_1, \ldots, n_k)$ is

$$\text{size}((n_1, \ldots, n_k)) = \sum_{i=1}^{k} n_i.$$

For a finite ξ-layer of a point set P of either the cubic lattice or the FCC lattice, we introduce the notions of frame and occupied lines. We recall that in both lattices the ξ-layers form square lattices. The *frame* of a ξ-layer is the minimal rectangle that surrounds all points in the layer and is oriented in parallel to the connections of neighbors in the layer. For an x-layer in the cubic lattice, these connections are paralell to the neighbor vectors $\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. Temporarily, we call these vectors *layer vectors*. For an x-layer in the FCC lattice the layer vectors are $\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$. We define the *width (resp. height) of the frame* as the number of lines along the first (resp. second) layer vector that intersect with lattice points and intersect the frame. Furthermore, the lines along a layer vector that intersect with the x-layer are called *occupied lines*. For a finite x-layer Q of P, we define the function occlines by occlines$(Q) = (a, b)$, where a (resp. b) is the number of occupied lines of Q along the first (resp. second) layer vector. Then, for the x-layer decomposition $P_1, \ldots, P_k$ of P, the sequence $(|P_1|, a_1, b_1), \ldots, (|P_k|, a_k, b_k)$, where $(a_i, b_i) = \text{occlines}(P_i)$, is called the *frame sequence of P in dimension x*. Analogously, one defines frame sequences in the dimensions y and z. The *size of a frame sequence* $((n_1, a_1, b_1), \ldots, (n_k, a_k, b_k))$ is

$$\text{size}(((n_1, a_1, b_1), \ldots, (n_k, a_k, b_k))) = \sum_{i=1}^{k} n_i.$$

The size of a frame sequence s should not be confused with its length $|s|$.
We note that width and height of the frame are always greater or equal than the numbers of occupied lines. They are equal if and only if the frame

contains no lines along layer vectors that intersect the lattice but do not intersect the ξ-layer of P.
For a ξ-layer decomposition $P_1, \ldots, P_k$ of a point set P, it is reasonable to distinguish between contacts within one of the layers P_i and contacts between points in successive layers. A contact between $\vec{p}$ and $\vec{p}'$ that are elements of the same layer P_i $(1 \leq i \leq k)$ is called *layer contact*. A contact between points $\vec{p}$ and $\vec{p}'$ in successive layers, i.e. $\vec{p} \in P_i$ and $\vec{p}' \in P_{i+1}$ $(1 \leq i < k)$, is called *interlayer contact*.

2.2 The HP-Model

The HP-model is a protein model that abstracts from real proteins in two important ways.

1. Instead of modeling the positions of all atoms of the protein, it models only the backbone structure of the protein, i.e. one position for each amino acid. Furthermore, these positions are constrained to points of a lattice.

2. Only the hydrophobic interaction between the amino acids is modeled, therefore the model distinguishes only two kinds of amino acids, namely hydrophobic (H) and polar (P).

2.2.1 Definition

We will now define the HP-model for a fixed lattice L. An *HP-sequence* is a protein sequence that is a word of the alphabet $\Sigma = \{H, P\}$. By seq_i, we denote the i-th element of the HP-sequence seq.
We define a *structure str of a sequence seq* as a tuple

$$str \in L^{|seq|}$$

such that

$$\forall 1 \leq i < |seq| : str_i \text{ and } str_{i+1} \text{ are neighbors, and} \tag{2.5}$$
$$\forall i \neq j : str_i \neq str_j. \tag{2.6}$$

In both, in a sequence and in a structure, we refer to the amino acids as *monomers*. We distinguish *H-monomers* and *P-monomers* as specified by the sequence.

Condition (2.5) is known as the *chain constraint* and claims that monomers that are successive in the chain are neighbored in the lattice. Condition (2.6) is denoted *self-avoiding constraint*. It claims that no lattice position is occupied twice by the structure. Thus, a structure is a self-avoiding walk on the lattice L (cf. [MS96]).
Given a structure *str* of a sequence *seq*, the *number of HH-contacts*, i.e. the number of contacts between H-monomers in *str*, is defined as

$$\text{HH-contacts}(seq, str) = \left| \left\{ (i,j) \;\middle|\; \begin{array}{c} 1 \leq i < j \leq |seq|, \\ seq_i = H,\ seq_j = H, \text{and} \\ str_i \text{ and } str_j \text{ are neighbors} \end{array} \right\} \right|.$$

The *HP energy function* is a function that assigns an energy value in $\mathbb{Z}$ to pairs (seq, str). This energy value, which is termed the *HP-energy of seq and str*, is defined in terms of contacts as

$$\text{HP-energy}(seq, str) = -\,\text{HH-contacts}(seq, str) + \text{HH-chain}(seq), \qquad (2.7)$$

where HH-chain(seq) is the number of HH-contacts in an elongated chain of *seq*. More formally, HH-chain(seq) is defined as the number of pairs $(i, i+1)$, for $1 \leq i < |seq|$, where $seq_i = H$ and $seq_{i+1} = H$.
For a sequence *seq*, a structure *str* is called *native* if and only if its HP-energy is minimal among all structures of *seq*.
Formally, the *HP-model for a lattice L*, shortly called *L-HP-model*, is a triple consisting of the set of HP-sequences, the set of structures for L, and the HP energy function for L.

2.2.2 Properties

In the following, we fix a sequence *seq*. Then, the number HH-chain(seq) is also fix.
Our first observation is that in HP-models minimizing the HP-energy is equivalent to maximizing the number of HH-contacts due to Equation 2.7. In consequence, we can choose either view of the optimization problem of structure prediction in HP. Almost exclusively, we adopt the view of maximizing the number of HH-contacts for our discussions.
Furthermore, the number of HH-contacts in a structure *str* depends only on the location of H-monomers in *str* and in particular not on their order given by the sequence. This leads to the definition of a hydrophobic core.
The *hydrophobic core of a structure str* is defined as the set of positions occupied by an H-monomer in *str*. The number of HH-contacts of *seq* and

str is equal to the number of contacts in the hydrophobic core C of *str*, i.e.

$$\text{HH-contacts}(seq, str) = \text{contacts}(C).$$

The concept of a hydrophobic core is of interest, since all known efficient algorithms for exact structure prediction in HP-models and HP-type models base on this notion [YD93, YD95, Bac98b, BWBB99]. Hydrophobic cores are also central for our approach.

2.3 Constraint Programming

Constraint programming (CP) means programming using constraints as a language construct. Here, a constraint is a condition which is satisfied by all solutions to our problem. That is, a constraint is a piece of syntax, more precisely an atom as it is known from mathematical logic, with the semantics of a relation between its variables. Some examples of constraints are $x \leq y$, $x + y \doteq z$, or $\vec{p}$ and $\vec{p}'$ are neighbors.
CP consists of a set of techniques for modeling problems by constraints on the solutions of the problem and for solving the modeled problems. Most notably, using constraints allows to draw conclusions from only partially specified data. A main application area of constraint programming techniques is solving NP-complete combinatorial problems. There, reasoning about incomplete data can prune the search for solutions by detecting inconsistencies of partial solutions and propagating partial information in order to gain (explicit) knowledge about the solutions.

2.3.1 Constraint Satisfaction Problems

In CP, problems are represented in the form of constraint satisfaction problems. A *finite constraint satisfaction problem (CSP)* P consists of

- a set of *variables* $\mathcal{X} = \{\mathtt{X}_1, \ldots, \mathtt{X}_n\}$,
- a set of finite domains $\mathcal{D} = \{D_1, \ldots, D_n\}$, where D_i is the set of possible values (i.e., the domain) of the variable $\mathtt{X}_i$.
- a set of constraints $\mathcal{C}$ defining relations between the variables $\mathcal{X}$.

We write $\text{dom}(\mathtt{X})$ to denote the domain in $\mathcal{D}$ which is associated to the variable $\mathtt{X} \in \mathcal{X}$.

Each constraint C is imposed on a tuple of variables in $\mathcal{X}$. For the constraint C, we denote this tuple by $X(C)$. We may write $C(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ for denoting that C is imposed on the variables $X(C) = (\mathtt{X}_1, \ldots, \mathtt{X}_n)$. A constraint C, where $X(C)$ is a n-tuple, is called *n-ary*.
An *assignment* of the variables $\mathcal{X}$ is a function $A : \mathcal{X} \to \bigcup \mathcal{D}$, where $A(\mathtt{X}) \in \mathrm{dom}(\mathtt{X})$. A constraint C on the tuple $X(C) = (\mathtt{X}_1, \ldots, \mathtt{X}_n)$ of variables is interpreted as a sub-set $\mathrm{T}(C)$ of the Cartesian product $\mathrm{dom}(\mathtt{X}_1) \times \cdots \times \mathrm{dom}(\mathtt{X}_n)$. We call such a constraint C *satisfied by an assignment A* if and only if $(A(\mathtt{X}_1), \ldots, A(\mathtt{X}_n)) \in \mathrm{T}(C)$. Note that, in consequence, a constraint can be defined semantically by specifying its set of tuples $\mathrm{T}(C)$.
A *solution* S of a CSP is an assignment of the variables $\mathcal{X}$ that satisfies all constraints in $\mathcal{C}$.
Given a CSP, one asks two different kinds of questions. First, is there a solution to the CSP and second, how many solutions are there. Usually these questions are answered constructively and then are equivalent to asking for one solution of the CSP or all solutions of the CSP, respectively.
There is always a naive algorithm for these problems, namely the brute force approach, which enumerates all assignments of values to variables and checks each for satisfaction of the constraints. This approach is known as generate-and-test. Needless to say that this approach is inefficient and thus is in many practical applications completely useless.
A much more promising strategy is known as the generate-and-constrain approach. This method employs reasoning over constraints, where the basis of such reasoning is consistency.

2.3.2 Consistency for Solving CSPs

In constraint programming, the common approach to solve a CSP is to combine enumeration and making the CSP consistent. In general, a CSP is consistent if and only if the domains of the variables do not conflict with the constraints of the CSP. However, there are different kinds and levels of consistency, which define the meaning of "conflict" in the previous sentence in different ways. A complete discussion of consistency can not be the aim of this work.
Making a CSP consistent means to transform a CSP P into a consistent CSP P' that is equivalent, i.e. which has the same set of solutions. This transformation is always monotonic. This means that the domain of each variable in P' is a sub-set of the corresponding domain in P or equal to it. The narrowing of the domains from P to P' is known as *constraint*

propagation.
In general, constraint propagation is a computational service of the *constraint solver*, which forms a part of the run-time environment of a constraint programming system. In our implementation system Oz/Mozart, this computational work is done by *constraint propagators*, which work as program threads that access the constraint store. The *constraint store* consists of the domains of the variables. Different propagators communicate only via the constraint store.
We will only discuss one kind of consistency in more detail, which is usually termed arc consistency or hyper-arc consistency. Therefore, we continue our discussion of the CSP P from the previous sub-section.
Shortly, a tuple $\tau = \tau_1, \ldots, \tau_n$ is *consistent with variables* $\mathtt{X}_1, \ldots, \mathtt{X}_n$ if and only if $\tau_j \in \mathrm{dom}(\mathtt{X}_j)$ holds for all $1 \leq j \leq n$.
For an n-ary constraint $C(\mathtt{X}_1, \ldots, \mathtt{X}_n)$, we call $a \in \mathrm{dom}(\mathtt{X})$ *consistent with* C, if and only if either $\mathtt{X} \notin X(C)$ or $\mathtt{X}$ is the ith variable of C, namely $\mathtt{X}_i$, and there exists a $\tau \in \mathrm{T}(C)$ such that $a = \tau_i$. Note that each $\tau \in \mathrm{T}(C)$ is consistent with $\mathtt{X}_1, \ldots, \mathtt{X}_n$ already by the definition of $\mathrm{T}(C)$.
A constraint C is called *hyper-arc consistent* if and only if for all $x_i \in X(C)$, $\mathrm{dom}(x_i) \neq \emptyset$ and for all $a \in \mathrm{dom}(x_i)$ holds that $a \in \mathrm{dom}(x_i)$ is consistent with C. One uses the term *arc consistent* for denoting the special kind of hyper-arc consistency of 2-ary constraints.
In general, a CSP is called consistent if and only if all its constraints are consistent.

2.3.3 Constraint Optimization: Branch-and-Bound

Branch-and-bound is the standard method for optimization using constraints. This method solves problems, known as constraint optimization problems. A *constraint optimization problem (COP)* P is a tuple

$$P = (\mathcal{X} = \{x_1, \ldots, x_n\}, \mathcal{D} = \{D_1, \ldots, D_n\}, \mathcal{C}, f)$$

consisting of a constraint satisfaction problem $P' = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ and a weight function $f : D_1 \times \cdots \times D_n \to \mathbb{Z}$, which evaluates solutions of P' (objective function). A solution S of P is a solution of P' that has maximal weight $f(S(x_1), \ldots, S(x_n))$, written shortly $f(S)$, of all solutions of P'.
The *branch-and-bound* method works by solving a series of constraint satisfaction problems. First, the CSP $P_0 = P'$ is solved. If P_0 has no solution, then there is no solution to the COP P. Otherwise we obtain the solution S_0. Starting with $i = 0$ we apply the following step. From the CSP P_i, a

CSP P_{i+1} is generated by adding the constraint $f(x_1, \ldots, x_n) > f(S_i)$. In consequence, when we now solve the CSP P_{i+1}, we get only solutions that are better than S_i with respect to f. This step is applied repeatedly until for some $k \geq 1$ the CSP P_k has no solution. Then, the solution to P_{k-1} is known to be a solution of the COP P.
The classical branch-and-bound method is not the only solving strategy for a COP P (as given previously). The existence of a good upper bound $\bar{w}$ on the weight of solutions of P' paves the way for an alternative approach. There, we define a series of CSPs $(P_w)_{w \in \mathbb{Z}}$. A CSP P_w consists of the CSP P' with the additional constraint $f(S_i) = w$. Then, starting with $w = \bar{w}$, we iteratively investigate CSPs P_w, while decreasing w by one, until a solution of P_w is found. Our approach uses the latter strategy.

Chapter 3

Overview of Constraint-Based Structure Prediction

This chapter gives an overview on the constraint-based structure prediction approach that is developed in this thesis. The Sections 3.2, 3.3, and 3.4 shortly introduce the objects and main results of the single steps of constraint-based structure prediction. These sections correspond to Chapters 4, 5, and 6, respectively. The corresponding chapters discuss the steps and develop the results in detail. Each of the chapters can be read independently from the overview in this chapter. However, the overview, helps to integrate the single steps into the whole approach.

3.1 Constraint Models for Structure Prediction in HP-Models

We start by describing a straightforward constraint model for the protein structure prediction problem. Afterwards, we describe our improved constraint approach.

3.1.1 A First Constraint Model

The following approach is applicable to the HP-model in arbitrary lattices. For simplicity, we introduce the formal model for the cubic lattice only. The handling of other lattices, e.g. the face-centered cubic lattice, is analogous. Note in particular that every lattice has an integral representation. This allows to use the integer finite domain constraint system for any lattice.
We can encode the space of all possible conformations for a given sequence as a constraint problem. When we give this encoding in the following, please

note that all constraints can be encoded using the constraint system over finite integer domains, Boolean constraints, and reified constraints. If ϕ denotes an arbitrary constraint, then a *reified constraint* is a constraint $\mathtt{x} \leftrightarrow \phi$ with the semantic that the Boolean variable $\mathtt{x}$ is 1 if and only if ϕ holds, i.e. $\mathtt{x}$ reifies the truth of ϕ. An example of a reified constraint is given later by the constraint (3.1). Operationally, a reified constraint is propagated by setting x to 1 if the constraint store entails ϕ, and to 0 if the constraint store disentails ϕ. For a constraint ϕ to be *entailed* by the constraint store, ϕ must be satisfied by every valuation that satisfies the constraint store. We use also entailment constraints of the form $\phi \rightarrow \psi$, which are interpreted as follows. If a constraint store entails ϕ, then ψ is added to the constraint store. The constraint model can be directly implemented using the language Oz [Smo95], since this programming system supports finite domain variables, Boolean constraints, reified constraints, entailment and programmable, encapsulated search.

For the actual constraint model, we introduce new variables $\mathtt{X}_i$, $\mathtt{Y}_i$ and $\mathtt{Z}_i$ for every monomer i, which denote the x-, y-, and z-coordinate of $c(i)$. Since we are using a cubic lattice, we know that these coordinates are all integers. The domains are also finite, since we can restrict the possible values of these variables to $[\,1\,..\,2|seq|\,]$.[1] This is expressed by introducing the constraints

$$\mathtt{X}_i \in [\,1\,..\,2|seq|\,] \wedge \mathtt{Y}_i \in [\,1\,..\,2|seq|\,] \wedge \mathtt{Z}_i \in [\,1\,..\,2|seq|\,]$$

for every $1 \leq i \leq n$. The excluded-volume constraint is just given for $i \neq j$ by

$$(\mathtt{X}_i, \mathtt{Y}_i, \mathtt{Z}_i) \neq (\mathtt{X}_j, \mathtt{Y}_j, \mathtt{Z}_j).^{2}$$

For expressing that two successive monomers have unit distance, we introduce for every monomer i with $1 \leq i < |seq|$ three finite domain variables $\mathtt{Xnext}_i$, $\mathtt{Ynext}_i$, and $\mathtt{Znext}_i$. Then, we can express the unit-vector distance constraint by

$$\begin{array}{llll} \mathtt{Xnext}_i & = & |\mathtt{X}_i - \mathtt{X}_{i+1}| & \qquad \mathtt{Znext}_i \;=\; |\mathtt{Z}_i - \mathtt{Z}_{i+1}| \\ \mathtt{Ynext}_i & = & |\mathtt{Y}_i - \mathtt{Y}_{i+1}| & \qquad \mathtt{Xnext}_i + \mathtt{Ynext}_i + \mathtt{Znext}_i \;=\; 1. \end{array}$$

The constraints that are described above define the space of all possible conformations. Every valuation of $\mathtt{X}_i, \mathtt{Y}_i, \mathtt{Z}_i$ that satisfies the above constraints

[1] We even could have used the domain $[1..n]$. However, the domain $[\,1\,..\,2|seq|\,]$ is more flexible since we can assign an arbitrary monomer the vector (n, n, n), and still have the possibility to represent all possible conformations.

[2] This cannot be directly encoded in Oz [Smo95], but we reduce these constraints to difference constraints on integers.

is an *admissible* conformation for the sequence *seq*, i.e. a self-avoiding walk of length $|seq|$.
The simplest way to search for conformations with maximal number of contacts is to add constraints for counting this number. Then, one can directly enumerate the variables $\mathtt{X}_i$, $\mathtt{Y}_i$ and $\mathtt{Z}_i$. For HP-type models, we have to count contacts that are formed by two neighboring H-monomers. For this purpose, one introduces a variable $\mathtt{Contact}_{i,j}$ for $1 \leq i < j \leq |s|$ that is 1 if i and j have a contact in every conformation that is compatible with the valuations of $\mathtt{X}_i, \mathtt{Y}_i, \mathtt{Z}_i$, and 0 otherwise. Then, for $1 \leq i < j \leq |s|$, we introduce new FD-variables $\mathtt{Xdiff}_{i,j}$, $\mathtt{Ydiff}_{i,j}$, and $\mathtt{Zdiff}_{i,j}$ and constrain them by

$$\begin{aligned} \mathtt{Xdiff}_{i,j} &= |\mathtt{X}_i - \mathtt{X}_j| \qquad & \mathtt{Zdiff}_{i,j} &= |\mathtt{Z}_i - \mathtt{Z}_j| \\ \mathtt{Ydiff}_{i,j} &= |\mathtt{Y}_i - \mathtt{Y}_j| \qquad & \mathtt{Contact}_{i,j} &\in \{0,1\} \end{aligned}$$

$$\mathtt{Contact}_{i,j} \leftrightarrow (\mathtt{Xdiff}_{i,j} + \mathtt{Ydiff}_{i,j} + \mathtt{Zdiff}_{i,j} = 1). \tag{3.1}$$

The variable `HHContacts` counts the number of contacts between H-monomers, and is defined by

$$\mathtt{HHContacts} = \sum_{\substack{i<j \\ s_i=s_j=H}} \mathtt{Contact}_{i,j}. \tag{3.2}$$

Now, we apply constraint-based branch-and-bound enumeration on the variables $\mathtt{X}_i$, $\mathtt{Y}_i$, and $\mathtt{Z}_i$, thereby searching for a conformation with maximal number of contacts.
Figure 3.1 shows a flowchart of the structure prediction approach that uses this constraint model.

3.1.2 Improved approach

When we use the previously described approach alone, the search space will be restricted only poorly.[3] We want to give some explanation for the unsatisfying performance. For this approach, an efficient constraint solver is too weak in detecting whether a partial structure has the potential to achieve sufficiently many contacts. For improving this ability, one needs to introduce additional constraints to get bounds on the number of contacts. However, we

[3]Nevertheless, the above formulation is general enough to be used on arbitrary lattices and to be extended for various energy functions. For example, it can be used for lattice models with an extended alphabet like the HPNX-model [BWBB99], which models also electrostatic contacts in addition to hydrophobicity.

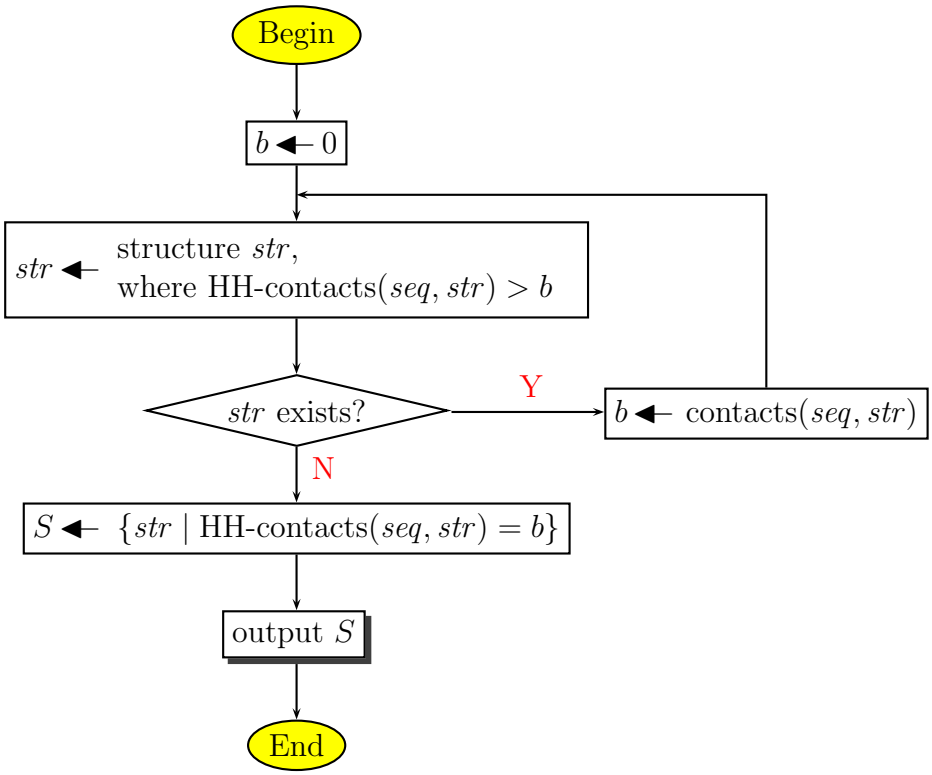


Figure 3.1: Predicting all native structures of a HP-sequence *seq* by the first constraint model (branch-and-bound). Output of the algorithm is the set of optimal structures S of *seq*.

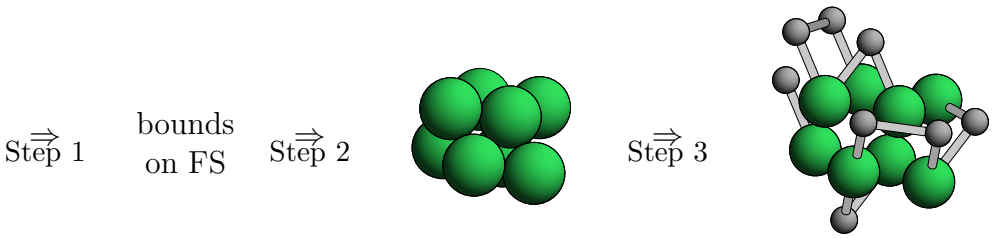


Figure 3.2: The improved constraint-based structure prediction approach. Step 1: bounding of frame sequences. Step 2: core construction. Step 3: threading.

do not see how to define good bounds using the previous approach. Furthermore, in order to apply branch-and-bound optimization, one needs a search heuristic that prefers low-energy conformations. Again, it seems unlikely to find such a heuristic.

Consequently, for strongly improving over the previous constraint model, we can not simply extend this model. Instead, we develop a completely new approach. We introduce a three step approach, which restricts the search space by the use of strong bounds on the number of contacts. This comes at the cost of developing lattice specific bounds on the number of contacts.

In two pre-computation steps, we restrict the set of all conformations to a subset that provably contains all minimal energy conformations. For this purpose, we apply a technique that has been employed first by the CHCC algorithm [YD93, YD95]. There, one calculates the set of positions that is occupied by H-monomers (hydrophobic core) first. The number of contacts of a hydrophobic core C is defined for general point sets in Chapter 2.[4]

Since we cannot guarantee in advance that a given sequence fits to the most compact hydrophobic cores, one has to consider also less compact hydrophobic cores in a systematic way (where generating the cores is a problem by its own). Luckily, the number of cores that have to be investigated for a sequence is usually feasible. Furthermore, the cores can be pre-calculated independently of a specific sequence.

For each core, we search for a conformation of the considered sequence, where the H-monomers form exactly this core. Consequently, such a conformation has exactly as many HH-contacts as there are contacts of the core. This process is called *threading* of the sequence to the core. We give an efficient constraint formulation to solve the threading problem in Section 3.4.

In order to provably find a minimal energy conformation, we start by threading on maximally compact cores of proper size[5], and then iteratively search for solutions on the next best cores, until a solution is found.

The remaining problem is generating the hydrophobic cores. Without any additional efforts, we would have a huge search space again, since nothing is known about the exact shape of the hydrophobic core in advance. Hence, one has to find again some restriction on the search space that allows us to calculate bounds and to apply constraint-based optimization. The first idea for restricting the hydrophobic core is to define the surrounding cuboid that contains the hydrophobic core. If one has a very tight cuboid, then

[4]Throughout this chapter, we will use the term hydrophobic core also for (optimal) point sets in general, due to their intended use.

[5]Obviously, the size of the core has to fit the number of H-monomers in the sequence.

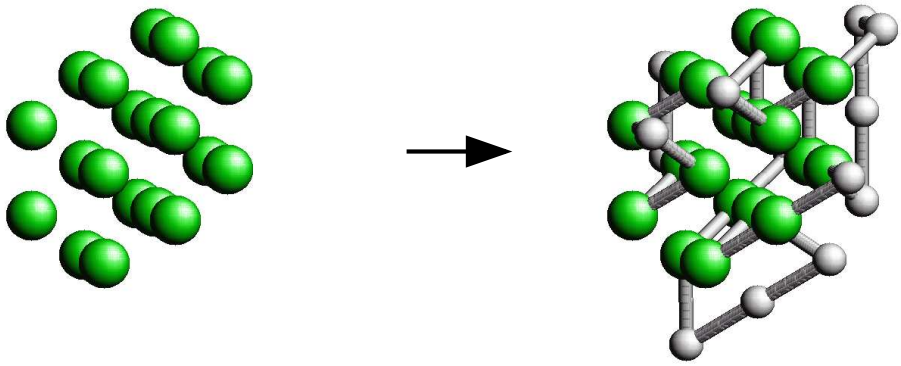


Figure 3.3: Threading of the HP-sequence 'HHPPPPHHHHPHHHPHHH-PPHHPHHPPPHHHPH' to a core in the FCC lattice. The figure shows the core (on the left) and the resulting structure (on the right), where H-monomers are shown as black beads and P-monomers as white ones. Note that the core-size equals the number of H-monomers in the sequence, namely 20.

the hydrophobic core in this cuboid must be rather compact. This claim obviously holds for the cubic lattice and is also of some use for the FCC.

However, this approach is not fine-grained enough for the FCC as well as for sub-optimal hydrophobic cores in the cubic lattice. We introduce therefore a more precise boundary for the hydrophobic core, which is obtained by splitting the lattice into layers. We recall that an x-layer is just a plane that is orthogonal to dimension x. For each such x-layer, we define the *frame* to be the minimal rectangle around all positions of the core in this layer. The corresponding *frame sequence* consists of the height and width of each frame in each x-layer, together with the number of H-monomers in this layer (see Figure 3.4). Please note that the exact position of the frames is *not* part of the frame sequence. To each frame sequence, we associate a bound which is an upper bound on the number of contacts in every hydrophobic core that has this frame sequence.

For enumerating all hydrophobic cores of size n with c contacts, we perform a constraint-based search, which is strongly restricted by the frame sequences for n H-monomers with a bound of at least c contacts.

Figure 3.2 sketches the three steps of our improved approach. The flowchart of Figure 3.5 describes our strategy for predicting all native structures of

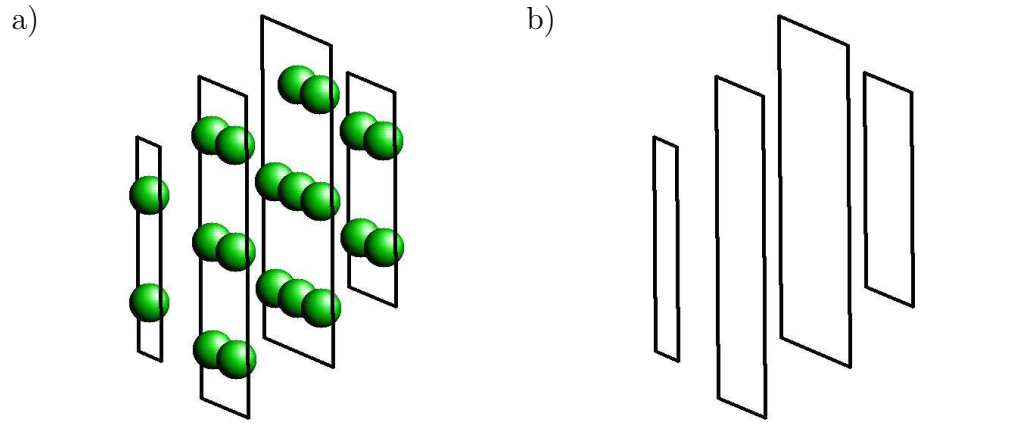


Figure 3.4: Hydrophobic cores and frame sequences. a) a hydrophobic core with frames. b) the corresponding frame sequence. a_i is the width and b_i is the height of the frame in the i-th layer. And n_i is the number of H-monomers in this layer.

a given sequence (cf. Figure 3.1). The flowchart integrates the three steps bounding, core construction, and threading.
The highly lattice-specific bounds on contacts will be overviewed in a section of their own. After the description of the bounds, we will describe the construction of the hydrophobic cores, and finally describe the threading method. Note that the order of our description follows the structure of the total prediction algorithm. Each of the following three sections corresponds to one chapter of the thesis.

3.2 An Upper Bound for Frame Sequences

As prerequisite for the enumeration of hydrophobic cores, we investigate the problem of generating the set of all frame sequences for a given number of points with a bound of at least c contacts.
The first step is to define the upper bound on contacts for a given frame sequence $(a_1, b_1, n_1) \ldots (a_l, b_l, n_l)$, which is discussed separately for the cubic lattice and the FCC lattice. We will start with the less complex case of the cubic lattice. Note that for the cubic lattice, there exists a previous bound on contacts by Yue and Dill [YD93, YD95]. However, we present our new bound for several reasons. First, the bound of Yue and Dill is incomplete as we will discuss in Chapter 4. Second, we can improve over their bound

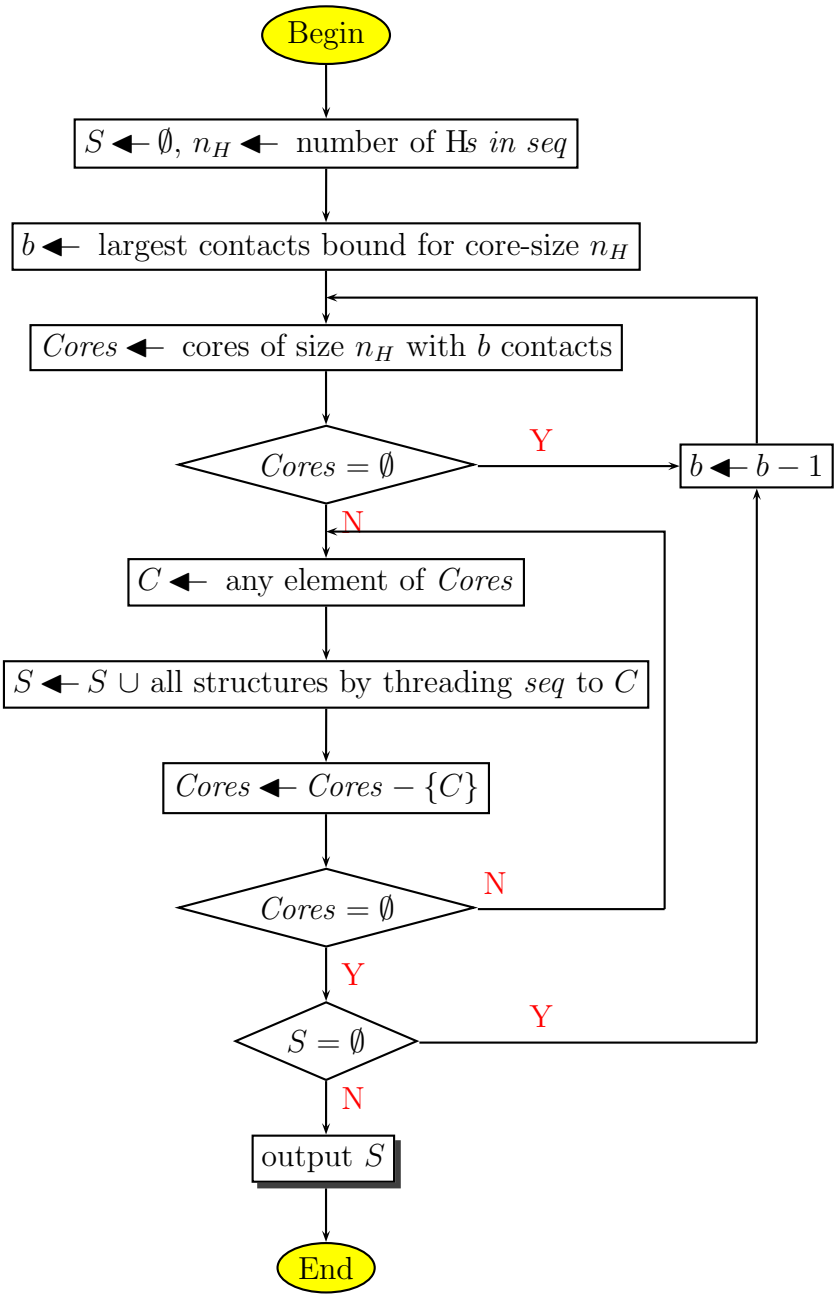
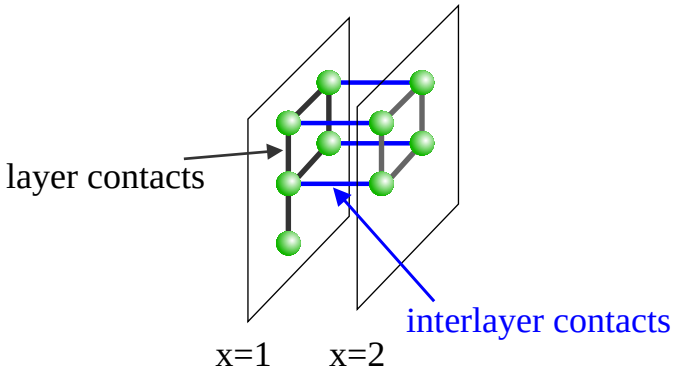


Figure 3.5: Predicting all native structures of a HP-sequence *seq* by the improved approach. The set S collects the optimal structures of *seq*. b is a bound on the contacts, which is finally the number of HH-contacts in the optimal structures. *Cores* always denotes the set of the cores that are the remaining candidates for structures of *seq* with b HH-contacts.

a)



b)

	Layer1	Layer2
	$a_1 = 2$	$a_2 = 2$
	$b_1 = 3$	$b_2 = 2$
	$n_1 = 5$	$n_2 = 4$

Figure 3.6: a) Layer and interlayer contacts b) Corresponding frame sequence

by investigating the distribution of H-monomers to layers. Third, the bound is instructive for understanding the more intricate case of the FCC lattice, since both bounds share a similar structure.

3.2.1 Frame Sequences in the Cubic Lattice

In Figure 3.6a, a hydrophobic core for the cubic lattice is shown, where we explicitly mark its two layers. Figure 3.6b gives the corresponding frame sequence. In Chapter 2, we introduced the terms layer contact and interlayer contact. An illustration of these terms, which describe two kinds of contacts, is provided by Figure 3.6a. The upper bound on the number of contacts in any core that satisfies the given frame sequence is defined as the sum of separate bounds for the number of layer *and* interlayer contacts.

In order to bound the layer contacts in the cubic lattice, we employ the concept of surface, which was used by [YD93, YD95] before. The surface of a core is defined via surface pairs in Chapter 2.

Now, imagine a single layer $x = k$ of the lattice that intersects the core. The *layer surface* of a hydrophobic core C in layer $x = k$ is the number of surface pairs of C, where both positions are in the (infinite) layer $x = k$.

Assume that there are n core positions in this layer $x = k$ and these positions are contained in a minimal rectangle of size $a \times b$, which was called the *frame of the layer*. Then, since every core position has four neighbors in the same layer that are either occupied by the core (then contributing to the number of contacts) or not (then contributing to the surface), the surface and layer

contacts are related via the equation

$$4 \cdot n = 2 \cdot \text{Contacts} + \text{Surface}. \tag{3.3}$$

Hence, minimizing the surface maximizes the number of contacts. Yue and Dill [YD93, YD95] observed for the cubic lattice that it is easier to minimize the surface instead of directly maximizing the number of layer contacts. In particular, if there are no cavities in the core (cf. Figure 2.4 in Chapter 2), then the layer surface is given by $2 \cdot (a + b)$ (compare Figure 3.7). Thus, $2 \cdot (a + b)$ is a lower bound on the surface. Using this in Equation 3.3 yields an upper bound on the number of contacts from a, b, and n.

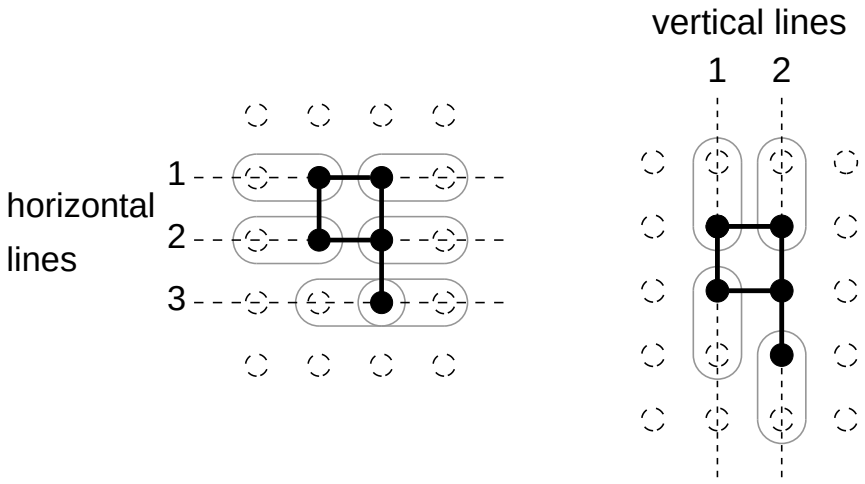


Figure 3.7: Horizontal and vertical surface. Every horizontal and vertical line through the hydrophobic core produces at least two surface pairs (or exactly two surface pairs, if there are no cavities in the core). The grey ovals mark the pairs of surface points and corresponding core positions.

For the cubic lattice, there is a straightforward upper bound on the number of interlayer contacts. Given two successive layers $x = k$ and $x = k + 1$, every position in layer $x = k$ has exactly one neighbor position in $x = k + 1$ and vice versa. Hence, there can be at most n_k and at most n_{k+1} interlayer contacts between $x = k$ and $x = k+1$. That is, for two successive layers with n_k and n_{k+1} core positions, there are at most $\min(n_i, n_{i+1})$ many interlayer contacts.

3.2.2 Frame Sequences in the FCC Lattice

Our key to bounds for the FCC lattice is partitioning the face-centered cubic lattice into layers that each form a square lattice (as in the cubic lattice). In the FCC, these layers are arranged such that every point in one layer has four neighbors in the next layer (cf. Figure 1.5). Note that due to the partitioning, the definitions of layer sequences as well as layer and interlayer contacts from the cubic lattice apply also for the FCC. Furthermore, we can use the same bound for the layer contacts as in the case of the cubic lattice. For the interlayer contacts in the face-centered cubic lattice, the situation is more intricate as in case of the cubic lattice, since every position in a layer can form a contact with up to 4 neighbors in the next layer.
The key problem for bounding the total number of interlayer contacts is the bounding of interlayer contacts between two successive layers $x = k$ and $x = k + 1$. Obviously, it is infeasible to search through all possible pairs of layers that satisfy the parameters a_k, b_k, n_k and $a_{k+1}, b_{k+1}, n_{k+1}$ in order to obtain a tight bound.
However, imagine that we know the distribution of monomers in the layer $x = k$. Then, we can count how many points in the layer $x = k+1$ form $1, 2, 3$ and 4 contacts to the first layer. Formally, we define a position $\vec{p}$ in layer $x = k+1$ to be an *i-point for the core C in layer $x = k$* (with $i = 1, 2, 3$ or 4) if $\vec{p}$ has i neighbors that are contained in layer $x = k$ and C (see Figure 3.8). We get a bound on the number of interlayer contacts by distributing the n_{k+1} elements of the second layer to these i-points. There, we fill the positions greedily, starting with 4-points and continuing with decreasing i.
In Chapter 4, we argue that there is a relation between the frame $a \times b$, the number of elements n, and the numbers of i-points of a layer. Only three further parameters of the layer ($m_{\text{no}}, m_{\text{nt}}$, and m_x) are sufficient to determine the numbers of i-points exactly. m_{no} denotes the number of pairs of non overlapping succeeding lines, m_{nt} is the number of pairs of not touching succeeding lines, and m_x gives the number of x-steps in the layer.
We will derive that the numbers of i-points, written $\#i$ for $i = 1 \ldots 4$, are determined as

$$\begin{aligned}
\#4 &= n - \frac{1}{2}s + 1 + m_{\text{no}} \\
\#3 &= m_x - 2(m_{\text{no}} - m_{\text{nt}}) \\
\#2 &= s - 4 - 2\#3 - 3m_{\text{no}} - m_{\text{nt}} \\
\#1 &= \#3 + 2m_{\text{no}} + 2m_{\text{nt}} + 4
\end{aligned}$$

This relation is then simplified further and is related to the number of inter-

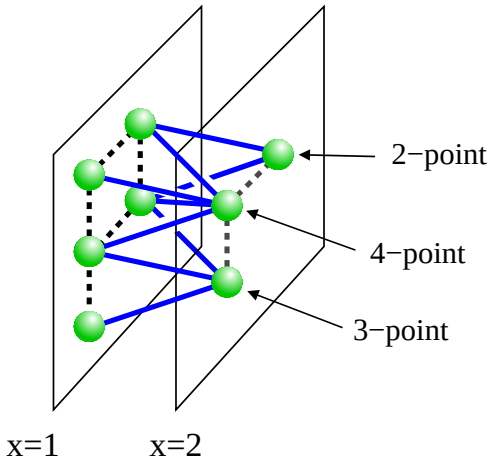


Figure 3.8: Definition of i-points

layer contacts. Finally, we can compute a bound on the number of interlayer contacts by only enumerating the possible values of m_{no}, instead of enumerating the exponentially many possible layers.

3.2.3 Generating Frame Sequence Sets

Now, we discuss the generation of a set of frame sequences for cores with given size n and a bound of at least c contacts. This generation is discussed for both lattices uniformly.

We start by computing a bound $\mathrm{B_C}(n, n_1, a_1, b_1)$ on the number of contacts in cores of size n and a first layer $x = 1$ that has n_1 elements and the frame $a_1 \times b_1$. This can be done efficiently for all n up to some upper limit and all n_1, a_1, b_1 at the same time using a dynamic programming (DP) algorithm. This algorithm fills a four-dimensional matrix for evaluating the recursion of Equation (3.4). Figure 3.9 provides an illustration of this recursion. We define two functions $\mathrm{B_{LC}}$ and $\mathrm{B_{ILC}}$, which denote the lattice specific bounds as they are described above. The values $\mathrm{B_{LC}}(n_1, a_1, b_1)$ (resp. $\mathrm{B_{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2)$) are the upper bound of the contacts on layer contacts for layers with parameters n_1, a_1, b_1 (resp. interlayer contacts between the two layers with parameters n_1, a_1, b_1 and n_2, a_2, b_2).

$$\mathrm{B_C}(n, n_1, a_1, b_1) = \max_{n_2, a_2, b_2} \left(\begin{array}{ll} & \mathrm{B_{LC}}(n_1, a_1, b_1) \\ + & \mathrm{B_{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2) \\ + & \mathrm{B_C}(n - n_1, n_2, a_2, b_2) \end{array} \right) \tag{3.4}$$

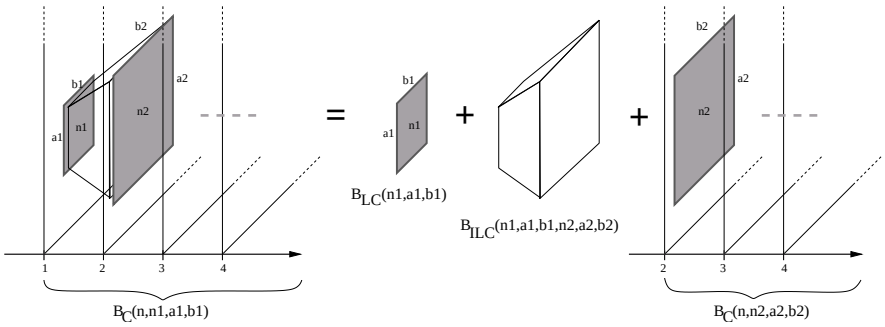


Figure 3.9: Illustration of Equation 3.4. Recursively, the equation reduces the bound for cores with first layer $x = 1$ to a bound of cores with first layer $x = 2$. The recursion equation abstracts from the continuation of the core in layers $x = 3, 4, \ldots$, which allows for efficient evaluation.

The frame sequence sets are generated by trace-back through the resulting four-dimensional matrix. Since we are interested in the frame sequences with at least a bound of c contacts, these sequences are not necessarily optimal. Note that we also generate these sub-optimal frame sequences from the DP-matrix, which is done by tolerating a limited deviation from the optimal path, when computing the trace-back.

3.3 Constructing the Hydrophobic Cores

In order to construct the hydrophobic cores of size n with at least c contacts, we use the corresponding complete set of frame sequences to restrict a constraint-based search.
Given the set of frame sequences, we know that each core must have one of these frame sequences. Otherwise, it could not form the required number of contacts c, due to our bound of the previous chapter. For the cubic lattice, it is furthermore straightforward that each core must have one of the frame sequences in every possible layer decomposition, i.e. either a decomposition along the x-axis into x-layers, one into y-layers, or one into z-layers. Now, for gaining the maximal information from the sequences also for FCC, one has to understand how the x-, y-, and z-layers are oriented to each other in the FCC-lattice. Figure 3.10 illustrates that the layers of different dimensions x, y, or z are orthogonal to each other as in the cubic lattice. However, in contrast to the cubic lattice, they can be imagined as being rotated by 45°. Due to this arrangement, we can apply the same constraint as for the

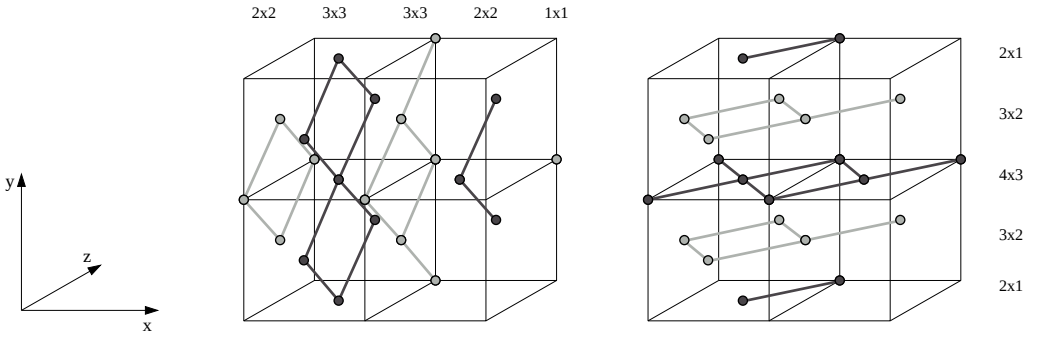


Figure 3.10: Two representations of a single hydrophobic core in the FCC lattice. The cores are embedded in a cubic structure to emphasize the building principle of this lattice as face-centered cubic. We have marked the x-layers (resp. y-layers) by showing their layer contacts. The use of light and dark ink emphasizes the layer structure.

cubic lattice. That is, one of the frame sequences must be satisfied in either dimension, also for the FCC lattice.
In order to enumerate the cores, we start by fixing the length of the frame sequence in every dimension. Since we search for connected cores, the lengthes immediately tell us the dimensions of a minimal surrounding cuboid that contains all points of the core and furthermore contains no empty layers.[6]
Note that not all combinations of frame sequence lengthes can be satisfied, which is however hard to detect at this stage of the search. Therefore, in case of the FCC lattice, we simply perform a complete enumeration of frame sequence lengthes. For the cubic lattice, we can restrict the enumeration of frame sequences further. The CHCC algorithm of [YD93, YD95] provides means to restrict the dimensions of the cuboid by an upper bound on the number of contacts. However, the original CHCC is incomplete for computing sub-optimal cores and we need to develop a new algorithm, which is based on CHCC.
As soon as the surrounding cuboid is fixed, we introduce boolean variables $\texttt{CorePos}_{\vec{p}}$ for every lattice point in the cuboid, which tell if the point belongs to the core. The variables $\texttt{CorePos}_{\vec{p}}$ are constrained to the (still partially known) frame dimensions and elements in each layer. Furthermore, the variables are constrained to the number of contacts in each layer and in total. For this aim, for each pair of points $\vec{p}, \vec{q}$, a boolean variable $\texttt{Contact}_{\vec{p},\vec{q}}$ is

[6]Note that we construct only connected cores; unconnected cores can still be composed from connected ones.

introduced. For example, the constraint for the total number of contacts is then easily expressed as

$$\sum \texttt{Contact}_{\vec{p},\vec{q}} \geq c.$$

Before enumerating the boolean variables directly, it is advantageous to introduce boolean variables for each line along the lattice vectors that intersects the cuboid and tells whether the points of the line are occupied. Then, these variables are enumerated first. Finally, constraints counting the overlapping and touching of lines as well as constraints relating the surface of layers and the whole core improve the constraint propagation during the search. Details of this approach are given in Chapter 5, which discusses the core construction problem for the FCC lattice.

3.4 Threading sequences to cores

The final problem is the threading of a given sequence to a hydrophobic core (see Figure 3.3), which yields the structures, where the H-monomers build the given hydrophobic core. This is discussed independently of the actual lattice. We define a *self-avoiding walk* as a sequence of lattice positions, where successive positions are lattice neighbors and no position occurs twice. Shortly, the *threading problem* asks for a self-avoiding walk, where all H-monomers are placed on core positions.
When given an HP-sequence *seq* of length n and a core C, we model the problem as CSP using the finite domain constraint system. We start by introducing finite domain variables $\texttt{X}_1 \ldots \texttt{X}_n$. The values of these variables are the positions of the corresponding monomers in the FCC lattice. Therefore, a valuation can encode a protein structure in our model. First, note that these variables have indeed finite domains. This is a consequence of the positions of H-monomers being in the finite core C and the P-monomers being connected to the H-monomers. Regarding the implementation, note that we can still use a standard finite domain constraint system with integer domains if we assign a unique number to each position.
The restriction of the H-monomers to core positions is now simply expressed by unary constraints

$$\texttt{X}_i \in C \text{ for } 1 \leq i \leq n,\ seq_i = H \tag{3.5}$$

The self-avoiding property of a structure means that all positions of monomers have to be different, which is directly expressed by an *all-different* constraint

on $\mathtt{X}_1 \ldots \mathtt{X}_n$. Hence, we introduce

$$\text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n). \tag{3.6}$$

Technically, we use the constraint of difference a la Régin [Reg94] for the H-monomers, which ensures hyper-arc-consistency, and a weaker propagating constraint for the P-monomers. Thus, we use the computationally expensive, complete all-different constraint only where it propagates most efficiently.
The walk property claims that successive monomers must occupy neighboring positions. For ensuring this property, we introduce the constraint $\text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$. We investigate now how we can guarantee hyper-arc consistency for this constraint. By a general result of Freuder [Fre82], arc consistency amounts to global consistency in a tree-structured network of binary constraints. We will use an instance of this result. Namely, that the n-ary walk constraint is hyper-arc consistent if and only if all 2-ary walk constraints $\text{Walk}(\mathtt{X}_i, \mathtt{X}_{i+1})$ are arc consistent.
We observed that the propagation is still rather weak if self-avoiding walks are modeled using the constraints $\text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ and $\text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$, which communicate only over the domains of the variables. To improve the propagation, we discuss the combined constraint

$$\text{SAWalk}(\mathtt{X}_1, \ldots, \mathtt{X}_n) = \text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n) \wedge \text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n).$$

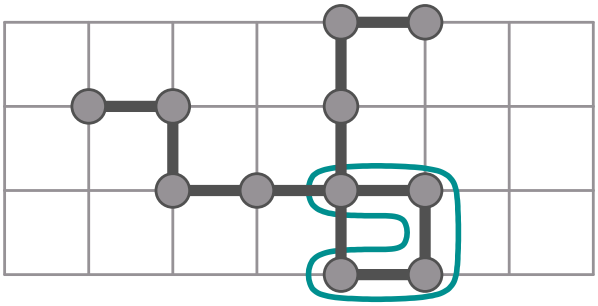


Figure 3.11: A walk that is not self-avoiding but 4-avoiding. Encircled is a sub-walk of length 4. Every sub-walk of length 4 is self-avoiding.

Since (very likely) there is no efficient propagator for the combined constraint, we investigate in Chapter 6 a relaxation of the self-avoiding walk constraint that provides better propagation but is still tractable. For variables

$\mathtt{X}_1, \ldots, \mathtt{X}_n$, the constraint SAWalk$(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ introduces the all-different constraint on all variables, which makes constraint propagation hard. Obviously, we can reduce the complexity if we enforce the all-different condition only for smaller subsets of the variables. It turned out that a reasonable choice is to guarantee the self-avoiding property only for each set of k successive variables. In order to formalize this, we introduce the concept of k-avoiding walks, which are walks that are locally self-avoiding for every sub-walk of length k (but not necessarily for the complete walk). Figure 3.11 shows a walk that is 4-avoiding, but neither 5-avoiding nor self-avoiding. The constraint Walk$[k](\mathtt{X}_1, \ldots, \mathtt{X}_n)$ is defined as constraining the variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$ to form a k-avoiding walk.

Chapter 4

Bounds on Contacts

In this chapter, we discuss the computation of upper bounds on the number of contacts in detail. As already discussed in Chapter 3, these bounds are used for the construction of compact hydrophobic cores.

A first section discusses bounds for the cubic lattice. We review the fundamental part of the CHCC-method of Yue and Dill [YD93, YD95]. Then, we investigate an improvement, which handles sub-optimally compact cores correctly. The bound on the number of contacts for the cubic lattice will not be discussed in more detail since this bound was already explained in sufficient detail in Chapter 3.

In the following sections, we deal with the much more intricate case of the FCC lattice. The bounds for the FCC are based on a decomposition of point sets into their x-layers. We give an upper bound of the number of contacts in a point set by its number sequence and one by its frame sequences. Please recall the terms *number sequence* and *frame sequence* from Chapter 2.

Section 4.3 reviews an upper bound for the number of contacts in the FCC lattice, which was originally developed in [Bac00b] and reviewed in more detail in [Bac04]. It defines a function $\mathrm{B}_{\mathrm{num}}$ such that any point set with the number sequence $n_1, \ldots, n_k$ has less or equal $\mathrm{B}_{\mathrm{num}}(n_1, \ldots, n_k)$ many contacts. The main contribution is a bound on the number of contacts between two succeeding layers (called interlayer contacts), where each of the two layers is connected. During the review of the number sequence bound, we introduce some definitions, which are fundamental for the number and the frame sequence bounds.

In the succeeding three sections, we develop a similar bound, which bounds point sets by their frame sequence, i.e. it defines a function B_{fr} such that any point set with frame sequence $(n_1, a_1, b_1), \ldots, (n_k, a_k, b_k)$ has less or equal than $\mathrm{B}_{\mathrm{fr}}((n_1, a_1, b_1), \ldots, (n_k, a_k, b_k))$ many contacts. Since arbitrary frames

(a, b) are allowed for the layers, a new bound on interlayer contacts has to be developed, where layers can be unconnected.
A last section of its own deals with the computation of

- the maximal number of contacts $c_{\max}(n)$ in any point set of size n, for a size n and
- the set of frame sequences FS, where $B_{\text{fr}}(FS) \geq c$, for a given size n and a number of contacts $c \leq c_{\max}(n)$.

The computations are done efficiently using dynamic programming. The implementation technique of lazy dynamic programming helps to improve run-time and space-consumption, and in the same time keeps the implementation clear and well structured. Most notably, this technique supports employing the bound on number sequences to help in computing frame sequences; thereby, we gain significant speed-up.

4.1 Constraint Hydrophobic Core Construction a la Yue and Dill

Yue and Dill [YD93, YD95] developed a structure prediction algorithm for the cubic HP-model, which is known as *constraint hydrophobic core construction (CHCC)*. A sub-problem of their work can be appropriately extended to constrain core construction in case of the cubic lattice.
In the cubic lattice, a cuboid is a set $C = [x_0 \dots x_1] \times [y_0 \dots y_1] \times [z_0 \dots z_1] \subset \mathbb{Z}^3$. The dimensions, i.e., the length, width, and height of the cuboid are $l = x_1 - x_0 + 1$, $w = y_1 - y_0 + 1$, and $h = z_1 - z_0 + 1$, respectively. The frame of a point set P with the smallest cuboid $C \supseteq P$ is the sorted list of the dimensions of the cuboid. By defining the frame as a sorted list instead of a tuple, two point sets with symmetrical smallest cuboid supersets have the same frame, which breaks the geometrical symmetries.
Using these terms, Yue and Dill calculate the frames of all optimal point sets of a given size, i.e. they compute for a size $n \in \mathbb{N}$ the set

$$\text{Frames}(c, n) = \{\text{frame of } P \mid P \text{ connected point set}, |P| = n, \\ \text{contacts}(P) \geq c\},$$

where $c \in \mathbb{N}$ is the maximal number of contacts in a point set of size n.
After describing the method of Yue and Dill, we will extend their approach for computing $\text{Frames}(c, n)$ for arbitrary values of $c \in \mathbb{N}$.

4.1.1 The CHCC Bound for Frames

The CHCC algorithm is based on a fundamental claim, namely that connected point sets of the cubic lattice with maximally many contacts can be reconfigured into connected point sets of a special form, while preserving the size, the number of contacts, and the frame.

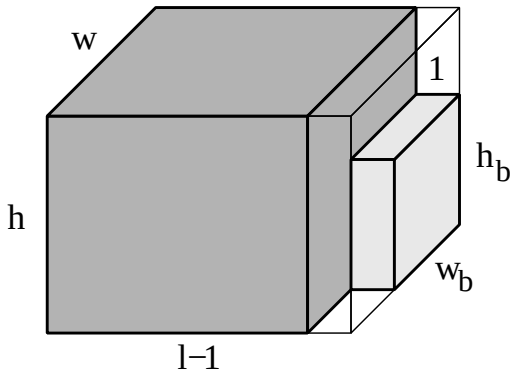


Figure 4.1: Special form for optimal point sets. The attached layer (light shade) need not fill the rectangle $h_b \times w_b$ completely.

A point set P is in this special form if and only if it is composed of a cuboid with dimensions $(l-1) \times w \times h$, where the frame of P is the sorted list of l,w, and h, and a single layer orthogonal to the x-axis.[1] The width and height of this layer are its dimensions perpendicular to the x-axis, they are denoted by w_b and h_b, respectively.
For a point set in this special form the number of contacts is calculated from the parameters l, w, h, w_b, and h_b via its relation to the surface as

$$\text{contacts}(l, w, h, w_b, h_b) = \frac{6n - S}{2},$$

where

$$S = 2[(l-1)(w+h) + wh + w_b + h_b].$$

There is a point set in the special form for every tuple of parameters l, w, h, w_b, and h_b, where

- the n points fit into a $l \times w \times h$ cuboid, i.e. $lwh \geq n$,

[1] Yue and Dill call this layer a barnacle layer

- the perimeter of the last layer is not too large, i.e. $w_b + w_h \leq w + h$, and
- the perimeter of the last layer $2(w_b + w_h)$ is minimal for a perimeter around the $n - (l-1)wh$ many points of the last layer.

Due to these considerations it is sufficient to maximize the number of contacts subjected to those constraints in order to find the maximal number of contacts $c_{\max}$ of n points. Furthermore, by enumerating all l, w, h, w_b, and h_b that satisfy the constraints and where contacts$(l, w, h, w_b, h_b) = c_{\max}$, we get all frames in Frames$(c_{\max}, n)$.

4.1.2 Extension of CHCC

For generating the frames of sub-optimal point sets, i.e. Frames(c, n), where $c < c_{\max}$, we need to reconfigure sub-optimal connected point sets again while preserving the frame, the number of elements and not decreasing the number of contacts.
Only optimal point sets can be reconfigured as in the fundamental claim of CHCC. In consequence, the CHCC method cannot be applied directly for enumerating frame sequences. A counter-example is shown in Figure 4.2. There, the frame $2, 2, 3$ cannot be filled in the special form at all since whatever frame dimension is chosen as l, w and h, there are too few elements in P. For example, for $l = 3$, $w = 2$, and $h = 2$, we cannot fill two layers of dimensions 2×2 completely. If we choose $l = 2$, $w = 3$, $h = 2$ and fill the 2x3x1 cuboid, then we do not have elements left to attach one layer, which is necessary to fill the frame.

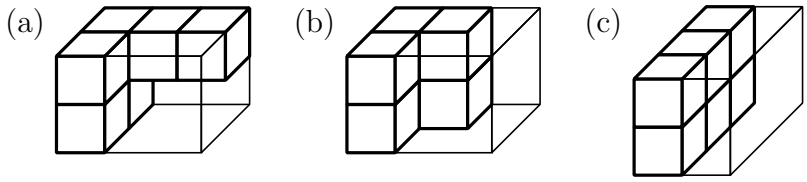


Figure 4.2: a) A sub-optimal point set that cannot be reconfigured to the special form, while preserving the frame. b) and c) Possible special forms.

As a remedy we redefine the special form such that arbitrary many minimally filled layers, i.e. layers of size 1, can be attached to the original form. The idea is that any connected point set can be reconfigured into this form, while

preserving the frame, number of elements and not decreasing the number of contacts. The extended special form is shown in Figure 4.3. This serves as a basis to construct the sets Frames(c, n) in analogy to the original CHCC.
The extended special form is specified by parameters l,h, and w for the surrounding cuboid, l_c, h_c, and w_c specifying the dimensions of the completely filled cuboid as $l_c - 1$, h_c, and w_c and the dimensions of the attached layer w_b and w_h. For these parameters, there is a coloring in this form with the number of contacts

$$\text{contacts}(l, w, h, l_c, w_c, h_c, w_b, h_b) = \frac{6n - S}{2},$$

where

$$S = 2[(l_c - 1)(w_c + h_c) + w_c h_c + w_b + h_b] + 4[l - l_c + w - w_c + h - h_c].$$

To construct the frame set it suffices to enumerate[2] all such parameters subject to the constraints

1. $\text{contacts}(l, w, h, l_c, w_c, h_c, w_b, h_b) = c$,
2. $lwh \geq n$,
3. $l_c \leq l, w_c \leq w, h_c \leq h$,
4. $w_b \leq w_c$, $h_b \leq h_c$ and
5. $2(w_b + h_b)$ is the minimal perimeter of any layer with $n - (l-1)wh - (l + w + h - l_c - w_c - h_c)$ elements.

4.2 Preliminaries

Representation of FCC. Throughout this chapter, we use a representation of the FCC lattice that rotates D_3 by $\phi = 45°$ along the x-axis. This rotation is illustrated in Figure 4.4. In order to get distance 1 between successive x-layers, and distance 1 between neighbors in one x-layer, we additionally scale the y- and z-axis, but leave the x-axis as it is.
We define the FCC-isomorphic lattice D_3' as the lattice that consists of the following sets of points:

$$D_3' = \{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mid \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{Z}^3 \text{ and } x \text{ even}\} \uplus \{\begin{pmatrix} x \\ y+0.5 \\ z+0.5 \end{pmatrix} \mid \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{Z}^3 \text{ and } x \text{ odd}\}.$$

[2] A good choice is using constraint search for this enumeration.

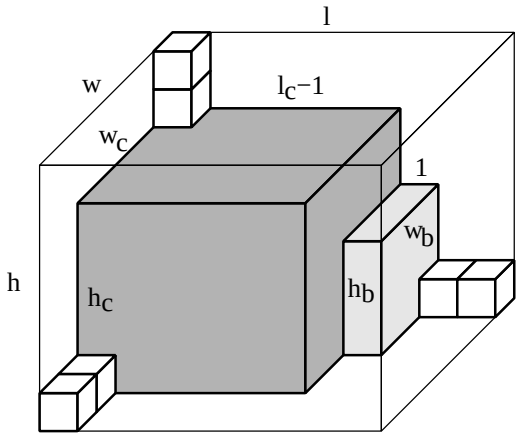


Figure 4.3: Special form for sub-optimal point sets. Again the light cuboid is not necessarily filled completely; each small white cube represents exactly one point.

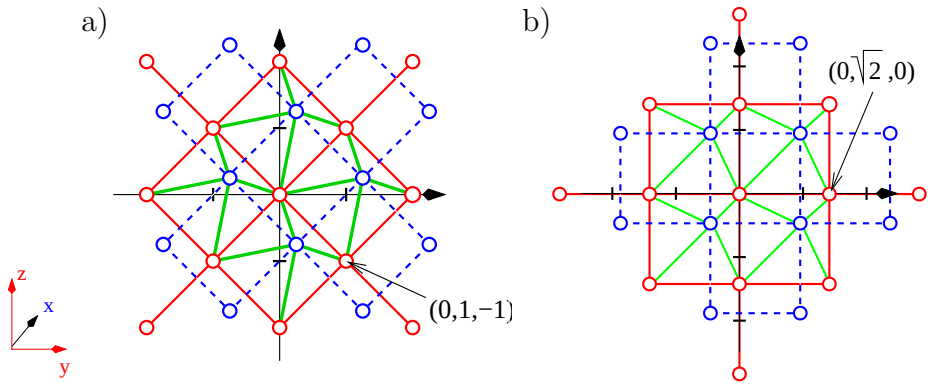


Figure 4.4: In figure a), we have shown two x-layers (where the x-axis is shown as the third dimension). The red circles are the lattice points in the first x-layer, where the red lines are the nearest neighbor connections. The blue circles are the points in the second x-layers. The green lines indicate the nearest neighbor connections between the first and the second x-layer. The figure b) shows FCC after rotation by 45°

The first set consist of the points in even x-layers, the second of the points in odd x-layers. The set $N_{D'_3}$ of *neighbor vectors* connecting neighbors in D'_3 is given by

$$N_{D'_3} = \left\{ \begin{pmatrix} 0 \\ \pm 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix} \right\} \uplus \left\{ \begin{pmatrix} \pm 1 \\ \pm 0.5 \\ \pm 0.5 \end{pmatrix} \right\}.$$

The vectors in the second set are the vectors connecting neighbors in two successive x-layers. Recall that two points $\vec{p}$ and $\vec{p'}$ in D'_3 are *neighbors* if $\vec{p} - \vec{p'} \in N_{D'_3}$.
The set D'_3 with the neighbor relation defined by $N_{D'_3}$ is isomorphic to the earlier representation of the FCC, namely the point set D_3 with neighborship by its minimal vectors.

Colorings. A *coloring* is a function $f : D'_3 \to \{0, 1\}$, where $f^{-1}(1) \neq \emptyset$. We will identify a coloring f with its corresponding (non-empty) point set

$$\{\vec{p} \mid f(\vec{p}) = 1\}.$$

The term coloring is thus almost a synonym of the term point set. We introduce the term coloring, for providing a functional view on point sets. This term was used before in related work [Bac98a, Bac00b, Bac01], and [Bac04]. Colorings inherit all properties and notations from point sets, for example the number of contacts of a coloring is the number of contacts of its corresponding point set and is denoted contacts(f). In particular, we use standard set notation for element relation $\vec{p} \in f_1$, size $|f_1|$, union $f_1 \cup f_2$, disjoint union $f_1 \uplus f_2$, and intersection $f_1 \cap f_2$, where f_1 and f_2 are colorings and $\vec{p}$ is a point.
A coloring f is called a *coloring of the plane* $x = c$ if $f(x, y, z) = 1$ implies $x = c$. We say that f is a *plane coloring* if there is a c such that f is a coloring of plane $x = c$.
With min-x(f) we denote the integer

$$\min\{\vec{p}_x \mid \vec{p} \in f\}.$$

max-x(f), min-y(f), max-y(f), min-z(f) and max-z(f) are defined analogously.
We use a special notation for sub-colorings of a plane coloring f, which is in particular used to split a coloring into disjoint sub-colorings at a row. For a row min-z(f) $\leq z \leq$ max-z(f), we define

$$f_{\theta z} = \{(c, y, z') \in f \mid z' \theta z\},$$

where θ is one of the operators $\leq, <, >, \geq$, or $=$. Note that when using this notation, we have to take care that the sub-colorings are defined. For example, $f_{>z}$ does not denote a valid coloring for $z = \text{max-z}(f)$, since

$$\{(c, y, z') \in f \mid z' > \text{max-z}(f)\}$$

is empty.
We define $\text{Surf}_{pl}(f)$ to be the plane surface of f, i.e.

$$\text{Surf}_{pl}(f) = |\{(\vec{p}, \vec{p'}) \mid (\vec{p} - \vec{p'}) \in N_{D_3'} \wedge f(\vec{p}) \wedge \neg f(\vec{p'}) \wedge \vec{p'}_x = c\}.$$

The frame of a plane coloring f is defined as $\text{frame}(f) = (a, b)$ by

$$\begin{aligned} a &= \text{max-y}(f) - \text{min-y}(f) + 1 \text{ and} \\ b &= \text{max-z}(f) - \text{min-z}(f) + 1. \end{aligned}$$

For connected plane colorings the plane surface is determined by the frame. In general, the number of rows and the number of columns that are occupied by f determine the plane surface. For a plane coloring f of $x = c$ define

$$\text{occ-z}(f, z) = \exists y : f(c, y, z) \text{ and } \text{occ-y}(f, y) = \exists z : f(c, y, z).$$

Furthermore, we define

$$\begin{aligned} \text{occlines-y}(f) &= \left|\left\{\ y \;\middle|\; \text{occ-y}(f, y)\ \right\}\right| \text{ and} \\ \text{occlines-z}(f) &= \left|\left\{\ z \;\middle|\; \text{occ-z}(f, z)\ \right\}\right|. \end{aligned}$$

For notational convenience denote the *occupied lines of a plane coloring* f by

$$\text{occlines}(f) = (\text{occlines-y}(f), \text{occlines-z}(f)).$$

For a plane coloring f, we call rows z, where $\text{occ-z}(f, z)$ holds, and columns y, where $\text{occ-y}(f, y)$, *occupied*, and *unoccupied* otherwise.
For bounding the number of contacts in a plane coloring, we define

$$\text{B}_{\text{LC}}(n, a, b) = \max\left\{\ \text{contacts}(c) \;\middle|\; \begin{array}{c} f \text{ is a plane coloring, } f \text{ has} \\ \text{occupied lines } (a, b), \text{ and } |f| = n \end{array}\ \right\}.$$

For computing this (tight) bound, we specialize the general relation of surface and contacts in a point set.

Proposition 4.2.1
For every cavity-free plane coloring f with $\text{occlines}(f) = (a, b)$*, we get*

$$\mathrm{B_{LC}}(n, a, b) = 2n - \frac{1}{2}\mathrm{Surf}_{pl}(f)$$

and

$$\mathrm{Surf}_{pl}(f) = 2(a + b).$$

Proof
Let f be a cavity-free coloring of plane $x = c$, $(a, b) = \text{occlines}(f)$. We show that $\mathrm{Surf}_{pl}(f) = 2(a + b)$. We need to show that for every y satisfying occ-y(f, y), there are exactly two pairs of points (p, p') with $p_y = p'_y = y$ and $p_x = p'_x = c$ such that

$$(p - p') \in N_{D'_3} \wedge f(p) \wedge \neg f(p').$$

Such a pair (p, p') is called a *surface pair*. Together with the analogous claim for every z, where occ-z(f, z), which (due to symmetry) would be shown strictly analogously, the proposition is implied.
For the left-to-right direction of the equivalence, choose a y such that there exists a z with $f(c, y, z)$. For

$$z^- = \min\{z | f(c, y, z)\} \text{ and } z^+ = \max\{z | f(c, y, z)\},$$

the pairs

$$\left(\begin{pmatrix} c \\ y \\ z^- \end{pmatrix}, \begin{pmatrix} c \\ y \\ z^- - 1 \end{pmatrix}\right) \text{ and } \left(\begin{pmatrix} c \\ y \\ z^+ \end{pmatrix}, \begin{pmatrix} c \\ y \\ z^+ + 1 \end{pmatrix}\right)$$

are surface pairs. Assume there is a further surface pair (p, p') with $p_y = p'_y = y$. Then, necessarily $z^- < p_z < z^+$. By the cavity-freeness of f, p' has to be colored. The reverse direction is trivial. □

4.3 A Bound on Number Sequences

Here, we review an upper bound for the number of contacts of a coloring f in the FCC lattice, given the number of elements in f in each layer $x = c$.
Our aim is defining a function $\mathrm{B_{num}}$, where for any coloring f with x-layer decomposition $f_1, \ldots, f_k$, the number of contacts contacts(f) is at most $\mathrm{B_{num}}(|f_1|, \ldots, |n_k|)$.
For this purpose, we develop an upper bound on the number of interlayer contacts between two connected and cavity-free plane colorings, given their number of elements.

4.3.1 Number of Points with 1,2,3, and 4 Contacts

We start with introducing the notion of an i-point ($i = 1, 2, 3, 4$). An i-point of a coloring f_c of the plane $x = c$ is a point of the layer $x = c+1$, which has i-many neighbors in the coloring f_c (cf. Figure 3.8). We make the definition from Chapter 3 more precise.

Definition 4.3.1 (i-point)
For a coloring f of plane $x = c$, a point $\vec{p}$ of layer $x = c + 1$ is called an i-point of f *if and only if there are exactly i many neighbors $p' \in f$ of $\vec{p}$. For $i = 1, 2, 3,$ or 4, $\#i(f)$ denotes the number of i-points of f in the layer $x = c + 1$.*

Note that due to symmetry, one could equivalently define i-points in the layer $x = c - 1$ for a plane coloring of $x = c$. Due to the structure of the FCC we would count exactly the same numbers of i-points in the layer $x = c - 1$ and in the layer $x = c + 1$.

If we know only the number of i-points of a coloring f and the number of elements n' in the successive plane coloring f', it is easy to compute an upper bound on the number of interlayer contacts between the two plane colorings. For this aim, we distribute the $|f'|$ many points greedily to the i-points of f (with decreasing order) and sum up the contacts.

Hence, our strategy is to bound the number of i-points given only the size of the coloring. If we bound those numbers independently of each other, the resulting bound will be rather weak. However, one observes a dependency between the number of i-points of a plane coloring, when given the number of elements, the plane surface of the coloring. A further parameter is the number of x-steps, which is defined as follows.

Definition 4.3.2 (x-step)
An x-step for a plane coloring f *is a triple $(\vec{p_1}, \vec{p_2}, \vec{p_3})$ such that $f(\vec{p_1}) = 0$, $f(\vec{p_2}) = 1 = f(\vec{p_3})$, $\vec{p_1} - \vec{p_2} = \pm \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ and $\vec{p_1} - \vec{p_3} = \pm \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. With* x-steps$(f)$ *we denote the number of x-steps of f.*

Now, we give the dependency of the numbers of i-points.

LEMMA 4.3.3
Let f be a connected, cavity-free plane coloring.

$$\#4(f) = n + 1 - \frac{1}{2}\text{Surf}_{pl}(f) \tag{4.1}$$
$$\#3(f) = \text{x-steps}(f) \tag{4.2}$$
$$\#2(f) = \text{Surf}_{pl}(f) - 4 - 2\#3(f) \tag{4.3}$$
$$\#1(f) = \#3(f) + 4 \tag{4.4}$$
$$\tag{4.5}$$

With this lemma we cite Lemma 1 of [Bac00b]. Note that there, the number of two-points was equivalently claimed to be

$$\#2(f) = 2n - 2\#4(f) - 2\#3(f) - 2.$$

This equivalence is shown by rewriting:

$$\begin{aligned} & 2n - 2\#4(f) - 2\#3(f) - 2 \\ = \; & 2n - 2(n - \frac{1}{2}\text{Surf}_{pl}(f) + 1) - 2\#3(f) - 2 \\ = \; & \text{Surf}_{pl}(f) - 4 - 2\#3(f). \end{aligned}$$

Here, we will give only a proof sketch for Lemma 4.3.3.
PROOF (SKETCH)
Let f be a connected, cavity-free plane coloring with $(a, b) = \text{frame}(f)$, $n = |f|$. We show the claims by induction on the number of rows b.

Case $b = 1$. Up to translation the points of f are completely determined, since f is connected. It holds $a = n$ and $\text{Surf}_{pl}(f) = 2(n+1)$. The number of 4-points is $0 = n + 1 - \frac{1}{2}\text{Surf}_{pl}(f)$, there are no x-steps and no 3-points, the number of 2-points is $2(n-1) = \text{Surf}_{pl}(f) - 4 - 2\#3(f)$. Finally the number of 1-points is $4 = \#3(f) + 4$.

Case $b > 1$. Due to induction hypothesis the claims hold for the plane coloring $f' = f_{<z}$, where $z = \text{max-z}(f)$. The coloring f extends f', by the attached row $f_{=z}$. All i-points, which do not have contacts to points in the row $f_{=z}$, are unchanged for f' and f.
The rows $f_{=z-1}$ and $f_{=z}$ overlap. Since the rows are cavity-free, the points in each row are connected. The only degree of freedom for the configuration of the last two rows is to shift them against each other. Let r denote the

number of pairs $\vec{p} \in f_{=z-1}$ and $\vec{p'} \in f_{=z}$, where $p_y = p'_y$. We get immediately, that

$$\begin{aligned} \#4(f) =& \#4(f') + (r-1) \\ \text{and } \mathrm{Surf}_{pl}(f) =& \mathrm{Surf}_{pl}(f') + 2(|f_{=z}| - r) + 2. \end{aligned}$$

This proves (4.1), due to

$$\begin{aligned} & \#4(f') + (r-1) \\ \overset{\text{(ind.hyp.)}}{=} \quad & |f'| + 1 - \frac{1}{2}\mathrm{Surf}_{pl}(f') + (r-1) \\ = \quad & n - |f_{=z}| + 1 - \frac{1}{2}\mathrm{Surf}_{pl}(f') + r - 1 \\ = \quad & n + 1 - \frac{1}{2}(\mathrm{Surf}_{pl}(f') + 2|f_{=z}| - 2r + 2) \\ = \quad & n + 1 - \frac{1}{2}\mathrm{Surf}_{pl}(f). \end{aligned}$$

By case distinction on the arrangement of the last two lines, we can show the sub-claims (4.2) and (4.3) using the induction hypothesis for $f_{<z}$.
The last equation follows due to the relation

$$\sum_{i=1}^{4} i \cdot \#i(f) = 4 \cdot n.$$

□

Thus, when given the surface and number of elements, the number of 4-points is fixed. However, the number of 1,2, and 3-points can still vary. As we will see, we can maximize the number of 3-points for getting an upper bound on the number of contacts.

4.3.2 Bound for 3-Points

For bounding the number of 3-points, we investigate the number of x-steps in a connected plane coloring. First, we introduce the notion of a detailed frame.
The *detailed frame* of a coloring f is the tuple $(a, b, i_{lb}, i_{lu}, i_{rb}, i_{ru})$, where (a, b) is the frame of f and i_{lb} is the number of diagonals that can be drawn from the left-bottom corner. i_{lu}, i_{rb}, i_{ru} are defined analogously. For a coloring f with detailed frame $(a, b, i_{lb}, i_{lu}, i_{rb}, i_{ru})$, we call $\vec{i} = (i_{lb}, i_{lu}, i_{rb}, i_{ru})$ the *indent vector of* f.

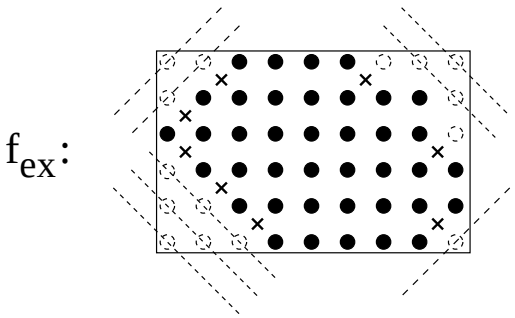


Figure 4.5: Detailed Frame

The number of x-steps depends on the detailed frame and the number of diagonal cavities.

Definition 4.3.4 (Diagonal Cavity)
A diagonal cavity *in f is a k-tuple of points $(\vec{p_1}, \ldots, \vec{p_k})$ of D'_3 with $k \geq 3$ such that*

- $f(\vec{p_1}) = 1 = f(\vec{p_k})$,
- $\forall 1 < j < k : f(\vec{p_j}) = 0$,
- $\forall 1 \leq j < k : \vec{p}_{j+1} = \vec{p_j} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$
  or $\forall 1 \leq j < k : \vec{p}_{j+1} = \vec{p_j} + \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$.

The number of diagonal cavities *in f is denoted* $\mathrm{diagcav}(f)$.

For example, consider the plane coloring f_{ex} as given in Figure 4.3.2. Then the detailed frame of f_{ex} is $(6, 9, 3, 2, 1, 2)$. The number of 3-points (indicated by crosses $\times$) for f_{ex} is $8 = 3 + 2 + 1 + 2$. Furthermore, f_{ex} does not contain diagonal cavities.
We cite Lemma 2 of [Bac00b] for the bound on the number of x-steps.

Lemma 4.3.5
For any coloring f with index vector $(i_{lb}, i_{lu}, i_{rb}, i_{ru})$, holds

$$\text{x-steps}(f) = i_{lb} + i_{lu} + i_{rb} + i_{ru} - \mathrm{diagcav}(f).$$

A proof of this lemma is given in [Bac00b] by induction on the number of columns.

COROLLARY 4.3.6
Let f be a connected, caveat-free plane coloring with frame (a,b), then

$$\text{x-steps}(f) \leq 2(\min(a,b)-1).$$

In the case of connected plane colorings, we search for a given number of points n and a frame (a,b) the maximal number of x-steps. For this purpose, we define for some indent vector $\vec{i} = (i_1, i_2, i_3, i_4)$,

$$\text{vol}(a,b,\vec{i}) = ab - \sum_{1\leq j\leq 4} \frac{i_j(i_j+1)}{2}.$$

$\text{vol}(a,b,\vec{i})$ is the maximal number of points that can be colored by any f that has indent vector $\vec{i}$ and frame (a,b). $\vec{i} = (i_1,i_2,i_3,i_4)$ is called *maximal* for (a,b) if and only if

$$\sum_{1\leq j\leq 4} i_j = 2(\min(a,b)-1).$$

For example, if $b \leq a$, then the indent vector $\vec{i}$ is maximal for (a,b) if every coloring with frame (a,b) and indent vector $\vec{i}$ has exactly one colored point in the first and last column.
$\text{vol}(a,b,\vec{i})$ can now be used to calculate the maximal number of x-steps that can be achieved given n colored points and frame (a,b). The maximal number of x-steps is achieved if we make the indents as uniform as possible. For this purpose, define

$$\text{edge}(n,a,b) = \max\{k \in \mathbb{N} \mid \text{vol}(a,b,(k,k,k,k))\}.$$

$k = \text{edge}(n,a,b)$ defines the maximal possible uniform indent. Then

$$r = \text{ext}(n,a,b) = \lfloor \frac{ab - 4\frac{k(k+1)}{2} - n}{k+1} \rfloor$$

defines the number of times r we can extend the uniform indent by 1.

DEFINITION 4.3.7 (NORMAL)
n is called normal for (a,b) *if either*

$$4k + r < 2(a-1),$$

or

$$4k + r = 2(a-1) \quad \textit{and} \quad ab - 4\frac{k(k+1)}{2} - r(k+1) = n.$$

Now, there are two upper bounds that can be given for the number of x-steps, given n colored points and frame (a, b). The first is given by the indent vector. The second by the fact, that in cavity-free and connected plane colorings, there may be at most two x-steps between every two successive lines, which yields at most $2\min(a, b) - 1$. Thus, we define

$$\mathrm{B}_{\text{x-steps}}(n, a, b) = \min \left\{ \begin{array}{l} 4\,\mathrm{edge}(n, a, b) + \mathrm{ext}(n, a, b) \\ 2\min(a, b) - 1 \end{array} \right\}.$$

Then, we get

Lemma 4.3.8 (Bound on x-steps)
For any connected, cavity-free plane coloring f,

$$\text{x-steps}(f) \leq \mathrm{B}_{\text{x-steps}}(|f|, a, b),$$

where $(a, b) = \mathrm{frame}(f)$.

4.3.3 Upper Bound for Interlayer Contacts

First, we bound the number of interlayer contacts between two plane colorings. We introduce a notation for the number of such interlayer contacts.

Definition 4.3.9 (Interlayer Contacts)
For two plane colorings f and f' of different planes, we define

$$\mathrm{IC}_f^{f'} = |\{(\vec{p}, \vec{p}') \mid \vec{p}, \vec{p}' \text{ neighbors}, \vec{p} \in f, \vec{p}' \in f'\}|.$$

Due to Lemma 4.3.3, we can compute the number of i-points in a connected plane coloring f if we know $n = |f|$, $(a, b) = \mathrm{frame}(f)$, and $\ell = \text{x-steps}(f)$ as $\#i(f) = \#i(n, a, b, \ell)$, where we define

$$\begin{aligned} \#4(n, a, b, \ell) &= n + 1 - (a + b) \\ \#3(n, a, b, \ell) &= \ell \\ \#2(n, a, b, \ell) &= -2\ell + 2(a + b) - 4 \\ \#1(n, a, b, \ell) &= \ell + 4. \end{aligned}$$

Due to this, we derive a bound on the interlayer contacts $\mathrm{IC}_f^{f'}$, given these parameters and the size n' of f', namely $\mathrm{IC}_f^{f'} \leq \mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n, a, b, \ell; n')$ for

$$\mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n, a, b, \ell; n') = \sum_{i=1}^{4} i \cdot \min(\#i(n, a, b, \ell), \mathrm{r}_i),$$

where we use some auxiliary variables

$$\begin{aligned} & \mathrm{r}_4 = n' \text{ and} \\ \text{for } i = 3,2,1, \quad & \mathrm{r}_i = \max(0, \mathrm{r}_{i+1} - \#(i+1)(n,a,b,\ell)). \end{aligned}$$

One can show, that $\mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n,a,b,\ell;n')$ is maximized for fixed n, a, b, and n' if we maximize ℓ.

Proposition 4.3.10
For $a, b, \ell, n, n' \in \mathbb{N}$,

$$\mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n,a,b,\ell+1;n') \geq \mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n,a,b,\ell;n').$$

The proposition holds, since increasing the number of 3-points by one, trades two 2-points into one 3-point and one 1-point.
Thus, replacing ℓ with its upper bound $\mathrm{B}_{\text{x-steps}}(n,a,b)$ yields an upper bound on the interlayer contacts. Hence, we define

$$\mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n,a,b;n') = \mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n,a,b,\mathrm{B}_{\text{x-steps}}(n,a,b);n'). \tag{4.6}$$

and get the following theorem.

Theorem 4.3.11
For two connected plane colorings f and f' of different planes holds

$$\mathrm{IC}^{f'}_{f} \leq \mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n,a,b;n'),$$

where $n = |f|$, $(a,b) = \mathrm{frame}(f)$, and $n' = |f'|$.

4.3.4 Contact Bound by Number Sequence

Finally, the bound shall be used for defining $\mathrm{B}_{\mathrm{num}}$. As a last preparation, we need to circumvent the restriction to connected plane colorings in the bound $\mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n,a,b;n')$.
Since we are only interested in bounding the contacts between colorings, given the number of elements in both plane colorings, we can use the following proposition.

Proposition 4.3.12
For each plane coloring f with frame (a,b) and n elements, there exists a connected plane coloring f' with frame (a',b') and n' elements with $n' = n$, $a' \leq a$, and $b' \leq b$ that has at least as many contacts.

Due to this proposition, it suffices to enumerate only frames, where a connected coloring for n elements exists in order to bound the number of contacts of two plane colorings with given size. [Bac00b] goes beyond this consideration and shows that enumerating only normal and almost normal frames is sufficient for this purpose. This improvement can then be used to speed up the computation in practice.

Definition 4.3.13 (Number Sequence Bound)
For a number sequence $(n_1, \ldots, n_k)$ *define*

$$\begin{aligned} \mathrm{B}_{\mathrm{num}}(n_1, \ldots, n_k) = &\sum_{i=1}^{k-1} \max_{n\text{-frame } (a,b)} (\mathrm{B}_{\mathrm{LC}}(n_i, a, b) + \mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n_i, a, b; n_{i+1})) \\ &+ \ \mathrm{B}_{\mathrm{LC}}(n_k, a^m, b^m), \end{aligned}$$

where

$$\begin{aligned} a^m &= \lceil \sqrt{n_k} \rceil, \\ b^m &= \lceil n_k / a^m \rceil, \end{aligned}$$

and (a, b) *is an* n*-frame if and only if* $ab \geq n$ *and* $a + b - 1 \leq n$.

Theorem 4.3.14
For any point set f *with number sequence* $(n_1, \ldots, n_k)$ *holds*

$$\mathrm{contacts}(f) \leq \mathrm{B}_{\mathrm{num}}(n_1, \ldots, n_k).$$

4.4 Properties of Connected and Unconnected Plane Colorings

Let f be a coloring of plane $x = c$. A *horizontal cavity in* f is a k-tuple of points $(\vec{p_1}, \ldots, \vec{p_k})$ such that

$$\begin{aligned} &\forall 1 \leq j < k : \ ((\vec{p_{j+1}} - \vec{p_j})_y = 1), \{\vec{p}_1, \vec{p}_k\} \in f \\ \text{and } &\forall 1 < j < k : \ \vec{p_j} \notin f. \end{aligned}$$

A *vertical cavity in* f is defined analogously satisfying instead

$$\forall 1 \leq j < k : \ ((\vec{p_{j+1}} - \vec{p_j})_z = 1).$$

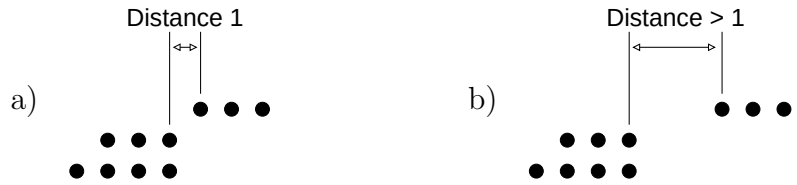


Figure 4.6: a) Non-overlapping vs. b) non-touching

We say that f contains a *cavity* if there is at least one horizontal or vertical cavity in f. f is called *cavity-free* if it does not contain a cavity. For developing an upper bound on the number of contacts, it is sufficient to handle only cavity-free colorings.
We introduce further parameters of a plane coloring f that allow us to bound the interlayer contacts. These parameters are the number of non-touching and non-overlapping rows.
Let f be a coloring of plane $x = c$. We define

$$\text{overlaps}(f, z) = \exists y : (f(c, y, z) \wedge f(c, y, z + 1)),$$

and

$$\#\text{non-overlaps}(f) = \left|\left\{\ z < \text{max-z}(f) \ \middle|\ \text{occ-z}(f, z) \wedge \neg\, \text{overlaps}(f, z)\ \right\}\right|.$$

We call an occupied row $z < \text{max-z}(f)$ in f *overlapping* if $\text{overlaps}(f, z)$ and otherwise non-overlapping. Non-touching rows are non-overlapping rows z, where the y-distance of the points in z and $z+1$ exceeds 1. An occupied row $z < \text{max-z}(f)$ of f is called *touching* if and only if $\text{touching}(f, z)$, where

$$\text{touching}(f, z) = \exists y, y' : f(c, y, z) \wedge f(c, y', z + 1) \wedge |y - y'| \leq 1.$$

It is called non-touching otherwise. The number of non-touching rows is

$$\#\text{non-touchs}(f) = \left|\left\{\ z < \text{max-z}(f) \ \middle|\ \text{occ-z}(f, z) \wedge \neg\, \text{touching}(f, z)\ \right\}\right|.$$

We call a coloring f with $\#\text{non-overlaps}(f) = 0$, an *overlapping* (otherwise *non-overlapping*) coloring. A coloring with $\#\text{non-touchs}(f) = 0$, is called a *touching* (otherwise *non-touching*) coloring.
For illustrating the terms, Figure 4.6a shows a non-overlapping, but still touching coloring f_a ($\#\text{non-overlaps}(f_a) = 1$ and $\#\text{non-touchs}(f_a) = 0$), whereas the coloring f_b in Figure 4.6b is non-overlapping and non-touching ($\#\text{non-overlaps}(f_b) = 1$ and $\#\text{non-touchs}(f_b) = 1$).

Note that for any coloring f, the row max-z(f) satisfies occ-z(f, max-z(f)), $\neg$ overlaps(f, max-z(f)) and $\neg$ touching(f, max-z(f)), since by definition of max-z(f) there hold occ-z(f, max-z(f)) and $\neg$ occ-z(f, max-z(f) + 1). However, by definition the last row is neither overlapping (touching) nor non-overlapping (non-touching), respectively.
One can show that a plane coloring is overlapping if and only if it is connected. Hence, the terms overlapping and connected are synonymous.

Proposition 4.4.1
A cavity-free plane coloring f is overlapping if and only if it is connected.

Proof
Let f be a cavity-free and connected coloring of plane $x = c$. Assume there is a $z <$ max-z(f) such that occ-z(f, z) and $\neg$ overlaps(f, z). Since f is connected, which implies that there is a path from a point in the first occupied line to a point in the last occupied line, there is a y, such that $f(c, y, z)$ and $f(c, y, z + 1)$. This is a contradiction to the definition of overlaps(f, z).
For the opposite direction, let f be cavity-free and overlapping. Then, f is connected, since

1. all points $\vec{p}, \vec{p}' \in f$ in the same line z, i.e. where $p_z = p'_z = z$, are connected, otherwise f would have a horizontal cavity.

2. for $z <$ max-z(f), where occ-z(f, z), there are neighbored points $\vec{p}, \vec{p}' \in f$, where $p_z = z$ and $p'_z = z + 1$, and in particular occ-z($f, z + 1$).

By 1 and 2, any point in f is connected to all points in the succeeding line of f. Then, by induction all points in f are connected to each other. □

We introduce a technical lemma, which will be employed for inductive arguments. A cavity-free coloring can be split at non-overlapping rows into sub-colorings with the nice property that the parameters of the coloring can be calculated from the sub-colorings in a simple way.

Lemma 4.4.2 (Split)
Let f be a cavity-free coloring of the plane $x = c$ with #non-overlaps(f) ≥ 1, *and let* min-z(f) $\leq z_s <$ max-z(f) *be a non-overlapping row. Then,*

1.

$f = f_{\leq z_s} \uplus f_{>z_s}$ *and the sub-colorings $f_{\leq z_s}$ and $f_{>z_s}$ are cavity-free.*

2.

$$\text{occlines}(f) = (\,\text{occlines-y}(f_{\leq z_s}) + \text{occlines-y}(f_{>z_s}),\\ \text{occlines-z}(f_{\leq z_s}) + \text{occlines-z}(f_{>z_s}))$$

3.

$$\#\text{non-overlaps}(f) = \#\text{non-overlaps}(f_{\leq z_s}) + \#\text{non-overlaps}(f_{>z_s}) + 1.$$

The split lemma 4.4.2 allows us to decompose a coloring at an non-overlapping row. Notably, the parameters of the two generated sub-colorings can easily be added up.

Proof of Lemma 4.4.2

We proof the three claims of the lemma separately. Let f, z_s, $f_{\leq z_s}$ and $f_{>z_s}$ given as defined in the lemma.

1. For the existence of the sub-colorings it suffices to show that the sub-colorings are non-empty. $f_{\leq z_s}$ is non-empty, since occ-z(f, z_s), and $f_{>z_s}$ is non-empty, since the row max-z(f) is occupied in f and $z_s <$ max-z(f). The sub-claim that $f_{\leq z_s}$ and $f_{>z_s}$ are disjoint follows directly from the definition.

   Assume that there is a cavity $(p_1, \ldots, p_n)$ in $f_{\leq z_s}$. By the definition of a cavity, $f_{\leq z_s}(p_1)$ and $f_{\leq z_s}(p_n)$ hold, which implies again by definition that all points of the cavity have z-values less or equal than z_s. Hence, the points $p_2, \ldots, p_{n-1}$, which are uncolored in $f_{\leq z_s}$ are also uncolored in f. Finally, this implies that $(p_1, \ldots, p_n)$ is a cavity in f, which is a contradiction. The case $f_{>z_s}$ is analogous.

2. The sub-claim

   $$\text{occlines-z}(f) = \text{occlines-z}(f_{\leq z_s}) + \text{occlines-z}(f_{>z_s})$$

   is almost obvious. Recall that

   $$\text{occlines-z}(f) = |\{z \mid \text{occ-z}(f, z)\}|.$$

   Now,

   $$\begin{aligned}\text{occlines-z}(f) &= |\{z \mid \text{occ-z}(f, z)\}|\\ &= |\{z \mid z \leq z_s \wedge \text{occ-z}(f, z)\} \uplus \{z \mid z > z_s \wedge \text{occ-z}(f, z)\}|\\ &= |\{z \mid \text{occ-z}(f_{\leq z_s}, z)\}| + |\{z \mid \text{occ-z}(f_{>z_s}, z)\}|\\ &= \text{occlines-z}(f_{\leq z_s}) + \text{occlines-z}(f_{>z_s}).\end{aligned}$$

For

$$\text{occlines-y}(f) = \text{occlines-y}(f_{\leq z_s}) + \text{occlines-y}(f_{>z_s}),$$

we show that

$$\text{for all } y\text{: not exist } z \leq z_s,\ z' > z_s\text{: } f(c,y,z) \text{ and } f(c,y,z') \qquad (4.7)$$

Assume there are for a y, $z \leq z_s$ and $z' > z_s$, such that $f(c,y,z)$ and $f(c,y,z')$. Then, since f is cavity-free, for all $z \leq z'' \leq z'$ there holds, $f(c,y,z'')$. In particular, $f(c,y,z_s)$ and $f(c,y,z_s{+}1)$, which contradicts, $\neg\,\text{overlaps}(f,z_s)$. The claim is a consequence of (4.7), since

$$\begin{aligned}
\text{occlines-y}(f) &= |\{y \mid \exists z : f(c,y,z)\}| \\
&= |\{y \mid \exists z \leq z_s : f(c,y,z)\} \cup \{y \mid \exists z > z_s : f(c,y,z)\}| \\
&=_{(4.7)} |\{y \mid \exists z \leq z_s : f(c,y,z)\}| \\
&\qquad + |\{y \mid \exists z > z_s : f(c,y,z)\}| \\
&= \text{occlines-y}(f_{\leq z_s}) + \text{occlines-y}(f_{>z_s}).
\end{aligned}$$

3. By definition, holds

$$\begin{aligned}
&\text{for } z < z_s : \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z) \\
&\qquad\qquad \iff \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f_{\leq z_s},z)
\end{aligned}$$

and

$$\begin{aligned}
&\text{for } z > z_s : \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z) \\
&\qquad\qquad \iff \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f_{>z_s},z).
\end{aligned}$$

Additionally we know, that $z_s < \text{max-z}(f)$ and

$$\text{occ-z}(f,z_s) \wedge \neg\,\text{overlaps}(f,z_s).$$

Thus,

$$\left\{\ z < \text{max-z}(f) \;\middle|\; \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z)\ \right\}$$

is disjointly decomposed into

- $\left\{\ z < z_s \;\middle|\; \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z)\ \right\}$,
- $\{z_s\}$, and
- $\left\{\ z_s < z < \text{max-z}(f) \;\middle|\; \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z)\ \right\}$.

Now, Claim 3 follows by calculation

$$\begin{aligned}
&\#\text{non-overlaps}(f)\\
=&\left|\left\{\ z < \text{max-z}(f) \;\middle|\; \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z)\ \right\}\right|\\
=&\left|\left\{\ z < z_s \;\middle|\; \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z)\ \right\}\right|\\
&+1+\left|\left\{\ z_s < z_s < \text{max-z}(f) \;\middle|\; \text{occ-z}(f,z) \wedge \neg\,\text{overlaps}(f,z)\ \right\}\right|\\
=&\left|\left\{\ z < \text{max-z}(f_{\leq z_s}) \;\middle|\; \text{occ-z}(f_{\leq z_s},z) \wedge \neg\,\text{overlaps}(f_{\leq z_s},z)\ \right\}\right|\\
&+1+\left|\left\{\ z_s < \text{max-z}(f_{>z_s}) \;\middle|\; \text{occ-z}(f_{>z_s},z) \wedge \neg\,\text{overlaps}(f_{>z_s},z)\ \right\}\right|\\
=&\#\text{non-overlaps}(f_{\leq z_s}) + \#\text{non-overlaps}(f_{>z_s}) + 1.
\end{aligned}$$

□

In the rest of this section, we give tight bounds on the number of colored points, given the parameters of the plane coloring. We will first state some properties of colorings with respect to occlines(f), #non-overlaps(f) and #non-touchs(f).
The occupied lines yield a first, simple restriction on the minimal number of elements in any cavity-free coloring. The maximal number of non-overlapping rows is bounded by the number of occupied lines in both dimensions.

Proposition 4.4.3
For every cavity-free coloring f,

$$|f| \geq \max(\text{occlines}(f)).$$

Proof
Let f be a cavity-free coloring, the definition of occlines(f) $= (a, b)$ postulates the existence of at least a and at least b colored points in f. □

Since by definition the maximal occupied row z can not be non-overlapping we immediately get that #non-overlaps(f) is less than occlines-y(f). The next lemma states in addition that #non-overlaps(f) is less than occlines-z(f). Intuitively, this is a consequence of the (non-trivial) fact that every non-overlapping row produces exactly one non-overlapping column.

Lemma 4.4.4
For a cavity-free coloring f, we get

$$\#\text{non-overlaps}(f) < \min(\text{occlines}(f)).$$

Proof
By induction over #non-overlaps(f). For the base case, let f be a plane coloring with #non-overlaps(f) = 0. Since f is non-empty, the claim holds trivially.
For the induction step, let f be a plane coloring with #non-overlaps(f) > 0. Let $f_{\leq z_m}$ and $f_{>z_m}$ be the colorings generating by split f at row z_m, where z_m is the minimal row with occ-z(f, z_m)$\wedge\neg$ overlaps(f, z_m). Since $z_m <$ max-z(f) by minimality, we know by Lemma 4.4.2 that

$$\begin{aligned}\text{occlines-y}(f) &= \text{occlines-y}(f_{\leq z_m}) + \text{occlines-y}(f_{>z_m})\\ \text{occlines-z}(f) &= \text{occlines-z}(f_{\leq z_m}) + \text{occlines-z}(f_{>z_m})\end{aligned}$$

and

$$\#\text{non-overlaps}(f) - 1 = \#\text{non-overlaps}(f_{\leq z_m}) + \#\text{non-overlaps}(f_{>z_m})$$

By the induction hypotheses, we get

$$\begin{aligned}\text{occlines-y}(f) &= \text{occlines-y}(f_{\leq z_m}) + \text{occlines-y}(f_{>z_m})\\ &\geq \#\text{non-overlaps}(f_{\leq z_m}) + 1 + \#\text{non-overlaps}(f_{>z_m}) + 1\\ &= \#\text{non-overlaps}(f) + 1\end{aligned}$$

and

$$\begin{aligned}\text{occlines-z}(f) &= \text{occlines-z}(f_{\leq z_m}) + \text{occlines-z}(f_{>z_m})\\ &\geq \#\text{non-overlaps}(f_{\leq z_m}) + 1 + \#\text{non-overlaps}(f_{>z_m}) + 1\\ &= \#\text{non-overlaps}(f) + 1\end{aligned}$$

□

There are two different measures that characterize the shape of a plane coloring f. One is frame(f), the other occlines(f). We will show that both notions coincide for touching plane colorings.

Proposition 4.4.5 (Overlapping Colorings)
Let f be a cavity-free coloring of plane $x = c$ with #non-touchs(f) = 0. *Then*

$$\forall\,\text{min-y}(f) \leq y \leq \text{max-y}(f) : \text{occ-y}(f, y) \tag{4.8}$$
$$\forall\,\text{min-z}(f) \leq z \leq \text{max-z}(f) : \text{occ-z}(f, z). \tag{4.9}$$

Furthermore, occlines(f) = frame(f).

PROOF
Let f be a cavity-free coloring of plane $x = c$ with #non-touchs$(f) = 0$.
Then, for Claim (4.9), assume that there is a z, where min-z$(f) \leq z \leq$ max-z(f) and $\neg$ occ-z(f, z). Choose z_0 to be the maximal value $z_0 < z$, such that occ-z(f, z_0). Such a z_0 exists, since $z >$ min-z(f). Since $\neg$ occ-z$(f, z_0{+}1)$, there holds $\neg$ touching(f, z_0).
For Claim (4.9), assume that there is a y, where min-y$(f) \leq y \leq$ max-y(f) and $\neg$ occ-y(f, y). Immediately, min-y$(f) <$ y $<$ max-y(f). We show that in f are either only points $\vec{p}$ with $p_y < y$ (on the left of the column y) or only points with $p_y > y$ (on the right of the column y), which is a contradiction.
In each row there are either only points on the left or only points on the right of column y. Otherwise, there is a cavity in f, since the column y is unoccupied. By induction on the number of rows, either all point in the coloring are on the left or all points are on right of y. For only one row this is already shown. For more than one row, the claim holds for all rows but the row $z_0 = \min_z f$ by induction. Since touching(f, z_0), all points in the row z_0 are on the same side of column y as the points in row $z_0 + 1$ and the whole plane coloring. □

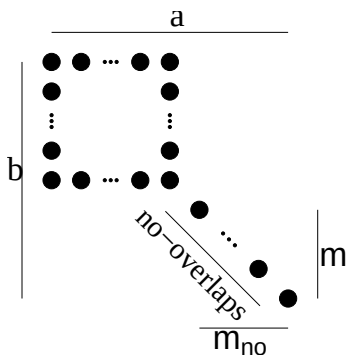


Figure 4.7: Coloring with maximal number of elements.

In a cavity-free plane coloring f, with given number of occupied lines in y and z direction and number of elements, only some numbers of non-overlapping rows #non-overlaps(f) can occur. In the following, we investigate this dependency of the number of non-overlapping rows m_{no}, the occupied lines (a, b) and the size n in a cavity-freeplane coloring f. We define

$$\text{n}_{\max}(a, b, m_{\text{no}}) = m_{\text{no}} + (a - m_{\text{no}})(b - m_{\text{no}})$$

and

$$\mathrm{n}_{\min}(a, b, m_{\mathrm{no}}) = a + b - 1 - m_{\mathrm{no}}.$$

The idea of the definition of $\mathrm{n}_{\max}(a, b, m_{\mathrm{no}})$ is that the number of elements is maximized if we have one large overlapping region and waste as little space as possible for the non-overlapping region. Hence, in this maximal coloring, all of the non-overlapping rows contain exactly one point. Such a coloring is shown in Figure 4.7.

LEMMA 4.4.6
All cavity-free colorings f satisfy

$$|f| \leq \mathrm{n}_{\max}(a, b, m_{\mathrm{no}}),$$

where $m_{\mathrm{no}} = \#\text{non-overlaps}(f)$ *and* $(a, b) = \text{occlines}(f)$.

PROOF
We proof the proposition by induction on $m_{\mathrm{no}} > 0$.

Case $m_{\mathrm{no}} = 0$**.** The maximal number of elements in an overlapping coloring with occupied lines (a, b) is clearly ab, thus the base case holds.

Case $m_{\mathrm{no}} > 0$**.** Let f be an arbitrary cavity-free coloring with $\text{occlines}(f) = (a, b)$ and $\#\text{non-overlaps}(f) = m_{\mathrm{no}}$. Furthermore, let z_s be minimal such that $\neg\,\text{overlaps}(f, z_s)$ holds. Then, due to Lemma 4.4.2, we can split f at the row z_s into $f_{\leq z_s}$ and $f_{>z_s}$ such that

- $f = f_{\leq z_s} \uplus f_{>z_s}$
- $\text{occlines}(f_{\leq z_s}) = (a', b')$ and $\text{occlines}(f_{>z_s}) = (a - a', b - b')$.
- $\#\text{non-overlaps}(f_{\leq z_s}) = 0$ and $\#\text{non-overlaps}(f_{>z_s}) = m_{\mathrm{no}} - 1$.

By Lemma 4.4.4, we know that

$$\text{occlines-y}(f_{>z_s}) \geq m_{\mathrm{no}} \quad \text{and} \quad \text{occlines-z}(f_{>z_s}) \geq m_{\mathrm{no}}.$$

Since $\text{occlines}(f_{>z_s}) = (a - a', b - b')$, this implies

$$a - m_{\mathrm{no}} \geq a' \quad \text{and} \quad b - m_{\mathrm{no}} \geq b'.$$

By induction hypothesis for $f_{\leq z_s}$ and $f_{>z_s}$,

$$\begin{aligned} |f| &= |f_{\leq z_s}| + |f_{>z_s}| \\ &\leq \mathrm{n}_{\max}(a', b', 0) + \mathrm{n}_{\max}(a - a', b - b', m_{\mathrm{no}} - 1). \end{aligned} \tag{4.10}$$

We temporarily introduce the function g, by

$$g(a', b') := \mathrm{n}_{\max}(a', b', 0) + \mathrm{n}_{\max}(a - a', b - b', m_{\mathrm{no}} - 1).$$

Given this, we need only to show that for any $1 \leq a' \leq a - m_{\mathrm{no}}$ and $1 \leq b' \leq b - m_{\mathrm{no}}$,

$$g(a', b') \leq \mathrm{n}_{\max}(a, b, m_{\mathrm{no}}).$$

We will determine the maximum of the function g, for

$$1 \leq a' \leq a - m_{\mathrm{no}} \quad \text{and} \quad 1 \leq b' \leq b - m_{\mathrm{no}}.$$

The explicit form of g is

$$\begin{aligned} g(a', b') =& a'b' + m_{\mathrm{no}} - 1 + (a - a' - (m_{\mathrm{no}} - 1))(b - b' - (m_{\mathrm{no}} - 1)) \\ =& a'b' + m_{\mathrm{no}} - 1 + (a' - a + m_{\mathrm{no}} - 1)(b' - b + m_{\mathrm{no}} - 1). \end{aligned}$$

$g(a', b')$ is linear in a' and b'. Hence, the optima of the function must be at the border of the domains for a' and b'. In other words, $g(a', b')$ is maximized within our restricted domain for at least one pair

$$(a', b') \in \{(1, 1), (a - m_{\mathrm{no}}, 1), (1, b - m_{\mathrm{no}}), (a - m_{\mathrm{no}}, b - m_{\mathrm{no}})\}.$$

By simple calculation,

$$\begin{aligned} g(1, 1) &= 1 + m_{\mathrm{no}} - 1 + (a - 1 - (m_{\mathrm{no}} - 1))(b - 1 - (m_{\mathrm{no}} - 1)) \\ &= m_{\mathrm{no}} + (a - m_{\mathrm{no}})(b - m_{\mathrm{no}}) = g(a - m_{\mathrm{no}}, b - m_{\mathrm{no}}) \end{aligned}$$

and

$$\begin{aligned} g(a - m_{\mathrm{no}}, 1) &= a - m_{\mathrm{no}} + m_{\mathrm{no}} - 1 + b - m_{\mathrm{no}} \\ &= a + b - (m_{\mathrm{no}} + 1) = g(1, b - m_{\mathrm{no}}). \end{aligned}$$

It remains to be shown that

$$m_{\mathrm{no}} + (a - m_{\mathrm{no}})(b - m_{\mathrm{no}}) \geq a + b - (m_{\mathrm{no}} + 1).$$

This holds since

$$\begin{aligned} & m_{\mathrm{no}} + (a - m_{\mathrm{no}})(b - m_{\mathrm{no}}) - (a + b - (m_{\mathrm{no}} + 1)) \\ = \; & 2m_{\mathrm{no}} + ab - am_{\mathrm{no}} - bm_{\mathrm{no}} + {m_{\mathrm{no}}}^2 - a - b + 1 \\ = \; & ab - a(m_{\mathrm{no}} + 1) - b(m_{\mathrm{no}} + 1) + (mno^2 + 2m_{\mathrm{no}} + 1) \\ = \; & ab - a(m_{\mathrm{no}} + 1) - b(m_{\mathrm{no}} + 1) + (mno + 1)^2 \\ = \; & (a - (m_{\mathrm{no}} + 1))(b - (m_{\mathrm{no}} + 1)) \end{aligned}$$

and

$$(a - (m_{\text{no}} + 1))(b - (m_{\text{no}} + 1)) \geq 0,$$

due to $a, b \geq m_{\text{no}} + 1$ by Lemma 4.4.4. □

LEMMA 4.4.7
For all cavity-free colorings f holds $\text{n}_{\min}(a, b, m_{\text{no}}) \leq |f|$, *where* $(a, b) = \text{occlines}(f)$ *and* $m_{\text{no}} = \#\text{non-overlaps}(f)$.

PROOF
We proof the claim by induction on the number of non-overlapping rows m_{no}.

Case $m_{\text{no}} = 0$**.** We show by induction on the number of columns a, that any overlapping coloring f with $\text{occlines}(f) = (a, b)$ satisfies

$$|f| \geq a + b - 1.$$

For the base case $a = 1$, obviously for any overlapping f with $\text{occlines}(f) = (1, b)$, $|f| = b = a + b - 1$.
For the induction case $a > 1$, we choose f as an overlapping coloring with $\text{occlines}(f) = (a, b)$. We split f into the colorings

$$f' = \{\vec{p} \mid p_y < \text{max-y}(f)\} \quad \text{and} \quad f'' = \{\vec{p} \mid p_y = \text{max-y}(f)\}.$$

Both sub-colorings exist, i.e. the sets f' and f'' are non-empty, by definition of $\text{max-y}(f)$ and $a > 1$. f' has to be overlapping, since f is overlapping. Obviously, $\text{occlines-y}(f') = a - 1$ holds. We have to give an lower bound for $|f''|$, which is $\text{occlines-z}(f'')$. Since f is cavity-free, we get

$$\begin{aligned} b &= \text{occlines-z}(f) \\ &= \text{occlines-z}(f') + \text{occlines-z}(f'') - \#\text{overlap\_points}, \end{aligned}$$

where

$$\#\text{overlap\_points} = |\{z \mid f(c, \text{max-y}(f) - 1, z) \wedge f(c, \text{max-y}(f), z)\}.$$

Note that $\#\text{overlap\_points} \geq 1$ since f has to be overlapping. Hence,

$$\begin{aligned} |f''| = \text{occlines-z}(f'') &= b - \text{occlines-z}(f') + \#\text{overlap\_points} \\ &\geq b - \text{occlines-z}(f') + 1 \end{aligned}$$

Hence, by induction hypothesis for f'

$$|f| = |f'| + |f''| \geq (a-1) + \text{occlines-z}(f') - 1 + b - \text{occlines-z}(f') + 1 = a + b - 1.$$

Case $m_{\text{no}} > 0$. For the main induction case, let z_s be the minimal row with occ-z$(f, z_s) \wedge \neg$ overlaps(f, z_s). By Lemma 4.4.2, we know that $f_{\leq z_s}$ and $f_{>z_s}$ is a decomposition of f into two disjoint cavity-free sub-colorings such that

1. #non-overlaps$(f_{\leq z_s}) = 0$ and #non-overlaps$(f_{>z_s}) = m_{\text{no}} - 1$, and
2. occlines$(f_{>z_s}) = (a - a', b - b')$, where $(a', b') =$ occlines$(f_{\leq z_s})$.

By induction hypothesis for $f_{\leq z_s}$ and $f_{>z_s}$, we get

$$\begin{aligned} |f| = |f_{\leq z_s}| + |f_{>z_s}| &\geq \mathrm{n}_{\min}(a', b', 0) + \mathrm{n}_{\min}(a - a', b - b', m_{\text{no}} - 1) \\ &= a' + b' - 1 + (a - a') + (b - b') - 1 - (m_{\text{no}} - 1) \\ &= a + b - 1 - m_{\text{no}}. \end{aligned}$$

□

For convenience, we define the following bounds on the number of non-overlapping rows:

$$\mathrm{no}_{\min}(n, a, b) = \min\{m_{\text{no}} \mid 0 \leq m_{\text{no}} \leq \min(a, b) - 1 \wedge n \geq \mathrm{n}_{\min}(a, b, m_{\text{no}})\}$$
$$\mathrm{no}_{\max}(n, a, b) = \max\{m_{\text{no}} \mid 0 \leq m_{\text{no}} \leq \min(a, b) - 1 \wedge n \leq \mathrm{n}_{\max}(a, b, m_{\text{no}})\}$$

Proposition 4.4.8
For any cavity-free coloring f with occlines$(f) = (a, b)$ *and* $|f| = n$ *holds* $\mathrm{no}_{\min}(n, a, b) \leq$ #non-overlaps$(f) \leq \mathrm{no}_{\max}(n, a, b)$.

Proof
Let f be any cavity-free coloring f with occlines$(f) = (a, b)$ and $|f| = n$. First, show that $\mathrm{no}_{\min}(n, a, b) \leq$ #non-overlaps$(f) = m_{\text{no}}$. Since for f holds $n \leq \mathrm{n}_{\max}(a, b, m_{\text{no}})$ and $0 \leq m_{\text{no}} \leq \min(a, b) - 1$ by Lemmata 4.4.4 and 4.4.7, the claim follows by the minimality of $\mathrm{no}_{\min}(n, a, b)$. The second claim, i.e. $\mathrm{no}_{\max}(n, a, b) \geq$ #non-overlaps(f), holds by an analogous argument. □

4.5 Number of i-Points for Cavity-Free Colorings

In the next two sections, we provide a bound on interlayer contacts. As in the case of connected plane colorings, we calculate for a coloring f of plane c the numbers of points having 4,3,2, and 1 contacts to f. These points were called i-points, please recall the definition of i-points from Section 4.3.

As we will show later (Theorem 4.7.2) we can bound the maximal number of interlayer contacts between $x = c$ and $x = c + 1$ if we fill the 4-points first, then (if points are left) the 3-points and so on. Before, we need some definitions and auxiliary lemmata.
In the following, let f be a plane coloring of plane $x = c$ and f' a plane coloring of plane $x = c'$, where $c \neq c'$. The *number of interlayer contacts of* f *and* f' is denoted $\mathrm{IC}_f^{f'}$. We define

$$\text{max-IC}(f; n) = \max \left\{ \mathrm{IC}_f^{f'} \;\middle|\; \begin{array}{c} f' \text{ is a plane coloring of } x = c + 1 \\ \text{with } |f'| = n \end{array} \right\}.$$

We will show that for an arbitrary plane coloring the number of i-points ($i \in \{1, 2, 3, 4\}$) depends only on the number of non-overlapping rows, the number of non-touching rows, and the number of x-steps.
We introduce the functions which compute the number of i-points, depending on $n = |f|$, $s = \mathrm{Surf}_{pl}(f)$, $m_{\mathrm{nx}} = \text{x-steps}(f)$, $m_{\mathrm{no}} = \#\text{non-overlaps}(f)$ and $m_{\mathrm{nt}} = \#\text{non-touchs}(f)$:

$$\begin{aligned}
\#4(n, s, m_{\mathrm{no}}, m_{\mathrm{nt}}, m_{\mathrm{nx}}) &= n - \frac{1}{2}s + 1 + m_{\mathrm{no}} \\
\#2(n, s, m_{\mathrm{no}}, m_{\mathrm{nt}}, m_{\mathrm{nx}}) &= s - 4 - 2\#3(n, s, m_{\mathrm{no}}, m_{\mathrm{nt}}, m_{\mathrm{nx}}) - 3m_{\mathrm{no}} - m_{\mathrm{nt}} \\
\#3(n, s, m_{\mathrm{no}}, m_{\mathrm{nt}}, m_{\mathrm{nx}}) &= m_{\mathrm{nx}} - 2(m_{\mathrm{no}} - m_{\mathrm{nt}}) \\
\#1(n, s, m_{\mathrm{no}}, m_{\mathrm{nt}}, m_{\mathrm{nx}}) &= \#3(n, s, m_{\mathrm{no}}, m_{\mathrm{nt}}, m_{\mathrm{nx}}) + 2m_{\mathrm{no}} + 2m_{\mathrm{nt}} + 4
\end{aligned}$$

The aim of this subsection is to show, that these functions really yield the numbers of i-points of any f, which satisfies the parameters.
For the moment, note that the total number of contacts of any coloring to its succeeding layer is always $4n$ and of course, the given numbers of i-points have to be consistent with this fact. We demonstrate this by simple calculation. Let f be a coloring as postulated.

$$\begin{aligned}
&4\#4(f) + 3\#3(f) + 2\#2(f) + 1\#1(f) \\
&= 4(n - \frac{1}{2}s + 1 + m_{\mathrm{no}}) + 3\#2(n, s, m_{\mathrm{no}}, m_{\mathrm{nt}}, m_{\mathrm{nx}}) \\
&\quad + 2(s - 4 - 2\#3(f) - 3m_{\mathrm{no}} - m_{\mathrm{nt}}) + (\#3(f) + 2m_{\mathrm{no}} + 2m_{\mathrm{nt}} + 4) \\
&= 4n - 2s + 2s + 4 - 8 + 4 + 4m_{\mathrm{no}} - 6m_{\mathrm{no}} + 2m_{\mathrm{no}} - 2m_{\mathrm{nt}} + 2m_{\mathrm{nt}} \\
&\quad + 3\#3(f) - 4\#3(f) - \#3(f) \\
&= 4n.
\end{aligned}$$

As preparation for the main result of the sub-section, we state two lemmas that investigate how to calculate the i-points of f from the two sub-colorings generated by splitting f at a non-overlapping or non-touching row.

LEMMA 4.5.1 (SPLIT 3-POINTS)
Let f be a cavity-free coloring of plane $x = c$ with #non-overlaps$(f) \geq 1$, *and let z_s be a non-overlapping row. Then,* $\#3(f) = \#3(f_{\leq z_s}) + \#3(f_{>z_s})$.

PROOF
We can show that neither $f_{\leq z_s}$, nor $f_{>z_s}$, nor f has a 3-point that lies between rows z_s and $z_s + 1$. This will imply that every 3-point for f lies either below z_s and is therefore also a 3-point for $f_{\leq z_s}$, or above $z_s + 1$ and is therefore also a 3-point for $f_{>z_s}$.
Let f, z_s, $f_{\leq z_s}$ and $f_{>z_s}$ given as defined. Let p be a 3-point for $f_{\leq z_s}$ (resp. $f_{>z_s}$) in plane $x = c + 1$. Then, there are exactly three points out of

$$N_p = \left\{ \begin{pmatrix} c \\ p_y+0.5 \\ p_z+0.5 \end{pmatrix}, \begin{pmatrix} c \\ p_y+0.5 \\ p_z-0.5 \end{pmatrix}, \begin{pmatrix} c \\ p_y-0.5 \\ p_z+0.5 \end{pmatrix}, \begin{pmatrix} c \\ p_y-0.5 \\ p_z-0.5 \end{pmatrix}, \right\}$$

that are contained in $f_{\leq z_s}$ (resp. $f_{>z_s}$). Hence, $p_z + 0.5 \leq z_s$ (resp. $p_z - 0.5 > z_s$) holds. This implies that these 3 points are colored in f, and that the fourth point cannot be colored in f, which implies that p is a 3-point for f.
For the reverse direction, let p be a 3-point for f. Assume p is not a 3-point in $f_{\leq z_s}$ or $f_{>z_s}$. The only case that is not immediately contradicting is $p_z = z_s + 0.5$. Since three of the points in N_p are colored by f, we know that either

$$\begin{pmatrix} c \\ p_y+0.5 \\ p_z+0.5 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} c \\ p_y+0.5 \\ p_z-0.5 \end{pmatrix}$$

or

$$\begin{pmatrix} c \\ p_y-0.5 \\ p_z+0.5 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} c \\ p_y-0.5 \\ p_z-0.5 \end{pmatrix}$$

are colored by f. This is an immediate contradiction to $\neg$ overlaps(f, z_s). □

LEMMA 4.5.2 (SPLIT AT MINIMAL NON-TOUCHING ROW)
Let f be a cavity-free coloring of plane $x = c$ with #non-touchs$(f) \geq 1$, *and let z_s be the minimal non-touching row. Then,*

$$\#\text{non-touchs}(f_{\leq z_s}) = 0, \tag{4.11}$$
$$\#\text{non-touchs}(f_{>z_s}) = \#\text{non-touchs}(f) - 1 \tag{4.12}$$
$$\text{x-steps}(f_{\leq z_s}) + \text{x-steps}(f_{>z_s}) = \text{x-steps}(f), \tag{4.13}$$

and

$$\forall i \in \{1, 2, 3, 4\} : \ \#i(f_{\leq z_s}) + \#i(f_{>z_s}) = \#i(f). \tag{4.14}$$

Proof
Let f and z_s be given as in the lemma.

Eq (4.11)**:** This holds by the minimality of z_s.

Eq (4.12)**:** Consider the z where

$$\exists y : f(c, y, z) \wedge \neg \,\text{touching}(f, z),$$

here called the *non-touching rows in* f. Obviously, all non-touching rows in $f_{>z_s}$ are non-touching rows in f. z_s is the minimal non-touching row in f. Since $z_s <$ min-z$(f_{>z_s})$, we know that z_s is not an non-touching row in $f_{>z_s}$. It remains to show that all non-touching rows $z > z_s$ in f are non-touching rows in $f_{>z_s}$. This holds, since

$$\forall c, y, z : z > z_s \wedge f(c, y, z) \iff f_{>z_s}(c, y, z)$$

by definition of $f_{>z_s}$.

Eq (4.13)**:** We show that whenever a triple of points defines an x-step in f, either all of its points have z-values less or equal than z_s or all points have z-values greater than z_s. We consider all possible violations of this sub-claim. Assume there is an x-step $(\vec{p}_1, \vec{p}_2, \vec{p}_3)$ of f, where

$$\vec{p}_i = (p_i^x, p_i^y, p_i^z) \quad (i = 1, 2, 3),$$

such that $p_1^z = p_2^z = z_s$ and $p_3^z = z_s + 1$. Since $|p_2^y - p_3^y| = 1$ this contradicts $\neg \,\text{touching}(f, z_s)$. The case $p_1^z = p_2^z = z_s + 1$ and $p_3^z = z_s$ is analogously contradicting. This implies that every x-step in f is a x-step of either $f_{\leq z_s}$ or $f_{>z_s}$.

Finally, every x-step in $f_{\leq z_s}$ or $f_{>z_s}$ has to be an x-step in f, since in any x-step $(\vec{p}_1, \vec{p}_2, \vec{p}_3)$ the uncolored p_1 and the colored p_2 have equal z-values by definition. This implies for a x-step of $f_{\leq z_s}$ that $p_2^z \leq z_s$ and analogously for $f_{>z_s}$. Hence, in any case of a x-step $(\vec{p}_1, \vec{p}_2, \vec{p}_3)$ the uncolored $\vec{p}_1$ is also uncolored in f.

Eq (4.14)**:** We will show that for every $i = 1, \ldots, 4$, and for every i-point $\vec{p}$ the neighbors $\vec{p}_1, \ldots, \vec{p}_i$ of p in f are either all in coloring $f_{\leq z_s}$ or all in coloring $f_{>z_s}$. For $i = 1$, this is trivial.

For $i \geq 2$, assume that there are two points $\vec{p}$ and $\vec{p}'$ of the points $\vec{p}_1, \ldots, \vec{p}_i$, where one is in row $z = z_s$ and one is in row $z = z_s + 1$. By definition of i-points, we know that $p^y - p'^y$ is either 0 or 1, which is an immediate contradiction to $\neg \,\text{touching}(f, z_s)$.

□

Before giving the numbers of i-points of general, cavity-free plane colorings, we study the numbers for touching colorings as a preparation. The following lemma was already given in [Bac04].

LEMMA 4.5.3
Let f be a cavity-free, touching coloring, where $s = \text{Surf}_{pl}(f)$, $n = |f|$, $m_{\text{no}} = \#\text{non-overlaps}(f)$, and $m_{\text{nx}} = \text{x-steps}(f)$. Then,

$$\begin{aligned}
\#4(f) =& n - \frac{1}{2}s + 1 + m_{\text{no}} \\
\#3(f) =& m_{\text{nx}} - 2m_{\text{no}} \\
\#2(f) =& s - 4 - 2\#3(f) - 3m_{\text{no}} \\
\#1(f) =& \#3(f) + 4 + 2m_{\text{no}}.
\end{aligned}$$

We proof this lemma using Lemma 4.3.3, which handles i-points in the case of connected, i.e. overlapping colorings.
PROOF
We show the claims by induction over the number of non-overlapping rows. Let f be a touching, cavity-free coloring, where $s = \text{Surf}_{pl}(f)$, $n = |f|$, $m_{\text{no}} = \#\text{non-overlaps}(f)$, and $m_{\text{nx}} = \text{x-steps}(f)$.

Case $m_{\text{no}} = 0$. The claims hold by Lemma 4.3.3.

Case $m_{\text{no}} > 0$. We choose z_s minimal, such that the row z_s in f is non-overlapping, i.e. occ-z(f, z_s) and $\neg$ overlaps(f, z_s). Then, we split f at the row z_s into colorings $f_{\leq z_s}$ and $f_{>z_s}$. Note that Lemma 4.4.2 is applicable in this case.
It is easy to see that for all points $\vec{p} = (p^x, p^y, p^z)$, where $p^z \leq z_s$ ($p^z > z_s$), $\vec{p}$ is an i-points of f if and only if it is an i-point of $f_{\leq z_s}$ ($f_{>z_s}$), respectively. Since the row z_s of f is non-overlapping, but touching, there are exactly two points $\vec{p}_0 \in f_{=z_s}$ and $\vec{p}_1 \in f_{=z_s+1}$, where $p_0^y - p_1^y = 1$ and the two rows $f_{=z_s}$ and $f_{=z_s+1}$ have the arrangement shown in Figure 4.8 (or the symmetrical one).
The point $(c+1, \frac{1}{2}(p_0^y + p_1^y), z_s + 0.5)$ is a 2-point in f and an 1-point in both colorings $f_{\leq z_s}$ and $f_{>z_s}$. All other points $(c+1, y, z_s + 0.5)$ are and i-point of either $f_{\leq z_s}$ or $f_{>z_s}$ if and only if they are an i-point of f. Furthermore, the coloring f has exactly two x-steps that do not occur in any of the sub-colorings $f_{\leq z_s}$ and $f_{>z_s}$.

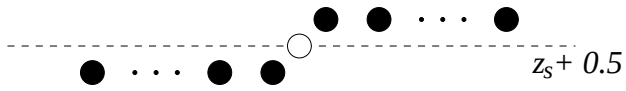


Figure 4.8: Arrangement of the rows $f_{=z_s}$ and $f_{=z_s+1}$. The white point is a two-point of the joint coloring f, but is a one-point in each of the sub-colorings $f_{\leq z_s}$ and $f_{>z_s}$.

Concluding, we get

$$\begin{aligned}
\#1(f) &= \#1(f_{\leq z_s}) + \#1(f_{>z_s}) - 2,\\
\#2(f) &= \#2(f_{\leq z_s}) + \#2(f_{>z_s}) + 1,\\
\#3(f) &= \#3(f_{\leq z_s}) + \#3(f_{>z_s}), \text{ and}\\
\#4(f) &= \#4(f_{\leq z_s}) + \#4(f_{>z_s}).
\end{aligned}$$

Using the induction hypothesis for $f_{\leq z_s}$ and $f_{>z_s}$ we show the claims by rewriting the formulae. Note that due to Lemma 4.4.2 and Proposition 4.2.1,

$$\mathrm{Surf}_{pl}(f_{\leq z_s}) + \mathrm{Surf}_{pl}(f_{>z_s}) = \mathrm{Surf}_{pl}(f).$$

$$\begin{aligned}
&\#1(f_{\leq z_s}) + \#1(f_{>z_s}) - 2\\
&\overset{\text{(ind.hyp.)}}{=} \#1(f_{\leq z_s}) + 4 + \#1(f_{>z_s}) + 4 + 2(m_{\mathrm{no}} - 1) - 2\\
&= \#3(f) + 4 + 2m_{\mathrm{no}}.
\end{aligned}$$

$$\begin{aligned}
&\#2(f_{\leq z_s}) + \#2(f_{>z_s}) + 1\\
&\overset{\text{(ind.hyp.)}}{=} \mathrm{Surf}_{pl}(f_{\leq z_s}) - 7 - 2\#2(f_{\leq z_s})\\
&+ \mathrm{Surf}_{pl}(f_{>z_s}) - 2\#2(f_{>z_s}) - 3(m_{\mathrm{no}} - 1)\\
&= \mathrm{Surf}_{pl}(f) - 4 - 2\#3(f) - 3m_{\mathrm{no}}
\end{aligned}$$

$$\begin{aligned}
&\#3(f_{\leq z_s}) + \#3(f_{>z_s})\\
&\overset{\text{(ind.hyp.)}}{=} m_{\mathrm{nx}} + 2 - 2(m_{\mathrm{no}} - 1)\\
&= m_{\mathrm{nx}} - 2m_{\mathrm{no}}.
\end{aligned}$$

$$\begin{aligned}
&\#4(f_{\leq z_s}) + \#4(f_{>z_s}) \\
&\overset{\text{(ind.hyp.)}}{=} \quad |f_{\leq z_s}| - \frac{1}{2}\text{Surf}_{pl}(f_{\leq z_s}) + 1 + 0 \\
&\quad + \quad |f_{\leq z_s}| - \frac{1}{2}\text{Surf}_{pl}(f_{>z_s}) + 1 + (m_{\text{no}} - 1) \\
&\quad = \quad n - \frac{1}{2}s + 1 + m_{\text{no}}
\end{aligned}$$

□

Lemma 4.5.4
Let f be a cavity-free coloring. Then

$$\forall i \in \{1,2,3,4\} : \#i(f) = \#i(n, s, m_{\text{no}}, m_{\text{nt}}, m_{\text{nx}}),$$

where $n = |f|$, $s = \text{Surf}_{pl}(f)$, $m_{\text{nx}} = \text{x-steps}(f)$, $m_{\text{no}} = \#\text{non-overlaps}(f)$ *and* $m_{\text{nt}} = \#\text{non-touchs}(f)$

Recall, that in a more more explicit form, the lemma claims that

$$\begin{aligned}
&\#4(f) = n - \frac{1}{2}s + 1 + m_{\text{no}} && \#3(f) = m_{\text{nx}} - 2(m_{\text{no}} - m_{\text{nt}}) \\
&\#2(f) = s - 4 - 2\#3(f) - 3m_{\text{no}} - m_{\text{nt}} && \#1(f) = \#3(f) + 2m_{\text{no}} + 2m_{\text{nt}} + 4.
\end{aligned}$$

Proof
We prove the claims by induction on $\#\text{non-touchs}(f)$.

Case $\#\text{non-touchs}(f) = 0$**.** This holds, due to Lemma 4.5.3.

Case $\#\text{non-touchs}(f) > 0$**.** There is at least one non-touching row (which is also a non-overlapping row by definition). Let $\text{min-z}(f) \leq z_s < \text{max-z}(f)$ be minimal, such that $\text{occ-z}(f, z_s)$ and $\neg\,\text{touching}(f, z_s)$.
Using Lemmata 4.4.2, 4.5.1 and 4.5.2 and the induction hypothesis for $f_{\leq z_s}$ and $f_{>z_s}$, we show that the claims for $i \in \{1,2,3,4\}$ hold for f.
By Lemma 4.4.2 and Proposition 4.2.1, we get immediately $\text{Surf}_{pl}(f_{\leq z_s}) + \text{Surf}_{pl}(f_{>z_s}) = \text{Surf}_{pl}(f)$.

Claim for $i = 4$.

$$\begin{aligned}
&\#4(f_{\leq z_s}) + \#4(f_{>z_s}) \\
&\overset{\text{(ind.hyp.)}}{=} |f_{\leq z_s}| - \frac{1}{2}\text{Surf}_{pl}(f_{\leq z_s}) + 1 + \#\text{non-overlaps}(f_{\leq z_s}) \\
&\quad + \quad |f_{>z_s}| - \frac{1}{2}\text{Surf}_{pl}(f_{>z_s}) + 1 + \#\text{non-overlaps}(f_{>z_s}) \\
&\quad = \quad |f| - \frac{1}{2}\text{Surf}_{pl}(f) + 2 + \#\text{non-overlaps}(f) - 1.
\end{aligned}$$

Claim for $i = 3$.

$$\begin{aligned}
&\#3(f_{\leq z_s}) + \#3(f_{>z_s}) \\
&\overset{\text{(ind.hyp.)}}{=} \text{x-steps}(f_{\leq z_s}) - 2(\#\text{non-overlaps}(f_{\leq z_s}) - \#\text{non-touchs}(f_{\leq z_s})) \\
&\quad + \quad \text{x-steps}(f_{>z_s}) - 2(\#\text{non-overlaps}(f_{>z_s}) - \#\text{non-touchs}(f_{>z_s})) \\
&\quad = \quad \text{x-steps}(f) - 2(\#\text{non-overlaps}(f) - 1 - (\#\text{non-touchs}(f) - 1)) \\
&\quad = \quad \text{x-steps}(f) - 2(\#\text{non-overlaps}(f) - \#\text{non-touchs}(f)).
\end{aligned}$$

Claim for $i = 2$.

$$\begin{aligned}
&\#2(f_{\leq z_s}) + \#2(f_{>z_s}) \\
&\overset{\text{(ind.hyp.)}}{=} \text{Surf}_{pl}(f_{\leq z_s}) - 4 - 2\#3(f_{\leq z_s}) \\
&\quad - \quad 3\#\text{non-overlaps}(f_{\leq z_s}) - \#\text{non-touchs}(f_{\leq z_s}) \\
&\quad + \quad \text{Surf}_{pl}(f_{>z_s}) - 4 - 2\#3(f_{>z_s}) \\
&\quad - \quad 3\#\text{non-overlaps}(f_{>z_s}) - \#\text{non-touchs}(f_{>z_s}) \\
&\quad = \quad \text{Surf}_{pl}(f) - 8 - 2\#3(f) \\
&\quad - \quad 3(\#\text{non-overlaps}(f) - 1) - (\#\text{non-touchs}(f) - 1) \\
&\quad = \quad \text{Surf}_{pl}(f) - 4 - 2\#3(f) - 3\#\text{non-overlaps}(f) - \#\text{non-touchs}(f).
\end{aligned}$$

Claim for $i = 1$.

$$\begin{aligned}
&\#1(f_{\leq z_s}) + \#1(f_{>z_s}) \\
&\overset{\text{(ind.hyp.)}}{=} \#3(f_{\leq z_s}) + 2\#\text{non-overlaps}(f_{\leq z_s}) + 2\,\#\text{non-touchs}(f_{\leq z_s}) + 4 \\
&\quad + \quad \#3(f_{>z_s}) + 2\#\text{non-overlaps}(f_{>z_s}) + 2\,\#\text{non-touchs}(f_{>z_s}) + 4 \\
&\quad = \quad \#3(f) + 2(\#\text{non-overlaps}(f) - 1) + 2(\#\text{non-touchs}(f) - 1) + 8 \\
&\quad = \quad \#3(f) + 2\#\text{non-overlaps}(f) + 2\,\#\text{non-touchs}(f) + 4.
\end{aligned}$$

□

4.6 Maximal Number of 3-Points

Due to the last lemma, if we consider colorings with given n, a, b, m_{no}, and m_{nt}, then increasing the number of x-steps m_{nx} does not affect the number of 4-points, but increases the number of 3-points and 1-points, while decreasing the number of 2-points. The increase of 3- and 1-points is 1 per x-step, the decrease of 2-points is 2 per 3-point. This pattern, which already occurred in the bound for connected plane colorings (cf. Section 4.3), grants that we maximize the possible number of interlayer contacts to a second plane with a given number of elements if we maximize the number of 3-points in the first plane.
For this purpose, we first show that we do not need to distinguish between non-touching and non-overlapping rows for determining the maximal number of 3-points. The reason is that number of 3-points does not change if one transforms a non-overlapping row into into a non-touching row. Consider as an example the two colorings in Figure 4.9. Then both f and f' have one

Figure 4.9: Change in the number of x-steps, when transforming an non-overlapping, but touching coloring into a non-touching one.

3-point (indicated in grey). By transforming the non-overlapping row in f into a non-touching row, f' looses two x-steps. Thus, for the number of 3-points, the effects of increasing #non-touchs$(\cdot)$ by 1 are completely balanced by decreasing x-steps$(\cdot)$ by 2.
Note that such a bound for the interlayer contacts using a bound for 3-points that does not distinguish between non-overlapping and non-touching rows slightly overestimates, since we assume the best case for the number of 2- and 1-points (note that in contrast to the number of 3-points, the number of 2- and 1-points depend on the exact number of non-touching rows).
We will now develop an upper bound on the number of 3-points of general plane colorings. For overlapping plane colorings f, we use the upper bound

$$\text{B}_{\text{x-steps}}(|f|, \text{occlines-y}(f), \text{occlines-z}(f)) \geq \text{x-steps}(f)$$

on the number of x-steps in f that was already given before (cf. Lemma 4.3.8). We improve this bound in the case of quadratic frames (a, a) and n is not

normal for (a, a). Here, we show that we have an upper bound of $2a - 3$ instead of $2a - 2$ if there is no maximal indent $\vec{i}$ with $n = \mathrm{vol}(a, a, \vec{i})$. We show in this case, that there must be a diagonal cavity.

LEMMA 4.6.1
For every overlapping cavity-free coloring f we get

$$\#3(f) \leq \mathrm{B}_{\#3}(|f|, a, b),$$

where $(a, b) = \mathrm{frame}(f)$ *and*

$$\mathrm{B}_{\#3}(n, a, b) = \begin{cases} \mathrm{B}_{\text{x-steps}}(n, a, b) & n \text{ is normal for frame } (a, b) \\ 2\min(a, b) - 2 & \text{else if } a \neq b \\ 2a - 2 & \text{else if } \exists \vec{i}: \ \vec{i} \text{ are maximal indents} \\ & \qquad \text{for } (a, a) \wedge n = \mathrm{vol}(a, a, \vec{i}) \\ 2a - 3 & \text{otherwise} \end{cases}$$

PROOF
Let f be an overlapping cavity-free coloring with frame (a, b). As previously argued, $\mathrm{B}_{\text{x-steps}}(n, a, b)$ and thus $2\min(a, b) - 2$ are bounds for $\#3(f)$.
For the case $a = b$, $\mathrm{B}_{\#3}(|f|, a, b)$ even improves the bound $2\min(a, b) - 2$. Let f be a coloring with frame (a, a) and indents $\vec{i} = (i_1, i_2, i_3, i_4)$. We have two cases:

$\vec{i}$ is not maximal. Then $i_1 + i_2 + i_3 + i_4 \leq 2\min(a, b) - 3$, which implies by Lemma 4.3.5 that

$$\#3(f) \leq i_1 + i_2 + i_3 + i_4 \leq 2\min(a, b) - 3.$$

$\vec{i}$ is maximal. Then $i_1 + i_2 + i_3 + i_4 = 2\min(a, b) - 2$. First, we show that

$$n \neq \mathrm{vol}(a, a, \vec{i}) \iff f \text{ contains diagonal cavity.} \tag{4.15}$$

For the claim (4.15), first note that $i_1 + i_2 + i_3 + i_4 = 2\min(a, b) - 2$ is equivalent to $i_1 + i_2 + i_3 + i_4 = 2a - 2$, since $a = b$. Let $(a, a, i^f_{lb}, i^f_{lu}, i^f_{rb}, i^f_{ru})$ be the detailed frame of f (where $\vec{i}$ is $i^f_{lb}, i^f_{lu}, i^f_{rb}, i^f_{ru}$ ordered by size). We will first show that in the first and last row of f, there is exactly one colored point. For the first row, the equation

$$a = i^f_{lb} + i^f_{rb} + u_b + c_b$$

holds, where c_b are the number of colored points in the first row, and u_b are the uncolored points which are not already excluded by the indents i^f_{lb} and i^f_{rb}. Similarly, we get

$$a = i^f_{lu} + i^f_{ru} + u_u + c_u$$

for the last row. Now

$$2a = i^f_{lb} + i^f_{rb} + u_b + c_b + i^f_{lu} + i^f_{ru} + u_u + c_u.$$

Since $i^f_{lb} + i^f_{rb} + i^f_{lu} + i^f_{ru} = 2a - 2$ by our assumption, we get

$$u_u + c_u + u_b + c_b = 2.$$

Since the first and last row must contain a colored point to justify the frame, we get $c_u \geq 1$ and $c_b \geq 1$. This immediately implies $c_u = 1$ and $c_b = 1$, i.e., the first and last row contains exactly one colored point.
Analogously, we get that the first and last column of f contains exactly one point. Hence, f has the form as described by Figure 4.10. If all points on lines $\vec{p}_1$—$\vec{p}_2$, $\vec{p}_1$—$\vec{p}_3$, $\vec{p}_2$—$\vec{p}_4$, and $\vec{p}_3$—$\vec{p}_4$ are colored by f, then all points in the region surrounded by these lines must be colored by f due to the cavity-freeness of f. But this implies that $|f| = \mathrm{vol}(a, a, \vec{i})$. Otherwise, if not all the points on these lines are colored, we have at least one diagonal cavity and $|f| < \mathrm{vol}(a, a, \vec{i})$. This concludes the proof of (4.15).
Now, by Equation (4.15), $n = \mathrm{vol}(a, a, \vec{i})$ implies that f does not contain a diagonal cavity, which implies by Lemma 4.3.5 that

$$\#3(f) = i_1 + i_2 + i_3 + i_4 = 2\min(a, b) - 2.$$

On the other hand, $n \neq \mathrm{vol}(a, a, \vec{i})$ implies that f contains at least 1 diagonal cavity. Since

$$\#3(f) = i_1 + i_2 + i_3 + i_4 - \mathrm{diagcav}(f)$$

by 4.3.5, we can immediately conclude that

$$\#3(f) \leq i_1 + i_2 + i_3 + i_4 - 1 = 2\min(a, b) - 3.$$

□

The given bound for overlapping colorings is now extended to arbitrary, cavity-free colorings. We define the predicate $\mathrm{valid}(n, a, b, m_{\mathrm{no}})$ by

$$\mathrm{valid}(n, a, b, m_{\mathrm{no}}) = m_{\mathrm{no}} < \min(a, b) \wedge \mathrm{n}_{\min}(a, b, m_{\mathrm{no}}) \leq n \leq \mathrm{n}_{\max}(a, b, m_{\mathrm{no}}).$$

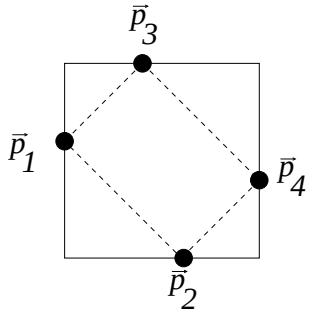


Figure 4.10: Coloring with maximal indents and frame (a, a).

Proposition 4.6.2
For any a, b, n, m_{no}, the existence of a coloring f with $|f| = n$, $\text{occlines}(f) = (a, b)$, and $\#\text{non-overlaps}(f) = m_{\text{no}}$ implies that $\text{valid}(n, a, b, m_{\text{no}})$ holds.

Proof
The claim is an immediate consequence of Lemmata 4.4.6, 4.4.7, and 4.4.4. □

Definition 4.6.3 (Bound on number of 3-points (general case))

$$\mathrm{B}_{\#3}(n, a, b, m_{\text{no}}) = \begin{cases} \bigstar & \text{valid}(n, a, b, m_{\text{no}}) \\ -\infty & \text{otherwise} \end{cases}$$

$$\bigstar = \begin{cases} \mathrm{B}_{\#3}(n, a, b) & m_{\text{no}} = 0 \\ \max \left\{ \begin{array}{c|l} \mathrm{B}_{\#3}(n', a', b', 0) & 1 \le n' \le n-1, \\ +\mathrm{B}_{\#3}(n-n', a-a', & 1 \le a' \le a-1, \\ b-b', m_{\text{no}}-1) & 1 \le b' \le b-1, \end{array} \right\} & \text{otherwise} \end{cases}$$

Lemma 4.6.4
For every cavity-free coloring f, holds

$$\#3(f) \le \mathrm{B}_{\#3}(n, a, b, m_{\text{no}}),$$

where

$$n = |f|, (a, b) = \text{occlines}(f), \text{ and } m_{\text{no}} = \#\text{non-overlaps}(f).$$

Proof
We proof the claim by induction on m_{no}.

Case $m_{\text{no}} = 0$**.** Here, the claim is satisfied by Lemma 4.6.1.

Case $m_{\text{no}} > 0$**.** Either there is no coloring for n, a, b, m_{no}, then nothing is to show, or there is at least one such coloring. In this case, let f be a cavity-free coloring with $|f| = n$, occlines$(f) = (a, b)$, and #non-overlaps$(f) = m_{\text{no}}$. We have to show, that there is a triple (n', a', b'), where

$$1 \leq n' \leq n-1,\ 1 \leq a' \leq a-1,\ \text{and } 1 \leq b' \leq b-1,$$

such that

$$\mathrm{B}_{\#3}(n', a', b', 0) + \mathrm{B}_{\#3}(n-n', a-a', b-b', m_{\text{no}}-1) \geq \#3(f).$$

Split the coloring f into two colorings $f_{\leq z_s}$ and $f_{>z_s}$ at the minimal line z_s with occ-z(f, z_s) and $\neg$ overlaps(f, z_s). Let

$$n' = |f_{\leq z_s}| \text{ and } (a', b') = \text{occlines}(f_{\leq z_s}).$$

Then, by Lemma 4.4.2, this implies that

$$n-n' = |f_{>z_s}| \text{ and } (a-a', b-b') = \text{occlines}(f_{>z_s}).$$

Since $f_{\leq z_s}$ and $f_{>z_s}$ are cavity-free colorings, valid$(f_{\leq z_s})$ and valid$(f_<)$ hold. Now, we get from the induction hypothesis for $f_{\leq z_s}$ and $f_{>z_s}$ that

$$\begin{gathered}\mathrm{B}_{\#3}(n', a', b', 0) + \mathrm{B}_{\#3}(n-n', a-a', b-b', m_{\text{no}}-1) \\ \geq \\ \#3(f_{\leq z_s}) + \#3(f_{>z_s}).\end{gathered}$$

By Lemma 4.5.1, $\#3(f_{\leq z_s}) + \#3(f_{>z_s}) = \#3(f)$. □

4.7 Bound on Interlayer Contacts

The bound on the number of 3-points can now be used to derive a bound on the number of interlayer contacts for arbitrary colorings, which is defined as follows.
First, we define a bound in the case of known number of non-overlapping rows and afterwards, apply this bound to get a general bound.

Definition 4.7.1 (General Interlayer Contacts Bound)

$$\begin{aligned}\mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2)(m_{\mathrm{no1}}) = {} & 4\min(n_2, \#4) \\ & +3\min(\#3, \max(n_2 - \#4), 0) \\ & +2\min(\#2, \max(n_2 - \#4 - \#3, 0)) \\ & +1\min(\#1, \max(n_2 - \#4 - \#3 - \#2, 0))\end{aligned}$$

where

$$\begin{aligned}\#4 &= n - a_1 - b_1 + 1 + m_{\mathrm{no1}} \\ \#3 &= \mathrm{B}_{\#3}(n_1, a_1, b_1, m_{\mathrm{no1}}) \\ \#2 &= 2(a_1 + b_1) - 4 - 2\#3 - 3m_{\mathrm{no1}} \\ \#1 &= \#3 + 2m_{\mathrm{no1}} + 4\end{aligned}$$

$$\mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2) = \max\left\{ \mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2)(m_{\mathrm{no1}}) \;\middle|\; \begin{array}{c} \mathrm{no}_{\min}(n_1, a_1, b_1) \\ \leq m_{\mathrm{no1}} \leq \\ \mathrm{no}_{\max}(n_1, a_1, b_1) \end{array} \right\}$$

Theorem 4.7.2
Let f_1 and f_2 be coloring of planes $x = c$ and $x = c + 1$, respectively. Let

$$n_1 = |f_1|,\ \mathrm{occlines}(f_1) = (a_1, b_1),\ |f_2| = n_2 \text{ and } \mathrm{occlines}(f_2) = (a_2, b_2).$$

Then

$$\mathrm{IC}_{f_1}^{f_2} \leq \min(\mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2), \mathrm{B}_{\mathrm{ILC}}(n_2, a_2, b_2; n_1)).$$

For convenience, we define

$$\mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2) = \min(\mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2), \mathrm{B}_{\mathrm{ILC}}(n_2, a_2, b_2; n_1)).$$

Proof
Let f_1 and f_2 be colorings with $n_1 = |f_1|$, $(a_1, b_1) = \mathrm{occlines}(f_1)$, $(a_2, b_2) = \mathrm{occlines}(f_2)$ and $n_2 = |f_2|$. Let $m_{\mathrm{no1}} = \#\text{non-overlaps}(f_1)$ and $m_{\mathrm{nt1}} = \#\text{non-touchs}(f_1)$. We will first show that

$$\mathrm{IC}_{f_1}^{f_2} \leq \mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2).$$

Once this is shown, the claim follows immediately since $\mathrm{IC}_{f_1}^{f_2} = \mathrm{IC}_{f_2}^{f_1}$.

Since f_1 is cavity-free, we know by Proposition 4.2.1 that the surface of f_1 is $2(a_1 + b_1)$. Define

$$\begin{aligned}\text{max-IC}(n_1, a_1, b_1, \#3, m_{\text{no}1}, m_{\text{nt}1}; n_2) = \\ 4\min(n_2, \#4) \\ +3\min(\#3, \max(n_2 - \#4), 0) \\ +2\min(\#2, \max(n_2 - \#4 - \#3, 0)) \\ +1\min(\#1, \max(n_2 - \#4 - \#3 - \#2, 0))\end{aligned}$$

where

$$\begin{aligned}\#4 &= n_1 - a_1 - b_1 + 1 + m_{\text{no}1} \\ \#2 &= 2(a_1 + b_1) - 4 - 2\#3 - 3m_{\text{no}1} - m_{\text{nt}1} \\ \#1 &= \#3 + 2m_{\text{no}1} + 2m_{\text{nt}1} + 4.\end{aligned}$$

By Lemma 4.5.4,

$$\begin{aligned}\text{IC}_{f_1}^{f_2} &\leq \text{max-IC}(f_1; n_2) \\ &= \text{max-IC}(n_1, a_1, b_1, \#3(f_1), m_{\text{no}1}, m_{\text{nt}1}; n_2).\end{aligned}$$

Now, if we relax the constraints on the parameters of f_1 by $\#3 \leq \text{B}_{\#3}(f_1)$ and $m_{\text{nt}1} > 0$, the term

$$\text{max-IC}(n_1, a_1, b_1, \#3, m_{\text{no}1}, m_{\text{nt}1}; n_2)$$

is maximized for $\#3 = \text{B}_{\#3}(f_1)$ and $m_{\text{nt}1} = 0$. Hence,

$$\begin{aligned}\text{max-IC}(f_1; n_2) &\leq \text{max-IC}(n_1, a_1, b_1, \text{B}_{\#3}(f_1), m_{\text{no}1}, 0; n_2) \\ &= \text{B}_{\text{ILC}}(n_1, a_1, b_1; n_2)(m_{\text{no}1}).\end{aligned}$$

Finally, this implies the claim of the theorem, since we maximize over $m_{\text{no}1}$ in the definition of $\text{B}_{\text{ILC}}(n_1, a_1, b_1; n_2)$. □

The bound on interlayer contacts is now used (together with the tight bound on layer contacts) for bounding the number of contacts in colorings with a given frame sequence.

Definition 4.7.3
Let $((n_1, a_1, b_1), \ldots, (n_k, a_k, b_k))$ *be a frame sequence. Then,*

$$\begin{aligned}&\text{B}_{\text{fr}}((n_1, a_1, b_1), \ldots, (n_k, a_k, b_k)) = \\ &\sum_{i=1}^{k} \big(\text{B}_{\text{LC}}(n_i, a_i, b_i) + \text{B}_{\text{ILC}}(n_i, a_i, b_i; n_{i+1}, a_{i+1}, b_{i+1})\big) \\ &+ \text{B}_{\text{LC}}(n_k, a_k, b_k).\end{aligned}$$

THEOREM 4.7.4
Let f be a finite point set of the face-centered cubic lattice with frame sequence $((a_1, b_1, n_1), \ldots, (a_k, b_k, n_k))$. Then,

$$\text{contacts}(f) \leq \mathrm{B}_{\text{fr}}((a_1, b_1, n_1), \ldots, (a_k, b_k, n_k)).$$

4.8 Generating Frame Sequences

In this section, we will develop an efficient method to compute the set of all frame sequences of a given size n, where the bound B_{fr} is greater or equal than a given number of contacts c.
In the section, we will overload the function B_{fr} twice, which was previoulsy defined for frame sequences. In all its forms, the function yields an upper bound on the number of contacts for a class of point sets, where the bound is based on frame sequences. Analogously, we will overload the function B_{num}, which denotes bounds based on number sequences.

DEFINITION 4.8.1 (FRAME SEQUENCE SET)
For $n, c \in \mathbb{N}$, we define

$$\text{FrameSeqs}(n, c) = \left\{ \; s_{\text{fr}} \textit{ frame sequence} \;\middle|\; \text{size}(s_{\text{fr}}) = n \textit{ and } \mathrm{B}_{\text{fr}}(s_{\text{fr}}) \geq c \; \right\}.$$

$\text{FrameSeqs}(n, c)$ contains all possible frame sequences for point sets of size n with at least c contacts.

PROPOSITION 4.8.2
For $n, c \in \mathbb{N}$,

$$\text{FrameSeqs}(n, c) \supseteq \left\{ \begin{array}{c} s_{\text{fr}} \textit{ frame sequence} \\ \textit{of connected point set } P \end{array} \;\middle|\; \begin{array}{c} |P| = n \\ \textit{and } \text{contacts}(P) \geq c \end{array} \right\}.$$

Notably, there is a naive method for constructing $\text{FrameSeqs}(n, c)$, namely enumerating all frame sequences of size n and check their bound. Due to the following proposition, this method works, however is infeasible.

PROPOSITION 4.8.3
The set of number sequences (frame sequences) of size n is finite. The size of the set is exponential in n.

PROOF
Recall that number sequences and frame sequences are only defined for connected point sets. Hence, for a number sequence $(n_1, \ldots, n_k)$ as well as for

a frame sequence $((n_1, a_1, b_1), \ldots, (n_k, a_k, b_k))$, every n_i, for $1 \leq i \leq k$, is at least 1. There are

$$\sum_{k=1}^{n} \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) = 2^{n-1}$$

many number sequences of size n, which is easily verified by induction over n. In a frame sequence $((n_1, a_1, b_1), \ldots, (n_k, a_k, b_k))$, for every n_i there are finitely many pairs (a_i, b_i), since $0 < a_i, b_i < n$. □

First, we focus on computing FrameSeqs(n, c), where c is the maximal number of contacts such that a frame sequence of size n with the bound c exists. Denote this number $\mathrm{B}_{\mathrm{fr}}(n)$, it is formally defined as

$$\mathrm{B}_{\mathrm{fr}}(n) = \operatorname*{argmax}_{c}(\text{FrameSeqs}(n, c) \neq \emptyset).$$

Later, we extend the method to compute FrameSeqs(n, c) for values of c that are less than maximal.

The reasons for concentrating on maximal and near-maximal contacts bounds are twofold. On the one hand, this is the common case, when we finally want to construct hydrophobic cores for protein structure prediction. On the other hand, only in this case, there are much less frame sequences with a bound of at least c than there are frame sequences in total. Hence, for c much smaller than $\mathrm{B}_{\mathrm{fr}}(n)$, the naive approach cannot be improved significantly, due to the raw number of frame sequences.

The very first step is computing $\mathrm{B}_{\mathrm{fr}}(n)$, which is an upper bound for the number of contacts in a point set of size n. Therefore we develop a recursive formula, which is then efficiently evaluated by materializing intermediary results, i.e. dynamic programming.

The bound on frame sequence (cf. Definition 4.7.3) can be rewritten using recursion as

$$\begin{aligned} \mathrm{B}_{\mathrm{fr}}((n_1, a_1, b_1)) &= \mathrm{B}_{\mathrm{LC}}(n_1, a_1, b_1) \\ \mathrm{B}_{\mathrm{fr}}((n_1, a_1, b_1), (n_2, a_2, b_2), \ldots, (n_k, a_k, b_k)) &= \mathrm{B}_{\mathrm{LC}}(n_1, a_1, b_1) \\ &\quad + \mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2) \\ &\quad + \mathrm{B}_{\mathrm{fr}}((n_2, a_2, b_2), \ldots, (n_k, a_k, b_k)). \end{aligned} \tag{4.16}$$

From this equation, we derive the recursive formula for computing $\mathrm{B}_{\mathrm{fr}}(n)$ as kind of a maximization variant.

$$\mathrm{B}_{\mathrm{fr}}(n) = \max_{n_1, a_1, b_1} \mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1) \tag{4.17}$$

where we define

$$\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1) = \mathrm{B}_{\mathrm{LC}}(n_1, a_1, b_1), \text{ if } n = n_1$$

and otherwise

$$\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1) = \max_{n_2, a_2, b_2} \begin{pmatrix} & \mathrm{B}_{\mathrm{LC}}(n_1, a_1, b_1) \\ + & \mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2) \\ + & \mathrm{B}_{\mathrm{fr}}(n - n_1; n_2, a_2, b_2) \end{pmatrix}.$$

Implicitly, for the maximization only reasonable values n_1, a_1, b_1 (n_2, a_2, b_2, respectively) are considered. Which triples n_1, a_1, b_1 are reasonable was broadly discussed in Sub-section 4.4. Note how we reduce the rest of the sequence $(n_2, a_2, b_2), \ldots, (n_k, a_k, b_k)$ to the length n of the sequence, while we replace the bound on a concrete frame sequence by the maximal bound of all frame sequences with a given size. This recursion equation was already sketched in Chapter 3, where Figure 3.9 served as illustration.
Interestingly, our presentation of B_{fr} hides some complexity. Namely, the evaluation of $\mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2)$ requires to evaluate a further recursion equation for the maximal number of 3-points. This equation is handled rather similar as the main recursion and is otherwise not mentioned further.
Similar things can be done using the bound on number sequences. We define $\mathrm{B}_{\mathrm{num}}(n)$ for number sequences in analogy to $\mathrm{B}_{\mathrm{fr}}(n)$ for frame sequences, then

$$\mathrm{B}_{\mathrm{num}}(n) = \max_{n_1} \mathrm{B}_{\mathrm{num}}(n; n_1) \tag{4.18}$$

where $\mathrm{B}_{\mathrm{num}}(n; n_1)$ is defined for $n = n_1$ as

$$\mathrm{B}_{\mathrm{num}}(n; n_1) = \mathrm{B}_{\mathrm{LC}}(n_1, a, b), \text{ where } a^{\mathrm{m}} = \lceil \sqrt{n_k} \rceil,\ b^{\mathrm{m}} = \lceil n_k / a^{\mathrm{m}} \rceil$$

and otherwise

$$\mathrm{B}_{\mathrm{num}}(n; n_1) = \max_{a_1, b_1, n_2} \begin{pmatrix} & \mathrm{B}_{\mathrm{LC}}(n_1, a_1, b_1) \\ + & \mathrm{B}^{\mathrm{c}}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2) \\ + & \mathrm{B}_{\mathrm{num}}(n - n_1; n_2) \end{pmatrix}.$$

Notably, $\mathrm{B}_{\mathrm{num}}$ yields an upper bound on B_{fr}, namely for all reasonable n_1, a_1, b_1 that describe a layer and $n \geq n_1$,

$$\mathrm{B}_{\mathrm{num}}(n; n_1) \geq \mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1).$$

4.8.1 Lazy Dynamic Programming

We implement B_{fr} by lazy dynamic programming. Here, we assume some familiarity with the technique of dynamic programming. Our method differs from classical dynamic programming in two important aspects, namely

- we compute $B_{fr}(n; n_1, a_1, b_1)$ lazily, i.e. only on demand, and
- we use a dynamic data structure for storing the intermediary results $B_{fr}(n; n_1, a_1, b_1)$. For implementing this structure, we suggest the use of a hash, which is indexed by tuples (n, n_1, a_1, b_1).

By the first aspect, we do not need to impose a certain order in which intermediary results are computed. However, in the same time we guarantee that only those values are computed that really contribute to the final result. This is achieved as follows. First, we ask for the final result. Then, each time an actual value of the function is requested, we look up in the data structure if the value is already known. Only if it is unknown, the value is computed. If for this computation further values of the function are needed, then these requests are evaluated following the same strategy. Finally, when a function value gets known, it is materialized in the data structure.
By the second aspect, we control the space requirements of the algorithm. By using a dynamic data structure, we allocate exactly the space that is required for storing intermediary values. Nothing has to be known about these requirements and the distribution of necessary function values in advance. These benefits come at the comparably small cost of using a hash instead of an array as data structure.
Note that in this way, the dynamic programming algorithm is very close to the original recursion equation. The evaluation order and organization of the storage is derived automatically during the run-time of the program. Thus, we totally avoid an important source of errors and possible reason for inefficiency with respect to time and space.

4.8.2 Bounded Dynamic Programming

The technique of lazy dynamic programming gets especially valuable, when we speed up the computation by bounding B_{fr} with B_{num}. Now, this is achieved by extending the previous computation scheme. Whenever a value $B_{fr}(n; n_1, a_1, b_1)$ is needed, this occurs always in a maximization. During such a maximization, it is possible to calculate a bound c such that whenever

$\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1) \leq c$, the actual value of $\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1)$ does not contribute to the maximum. In many cases, where

$$\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1) \leq c$$

we can already derive this from

$$\mathrm{B}_{\mathrm{num}}(n; n_1) \leq c.$$

Then, we can completely omit the computation of $\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1)$ due to our lazy implementation technique. Since $\mathrm{B}_{\mathrm{num}}(n; n_1)$ is computed much more efficiently than $\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1)$ this reduces time and space requirements significantly.

4.8.3 Sub-Optimal Trace-Back

The set of frame sequences s_{fr} of size n, where the bound is maximal, are derived by trace-back using the dynamic data structure, which stores the intermediary results. Note that values that are not already present in the data structure can be computed lazily as described above.
Assume that the maximal bound $c_{\max} = \mathrm{B}_{\mathrm{fr}}(n)$ is already computed. We start with all tuples (n, n_1, a_1, b_1), where $\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1) = c_{\max}$. These tuples are part of frame sequences in $\mathrm{FrameSeqs}(n, c_{\max})$.
For every tuple (n, n_1, a_1, b_1), there are tuples $(n - n_1, n_2, a_2, b_2)$, which are used for computing the value $\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1)$. All tuples $(n - n_1, n_2, a_2, b_2)$, where

$$\begin{aligned}&\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1)\\ &= \mathrm{B}_{\mathrm{LC}}(n_1, a_1, b_1) + \mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2) + \mathrm{B}_{\mathrm{fr}}(n - n_1; n_2, a_2, b_2)\end{aligned}$$

extend frame sequences in $\mathrm{FrameSeqs}(n, c_{\max})$. With these tuples, we proceed in a recursive manner.
However, our final aim is the computation of sets $\mathrm{FrameSeqs}(n, c_{\max} - s)$, where s is a (usually small) number $0 \leq s \leq c_{\max}$. In extension to the previous method, we start with all tuples (n, n_1, a_1, b_1), where $\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1) \geq c_{\max} - s$ and decrease s by the difference of $\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1)$ and $c_{\max}$.
For every tuple (n, n_1, a_1, b_1), we extend the frame sequences by all tuples $(n - n_1, n_2, a_2, b_2)$, where

$$\begin{aligned}&\mathrm{B}_{\mathrm{fr}}(n; n_1, a_1, b_1)\\ &\geq \mathrm{B}_{\mathrm{LC}}(n_1, a_1, b_1) + \mathrm{B}_{\mathrm{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2) + \mathrm{B}_{\mathrm{fr}}(n - n_1; n_2, a_2, b_2) - s.\end{aligned}$$

With these tuples, we proceed in a recursive manner, where we now decrease s by the difference of

$$\mathrm{B_{LC}}(n_1, a_1, b_1) + \mathrm{B_{ILC}}(n_1, a_1, b_1; n_2, a_2, b_2) + \mathrm{B_{fr}}(n - n_1; n_2, a_2, b_2)$$

and $\mathrm{B_{fr}}(n; n_1, a_1, b_1)$.
By and large, our sub-optimal trace-back strategy distributes the deviation from the maximal bound over the trace-back steps such that the total deviation does not exceed the initial value of s.

4.8.4 Results

We have implemented the dynamic programming algorithm for generating frame sequences in C++. By this implementation, all optimal frame sequences up to size 200 were computed in less than 12 hours on standard hardware (Pentium 4 at 2.4Ghz). In the same time, we computed the bounds $\mathrm{B_{fr}}(n)$, for $n = 1, \ldots, 200$. More results for the computation of optimal and sub-optimal frame sequences are given in Tables 4.2 and 4.1. In [BW01b], we still reported a computation time of 10 days for all (optimal) frame sequences up to 100. The huge speed up in comparison to this older implementation is due to the reported bounding of the computation by the number sequence bound.

size	run-time
50	4s
100	5min
150	100min
200	12h

Table 4.1: Some run-times for the computation of optimal frame sequences.

The plots of Figure 4.11 show the frame sequence bound and the number of optimal frame sequences in relation to the core size. The plots of the figure show the linear dependency of size and frame sequence bound (for larger sizes) and the weak dependency of size and number of optimal frame sequences.
The implementation is equiped for incremental computation. Before terminating, the program stores all materialized values $\mathrm{B_{fr}}(n; n_1, a_1, b_1)$ in a file. The values are read from this file, when the program is started again. Due to this, in later runs of the program all computed values can be reused for computing further frame sequences. Note that such incremental computation is made easy due to the use of lazy dynamic programming.

# contacts	# frame sequences	run-time
244	1	0.5s
243	7	0.5s
242	107	0.6s
241	638	1.3s
240	4,102	4.7s
239	20,436	90s

Table 4.2: Sub-optimal frame sequences of size 60. The table gives for a number of contacts the number of frame sequences and the run-time for their computation.

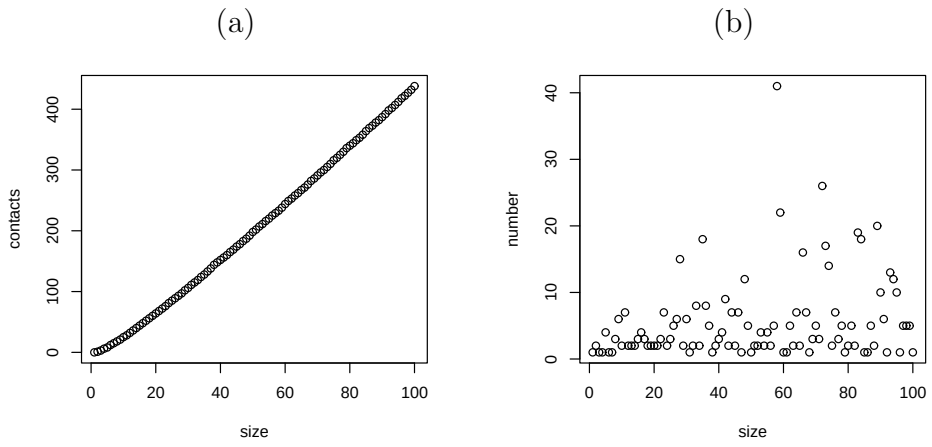


Figure 4.11: Frame sequence size against a) contacts in optimal sequences (frame sequence bound) and b) number of optimal frame sequences.

Chapter 5

Core Construction

This chapter discusses the construction of all connected point sets of the FCC lattice that have a given size n and at least a given number of contacts c. For efficiency of this construction, we employ the set FrameSeqs(n, c), which is computed efficiently due to the previous chapter.

In particular, we are interested in constructing point sets with maximally many contacts or almost maximally many contacts for the given size. Referring to their use as hydrophobic cores for structure prediction, such point sets are often called *cores*. Cores with maximally many contacts are called *optimal cores*.

Notably, we will only deal with connected cores in this chapter. In general, optimal cores are always connected. Hence, for enumerating optimal cores there is no need to handle unconnected cores. However, optimal structures can have sub-optimal and unconnected cores. As we discuss in Chapter 6, we can nevertheless avoid precomputing unconnected cores for structure prediction.

In this chapter, we only describe the method for point sets of the FCC explicitly and do not handle cores of the cubic lattice in more detail. The approach for the cubic lattice shares many similarities with the one for the FCC lattice and is otherwise more straightforward. As its main difference to the FCC appraoch, we employ the extension of CHCC, which was discussed in Section 4.1 of the previous chapter.

In the following, we describe the core construction problem in more detail and define it precisely. Then, after introducing some formal notation, the constraint algorithm for core construction is described. Finally, we give some results. The chapter is partially based on [Wil02].

5.1 Problem Specification

We define the *core construction problem for the FCC* as follows. Given a size n and a number of contacts c, construct the set

$$\text{Cores}(n, c) = \left\{ \; C \subset D_3 \;\middle|\; |C| = n,\ C \text{ connected},\ \text{contacts}(C) \geq c \; \right\}.$$

Note that this set is infinite, due to translation symmetry. Therefore, we will only virtually construct the set $\text{Cores}(n, c)$. The actual problem is to construct this set modulo geometrical symmetries. That is, we ask for only one representative of every equivalence class with respect to translations, rotations and reflections in $\text{Cores}(n, c)$. The core construction problem is a hard combinatorial problem. Therefore, we do not tackle this problem directly without the help of further knowledge.
Instead, we use results from the previous chapter to reformulate the problem. There is a related and more general problem, which is easier to solve directly. This problem is called the *core construction problem using frame sequences* and asks for the set of all cores C with at least c contacts that have a frame sequence out of a given set S in each dimension (x,y, or z), i.e.

$$\text{Cores}(S, c) = \left\{ \; C \subset D_3 \;\middle|\; \begin{array}{c} \text{contacts}(C) \geq c \text{ and} \\ \text{the frame sequence of } C \\ \text{in dimension } \xi \text{ is in } S, \\ \text{for all } \xi \in \{x, y, z\} \end{array} \right\},$$

where we overload the function Cores. Again, we actually ask for the set of cores modulo geometrical symmetries.
Since by the previous chapter

$$\text{FrameSeqs}(n, c) \supseteq \left\{ \; s_{\text{fr}} \text{ frame sequence of } C \;\middle|\; C \in \text{Cores}(n, c) \; \right\},$$

the set $\text{Cores}(n, c)$ is equal to the set $\text{Cores}(\text{FrameSeqs}(n, c), c)$.
In our reformulation of the core construction problem, we ask for $\text{Cores}(S, c)$, given $S = \text{FrameSeqs}(n, c)$, instead of directly asking for $\text{Cores}(n, c)$. We recall that $\text{FrameSeqs}(n, c)$ is efficiently computable, due to the last chapter.

Maximal number of contacts. Besides the core construction problem, we are also interested in the maximal number of contacts in any core of size n, i.e.

$$c_{\max}(n) = \operatorname*{argmax}_{c}(\text{Cores}(n, c) \neq \emptyset)$$

The maximal value for a given size n, where FrameSeqs(n, c) is not empty, was denoted $\mathrm{B}_{\mathrm{fr}}(n)$. $\mathrm{B}_{\mathrm{fr}}(n)$ is computed efficiently due to the last chapter. However, note that in general only

$$c_{\max}(n) \leq \mathrm{B}_{\mathrm{fr}}(n),$$

i.e. $\mathrm{B}_{\mathrm{fr}}(n)$ and $c_{\max}(n)$ are not necessarily equal for arbitrary $n > 0$. However, a solution to the core construction problem allows finding $c_{\max}(n)$. Since, then we can determine $c_{\max}(n)$ by iteratively computing for descending $c \leq \mathrm{B}_{\mathrm{fr}}(n)$, whether Cores$(n, c)$ is (not) empty.

5.2 Preliminaries

As set of neighbor vectors NV of the FCC lattice, we choose the set of minimal vectors of the FCC, i.e. as we recall from Chapter 2,

$$NV = \{\begin{pmatrix}\pm 1\\ \pm 1\\ 0\end{pmatrix}, \begin{pmatrix}\pm 1\\ 0\\ \pm 1\end{pmatrix}, \begin{pmatrix}0\\ \pm 1\\ \pm 1\end{pmatrix}\}.$$

We define a notation for *lines* in $\mathbb{R}^3$. For vectors $\vec{a}, \vec{u} \in \mathbb{Z}^3$, let Line$(\vec{a}, \vec{u})$ denote the set

$$\{\vec{p} \in \mathbb{R}^3 \mid \exists \lambda \in \mathbb{R} : \vec{p} = \vec{a} + \lambda \vec{u}\}.$$

For $\vec{a} \in D_3$, we are interested in *lattice lines*

$$\text{Line}(\vec{a}, \vec{u}), \text{ where } \vec{u} \in NV,$$

and further *non-lattice lines*

$$\text{Line}(\vec{a}, \vec{u}), \text{ where } \vec{u} \in \mathbb{Z}^3 \text{ has length 2.}$$

Recall the definition of ξ-layers from Chapter 2. We redefine the function occlines for the FCC lattice using terms of this chapter.

Definition 5.2.1
For an x-layer f of the plane $x = d$,

$$\text{occlines}(f) = (a, b),$$

where

$$a = \left|\left\{\ l \;\middle|\; \exists \vec{a} \in D_3 : l = \text{Line}(\vec{a}, \begin{pmatrix}0\\ 1\\ 1\end{pmatrix}) \wedge l \cap f \neq \emptyset\ \right\}\right|$$

and analogously

$$b = \left|\left\{\ l \;\middle|\; \exists \vec{a} \in D_3 : l = \text{Line}(\vec{a}, \begin{pmatrix}0\\ -1\\ 1\end{pmatrix}) \wedge l \cap f \neq \emptyset\ \right\}\right|.$$

Define occlines(f) *analogously for y-layers and z-layers f.*

Note that this redefinition is in accordance with the previous definitions of Chapter 2 and Chapter 4.

5.3 Constraint Model

5.3.1 Description of the Algorithm

Here, we discuss an algorithm for the core construction problem using frame sequences, i.e. for computing the sets $\mathrm{Cores}(S, c)$ given a set of frame sequences S of size n and a number of contacts c.
Our algorithm follows the constrain-and-generate principle, which is commonly used in constraint programming (cf. Chapter 2). For applying this principle, we need to model the problem by variables and constraints between the variables.
Introducing variables for points and constraining them is almost sufficient for defining solutions of the problem. However, this does not suffice for an efficient constraint program. Since we want to exploit the knowledge from the frame sequences, we model layers and lattice lines of the layers to express the constraints by the parameters a and b from the frame sequences. Furthermore, it is crucial to employ the dependencies between layers of different dimensions. To express those dependencies, we have to model non-lattice lines of the layers. The number of contacts c yields further constraints, which are non-redundant to the former ones since not every core satisfying the layer sequences has necessarily at least c contacts.
When giving the constraint model for the problem, we take care to make it completely symmetric, i.e. no geometrical symmetries, except translations, are broken. On the one hand this permits constraining the layers by the frame sequence sets in each of the three dimensions. On the other hand, this is crucial for the use of symmetry breaking as described in [BW99]. This symmetry breaking mechanism, which in our case breaks all symmetries by rotations and reflections, requires that either the symmetries are not broken by the constraint model or that the mechanism is told exactly which symmetries are broken. Otherwise, the search will be incomplete due to interference with the symmetry breaking.

5.3.2 Variables

All variables that are introduced in the following are *finite domain variables (FD-variables)*, which means that their values are restricted to values of finite integer domains. We formulate our model thinking of these variables as describing one actual core $C \in \mathrm{Cores}(S, c)$. Another (more accurate) view is that each assignment of the variables describes one of the cores in $\mathrm{Cores}(S, c)$ if and only if it satisfies the model constraints, which will be

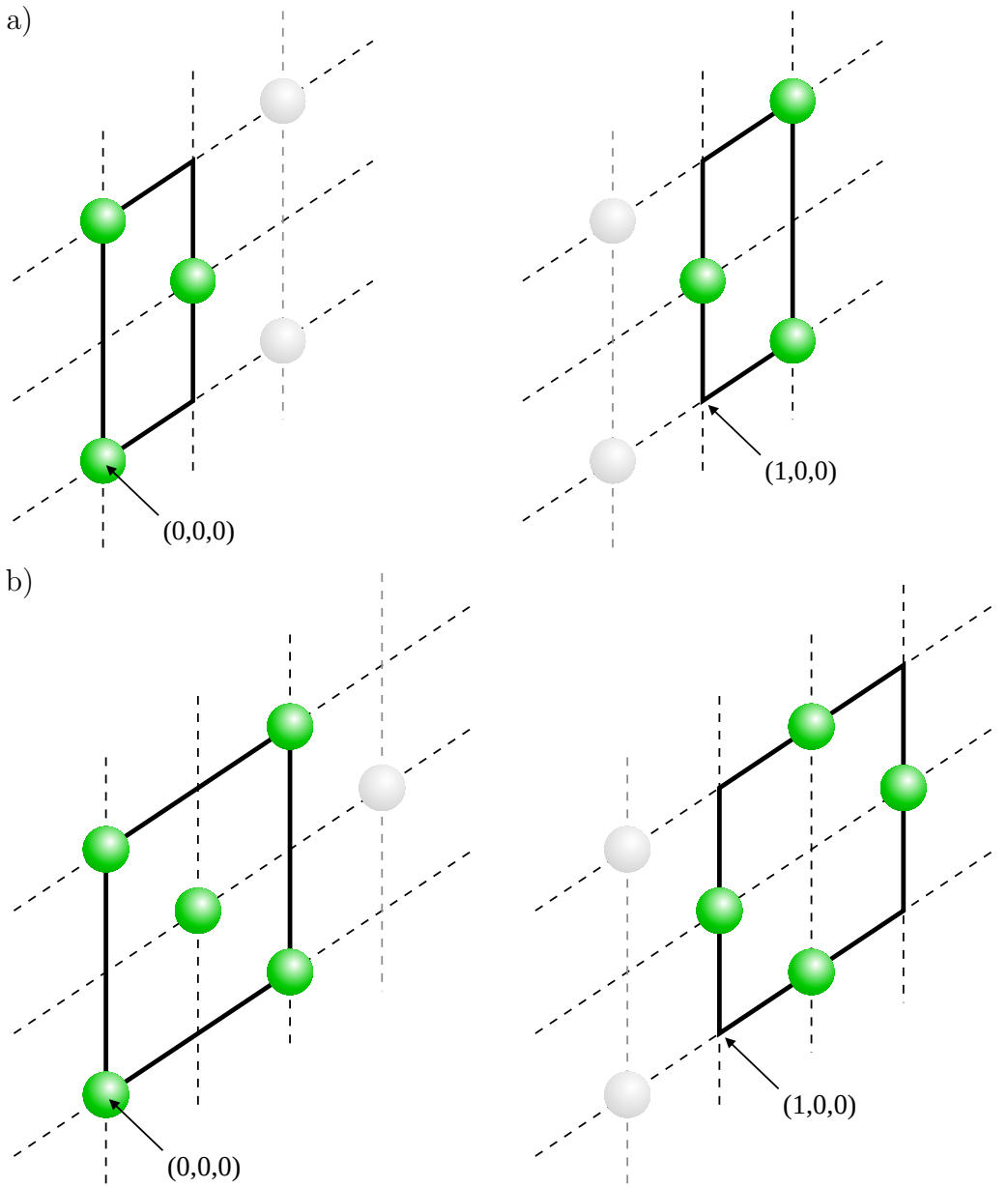


Figure 5.1: For a fixed surrounding cuboid, changing the parity of the minimal point in the cuboid excludes solutions or breaks symmetry. The points of the cuboid are marked by color. a) For a $3 \times 2 \times 1$ cuboid, fixing the parity breaks symmetry. b) For a $3 \times 3 \times 1$ cuboid (all dimensions are odd), fixing the parity even excludes possible cores.

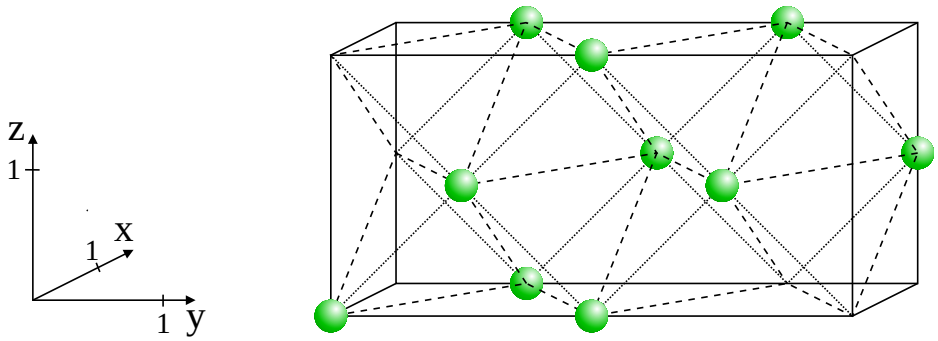


Figure 5.2: The cuboid for $\texttt{M.x} = 2$, $\texttt{M.y} = 5$, $\texttt{M.z} = 3$ and $\texttt{Min.x} + \texttt{Min.y} + \texttt{Min.z}$ even. The contacts within each x-layer are shown by dotted lines and the interlayer contacts between the two x-layers by dashed lines. The circles give an example core within the cuboid.

introduced later.

Denote the number of non-empty layers in the dimension $\xi \in \{x, y, z\}$ by $\texttt{M.}\xi$. All points of the core will be placed in a $\texttt{M.x} \times \texttt{M.y} \times \texttt{M.z}$ *surrounding cuboid*. When equating the dimensions of the surrounding cuboid and the numbers of non-empty layers, we employ that our cores are connected. From this fact, we can conclude that all non-empty layers are successive, which is easily shown by contradiction.

We can nearly fix the absolute coordinates of this cuboid for breaking translation symmetries. However, since D_3 contains only points of $\mathbb{Z}^3$ with even coordinate sum, the cuboid can only be fixed up to the minimal x, y, and z coordinate being one of $\{0, 1\}$. Otherwise, we would exclude solutions or at least break geometrical symmetries. (cf. Figure 5.1). We discussed the necessity of a symmetric model before in Sub-section 5.3.1.

We store these coordinates in FD-variables `Min.x`, `Min.y`, and `Min.z`, respectively. The surrounding cuboid consists of the points

$$\mathit{Cuboid} = \left(\begin{array}{cl} & \{\texttt{Min.x}, \ldots, \texttt{Min.x} + \texttt{M.x} - 1\} \\ \times & \{\texttt{Min.y}, \ldots, \texttt{Min.y} + \texttt{M.y} - 1\} \\ \times & \{\texttt{Min.z}, \ldots, \texttt{Min.z} + \texttt{M.z} - 1\} \end{array} \right) \cap D_3.$$

Please see Figure 5.2 for an illustration.

For every point $\vec{p} \in \mathit{Cuboid}$, we maintain a boolean FD-variable

$$\texttt{Point}[\vec{p}]\texttt{.occ} \in \{0, 1\}$$

that has value 1 if and only if the point $\vec{p}$ is element of the core C. Such a point $\vec{p}$, where $\texttt{Point}[\vec{p}].\texttt{occ} = 1$ is called a *core point* or *point of the core* C. Let $\xi \in \{x, y, z\}$. For every layer $\xi = d$, where

$$\texttt{Min}.\xi \leq d \leq \texttt{Min}.\xi + \texttt{M}.\xi - 1,$$

we introduce FD-variables

$$\texttt{Layer}[\xi, d].\texttt{n},\ \texttt{Layer}[\xi, d].\texttt{a},\ \text{and}\ \texttt{Layer}[\xi, d].\texttt{b}$$

for the layer parameters.
Note that we use a special notation for the variables with intentional similarity to notation found in programming languages. In this notation, we write indices of the variables in square brackets, like in $\texttt{Layer}[\xi, d]$, and we make use of named features, e.g. found in $\texttt{Min.x}$, where we select the feature $\texttt{x}$ of the tuple $\texttt{Min}$. Our notation gives a hint, how the variables may be organized in an implementation.
Furthermore, we use variables for all lattice and non-lattice lines within layers that intersect the cuboid. For $\vec{v}$ in

$$NV \cup \left\{ \begin{pmatrix} \pm 2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \pm 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \pm 2 \end{pmatrix} \right\},$$

there are FD-variables $\texttt{Line}[\vec{a}, \vec{v}].\texttt{n}$, for every set $\text{Line}(\vec{a}, \vec{v})$ that has a non-empty intersection with *Cuboid*. We identify two variables $\texttt{Line}[\vec{a}, \vec{v}].\texttt{n}$ and $\texttt{Line}[\vec{a}', \vec{v}].\texttt{n}$, if and only if

$$\text{Line}(\vec{a}, \vec{v}) = \text{Line}(\vec{a}', \vec{v}).$$

That is, there is only one variable for each line, which may be referenced in different ways for reasons of presentation.
$\texttt{Line}[\vec{a}, \vec{v}].\texttt{n}$ is the number of core points in $\text{Line}(\vec{a}, \vec{v}) \cap D_3$, i.e.

$$|\text{Line}(\vec{a}, \vec{v}) \cap C|.$$

Finally, we introduce one variable $\texttt{Contact}[\vec{p}, \vec{q}] \in \{0, 1\}$ for each pair of neighbored points $\vec{p}$ and $\vec{q} \in$ *Cuboid*, which signals, whether the points contribute a contact, i.e. whether $\vec{p}$ and $\vec{q}$ are core points.

5.3.3 Basic Constraints

Before formulating the constraints, we introduce a notation to express *reified constraints*, which reflect the entailment status of a constraint in a boolean

FD-variable (cf. the description of reified constraints in Chapter 3). Let $\mathfrak{c}$ be a constraint, fix a mapping δ, $\delta(\mathfrak{c}) \in \{0,1\}$, such that

$$\delta(\mathfrak{c}) = 1 \text{ if and only if } \mathfrak{c}.$$

The FD-variables are subject to the following constraints. First of all, we get

$$\sum_{\vec{p} \in Cuboid} \texttt{Point}[\vec{p}].\texttt{occ} = n$$

and

$$\sum_{\substack{\vec{p},\vec{q} \in Cuboid \\ \vec{p}-\vec{q} \in NV}} \texttt{Contact}[\vec{p},\vec{q}] \leq c.$$

In each dimension $\xi \in \{x,y,z\}$, a core C must have one of the frame sequences in S. Therefore, we want to introduce a constraint

$$\bigvee_{s_{\text{fr}} \in S} s_{\text{fr}} \text{ is frame sequence of } C \text{ in dimension } \xi.$$

On the finite domain variables of our model, this is expressed by the disjunction over all frame sequences $s_{\text{fr}} = (n_i, a_i, b_i)_{1 \leq i \leq |s_{\text{fr}}|}$ in S of

$$\text{for all } 1 \leq i \leq |s_{\text{fr}}| : \left(\begin{array}{l} \texttt{Layer}[\xi, \texttt{Min}.\xi + i - 1].\texttt{n} = n_i, \\ \texttt{Layer}[\xi, \texttt{Min}.\xi + i - 1].\texttt{a} = a_i, \\ \texttt{Layer}[\xi, \texttt{Min}.\xi + i - 1].\texttt{b} = b_i \end{array} \right).$$

Notably, this disjunction is made constructive, i.e. we propagate information out of the disjunction. In general, constructive disjunction is inefficient. However, in this special case we improve the propagation by introducing element constraints.[1] To prepare this, we index the frame sequences in the set S in an arbitrary order by numbers $1, \ldots, |S|$. We introduce finite domain variables $\texttt{FSeqIndex}[\xi]$, which encode the index of the frame sequence in dimension ξ. When for $\xi \in \{x,y,z\}$ the variables $\texttt{Min}.\xi$ and $\texttt{M}.\xi$ are fixed, we can compile lists for each variable $\texttt{Layer}[\xi, \texttt{Min}.\xi + i - 1].\texttt{n}$, $\texttt{Layer}[\xi, \texttt{Min}.\xi + i - 1].\texttt{a}$, and $\texttt{Layer}[\xi, \texttt{Min}.\xi + i - 1].\texttt{b}$, such that these variables have to be the $\texttt{FSeqIndex}[\xi]$-th elements of their corresponding list.

[1] In general, the element constraint constrains x to be the i-th element of a list, where x and i are FD-variables and the list is ground. In our implementation language Oz [Smo95], the element constraint is available and it is propagated efficiently.

It remains to constrain relations between the variables to get the basic constraint formulation for our problem. First, we relate lines to points by

$$\texttt{Line}[\vec{a},\vec{v}].\texttt{n} = \sum_{\vec{p}\in \mathrm{Line}(\vec{a},\vec{v})\cap \mathit{Cuboid}} \texttt{Point}[\vec{p}].\texttt{occ}$$

for all line variables. Then, we relate layer parameters to their layers by

$$\sum_{\substack{\vec{p} \text{ in layer } \xi = d \\ \text{of } \mathit{Cuboid}}} \texttt{Point}[\vec{p}].\texttt{occ} = \texttt{Layer}[\xi, d].\texttt{n}$$

for all $\xi \in \{x, y, z\}$ and $\texttt{Min}.\xi \leq d < \texttt{Min}.\xi + \texttt{M}.\xi$ and furthermore, for x-layers and lattice lines in direction $\begin{pmatrix}0\\1\\1\end{pmatrix}$introduce the constraints

$$\sum_{\substack{\mathrm{Line}(\begin{pmatrix}d\\r\\0\end{pmatrix},\begin{pmatrix}0\\1\\1\end{pmatrix})\cap \mathit{Cuboid}\neq\emptyset \\ r\in\mathbb{Z}}} \delta(\texttt{Line}[\begin{pmatrix}d\\r\\0\end{pmatrix},\begin{pmatrix}0\\1\\1\end{pmatrix}].\texttt{n} > 0) = \texttt{Layer}[x, d].\texttt{a}$$

and the analogous constraints for $\texttt{Layer}[x, d].\texttt{b}$, the y-layers, and z-layers. Now, we relate contacts to points and to the total number of contacts. For any contact variable $\texttt{Contact}[\vec{p},\vec{q}]$, we introduce

$$\texttt{Contact}[\vec{p},\vec{q}] = \delta(\texttt{Point}[\vec{p}].\texttt{occ} = 1 \wedge \texttt{Point}[\vec{q}].\texttt{occ} = 1).$$

Finally, we state

$$\sum \texttt{Contact}[\vec{p},\vec{q}] = c.$$

For completely constraining the problem, it remains to introduce a constraint for ensuring connectivity of the core. However, we are not aware of an efficient constraint propagator for this purpose. Therefore, we test whether a core is connected only after all FD-variables are completely determined. For better propagation, we introduce constraints that are implied by connectivity. Such a constraint is that each point $\vec{p}_0$ of the core has at least one neighbor $\vec{p}_1$ in the core, if there are at least two core points ($|C| \geq 2$). This is expressed by introducing for every $\vec{p}_0$ in the cuboid,

$$\sum_{\vec{p}_1 \text{ neighbor of } \vec{p}_0} \texttt{Point}[\vec{p}_1].\texttt{occ} \quad \geq \quad \texttt{Point}[\vec{p}_0].\texttt{occ}\,.$$

The previous constraints define the problem non-redundantly. For improving constraint propagation, we introduce redundant constraints like the following ones.

- The surrounding cuboid has to be large enough to include the core. Therefore, we introduce the constraints

$$\lceil \frac{\texttt{M.x} \cdot \texttt{M.y} \cdot \texttt{M.z}}{2} \rceil \geq n$$

if the sum $\texttt{Min.x} + \texttt{Min.y} + \texttt{Min.z}$ is even and

$$\lfloor \frac{\texttt{M.x} \cdot \texttt{M.y} \cdot \texttt{M.z}}{2} \rfloor \geq n$$

otherwise.

- The line variables are connected to the layer parameter n by constraints

$$\sum_{\substack{\text{Line}(\binom{d}{r}_{0}, \binom{0}{1}_{1}) \cap \mathit{Cuboid} \neq \emptyset \\ r \in \mathbb{Z}}} \texttt{Line}[\binom{d}{r}_{0}, \binom{0}{1}_{1}].\texttt{n} = \texttt{Layer}[x, d].\texttt{n}$$

and analogous ones.

Cavities in the core deserve a special discussion. Strong constraints can be introduced, whenever we know that there are no cavities in the core or in a certain layer. This knowledge may have two origins. Either, we choose to enumerate only cavity-free cores, or we allow cavities but the absence of cavities is derived.[2]
For example, for a line without cavities, we constrain that its core points are consecutive. Assume that the points in the intersection of the line and the cuboid are labeled $\vec{p}_1, \ldots \vec{p}_k$, such that $\vec{p}_i$ and $\vec{p}_{i+1}$ are neighbors $(1 \leq i < k)$. Then, if the line is not empty, this constraint is expressed by

$$\texttt{Point}[\vec{p}_1].\texttt{occ} + (\sum_{i=1}^{k-1} \delta(\texttt{Point}[\vec{p}_i].\texttt{occ} \neq \texttt{Point}[\vec{p}_{i+1}].\texttt{occ})) + \texttt{Point}[\vec{p}_k].\texttt{occ} = 2,$$

which states that there are exactly two transitions $0 \leftrightarrow 1$ in the sequence

$$(0, \texttt{Point}[\vec{p}_1].\texttt{occ}, \ldots, \texttt{Point}[\vec{p}_k].\texttt{occ}, 0).$$

5.3.4 Using Local Upper Bounds on the Number of Contacts

The number of contacts within each layer is bound by the parameters of the layer n, a, and b that are given by the frame sequence (cf. Chapter 4.) Note

[2] In particular, this can be derived from upper bounds on the contacts as it is discussed in the next subsection.

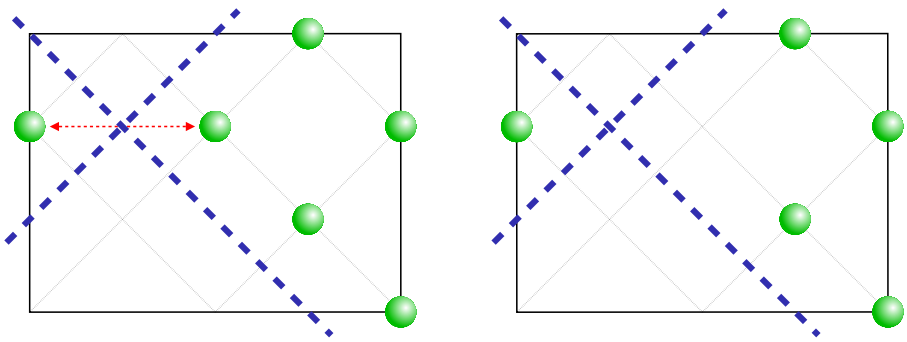


Figure 5.3: The thick dashed lines are drawn between non-overlapping pairs of lines. In both layers, we count one non-overlapping pair. In the right layer, we count one non-touching pair. (In the left layer, the lines are touching, which is indicated by the double arrow.)

that the exact number of contacts is determined by the parameters and the number of cavities in the layer. Thus, we can constrain the number of these (intra)layer contacts, which is represented by variables $\texttt{Layer}[\xi, d].\texttt{con}$.
Furthermore, we introduce redundant constraints that employ the upper bounds on the number of contacts between successive layers, called interlayer contacts. From Chapter 4 we know non-trivial upper bounds on the number of layer and interlayer contacts given parameters of the layers, namely the layer size, the previously defined occlines(f) and the number of *non-touching* rows and *non-overlapping* rows. Please, recall the definition of the latter terms from Chapter 4. For an additional illustration, we provide the Figure 5.3.
Furthermore, for

$$\texttt{Min}.\xi \leq d_1, d_2 < \texttt{Min}.\xi + \texttt{M}.\xi$$

and $d_2 = d_1 \pm 1$, we introduce FD-variables $\texttt{Interlayer}[\xi, d_1, d_2].\texttt{con}$ to hold the number of interlayer contacts between layers $\xi = d_1$ and $\xi = d_2$. This variable is constrained to the sum of the corresponding contact variables and the total number of contacts is constrained to the sum of the layer contacts and the interlayer contacts. The bound is strengthened and recomputed during the enumeration as more and more information, e.g. which lines intersect the core (see Figure 5.4), becomes known. Therefore, variables to hold the additional parameters, the number of non-overlapping rows and non-touching rows, are introduced for each layer and corresponding constraints

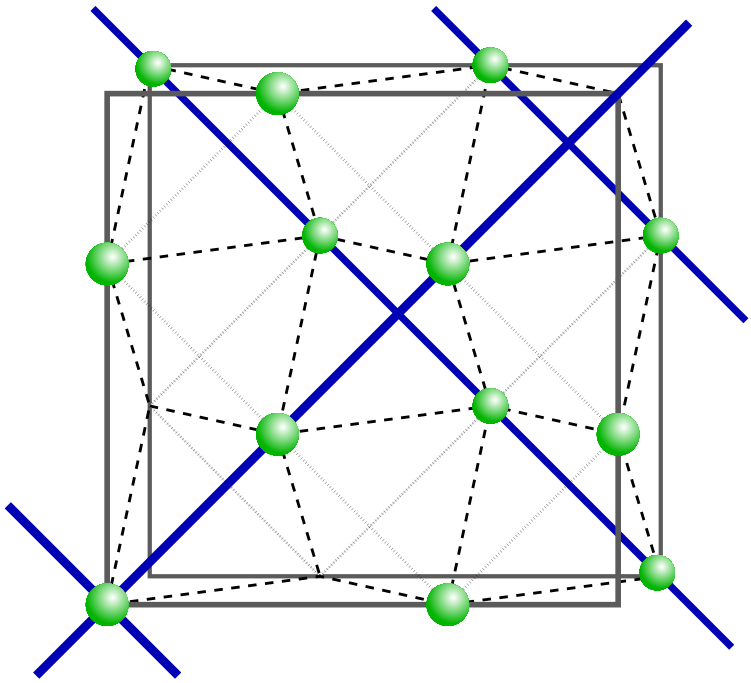


Figure 5.4: Represents an example situation in the search. The thick lines are already known to intersect the core, i.e. their corresponding line variables $\texttt{Line}[\vec{a},\vec{v}].\texttt{n}$ are greater or equal 1. Assume in each layer there are 5 core points, the beads mark remaining potential positions. The line constraints restrict the number of contacts, hence this additional knowledge is exploitable for the contacts bound.

are stated. Then, we introduce FD-variables $\texttt{Interlayer}[\xi, d_1, d_2].i$ to hold the number of core points in $\xi = d_2$ with at least $i = 1, 2, 3$, or 4 contacts to core points in $\xi = d_1$. Such points were called *i-points* (cf. Chapter 4). Finally we can bind $\texttt{Interlayer}[\xi, d_1, d_2].\texttt{con}$ to the sum

$$\sum_{1 \leq i \leq 4} i \cdot \texttt{Interlayer}[\xi, d_1, d_2].i.$$

5.3.5 Search strategy

We start the search by enumerating the variables `M.x`, `M.y`, `M.z`, `Min.x`, `Min.y`, and `Min.z`. This fixes the surrounding cuboid and allows in an implementa-

n	# contacts	# search-nodes	depth	time
40	152	167	17	17.2 s
60	243	182	72	4.6 s
82	349	220	37	14.2 s
102	447	54	20	8.2 s

Table 5.1: Search for all optimal cores of size n, given the layer sequences. We list the number of contacts, the number of nodes and depth of the search tree, and time of the constraint search for every core size n.

tion to construct all data structures. Then, we start by enumerating polar lines of the core. Polar lines are lattice lines through points of the surrounding cuboid that do not contain points of the core. Afterwards, we distribute over the point variables in order to fix the core completely. To break rotation and reflection symmetries, we employ symmetry breaking search [BW99]. This search is a special form of constrained search, which only finds solutions modulo given symmetries and employs this to prune the search tree.

5.4 Results

The presented constraint model was implemented in the constraint language Oz [Smo95]. Usually, the set of all optimally compact cores of a given size is found within a few seconds to minutes by our search program. Some results are shown in Table 5.1. The search program implements most of the presented ideas as well as additional redundant constraints.
We present some of the optimal cores for $n = 60$ and $n = 100$ elements in Figures 5.5 and 5.6. The cores are shown in plane sequence representation. This representation shows a coloring by the sequence of its occupied x-layers in the lattice D_3' (cf. preliminaries of Chapter 4, D_3' is mainly a transformed version of D_3). For each x-layer $x = d_1$ the lower left corner of the grid has coordinates $(d_1, 0, 0)$. The grid-lines have distance 1. The core points in each x-layer are shown as filled circles. There is a noteworthy difference between layers $x = d_1$, where d_1 is even and those where it is odd. In the latter ones the points have non-integer y and z coordinates. For illustrating the relation of plane sequence representation to a three-dimensional representation, we provide an additional view of one of the given example cores, in Figure 5.5b.

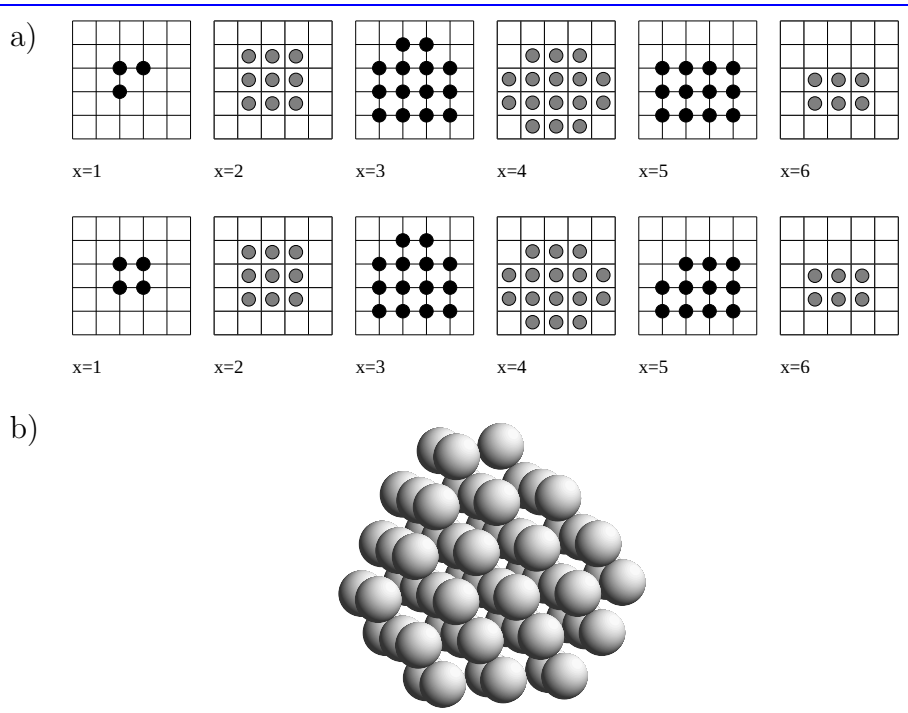


Figure 5.5: Cores of size 60. a) Two optimal cores in plane sequence representation. b) Three-dimensional view of the first core from (a).

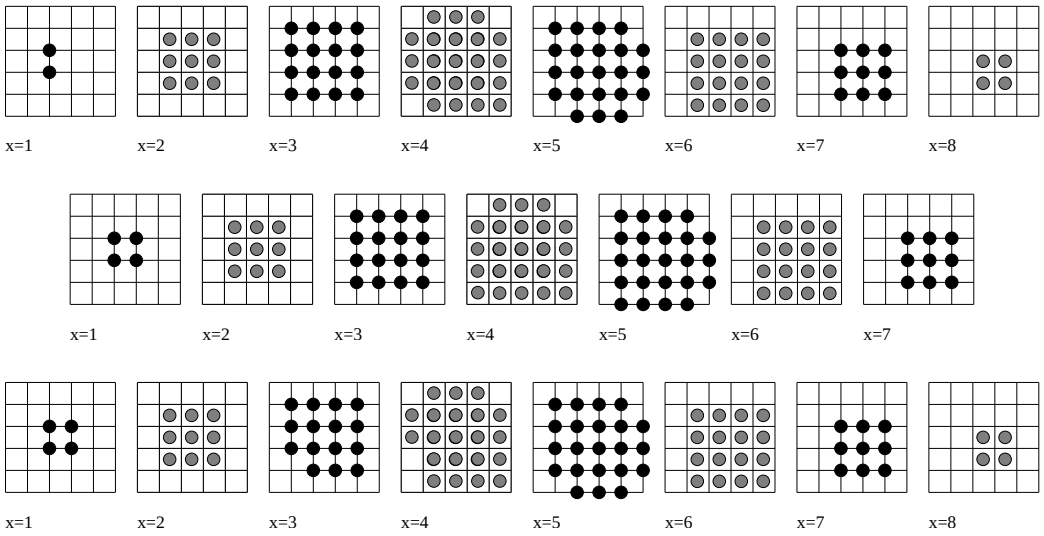


Figure 5.6: Plane sequence representations of three optimal cores of size 100.

Chapter 6

Threading to Cores

This chapter describes the last step of our structure prediction approach, where we finally compute the native structures of sequences in the HP-models of the cubic and face-centered cubic lattice.

In Section 2.2 from Chapter 2, we described the HP-model and discussed its properties. As a main property of the model, given a sequence *seq*, the HP-energy of a structure *str* is determined by the number of contacts in its hydrophobic core C. As we recall from Chapter 2, the HP-energy is given by

$$-\,\text{contacts}(C) \;+\; \text{HH-chain}(\mathit{seq}).$$

In consequence, a structure for a sequence *seq* is native, if and only if its hydrophobic core has maximally many contacts among all hydrophobic cores of structures of *seq*.

Due to the last two chapters, we can already compute cores with maximally many contacts as well as cores with a given number of contacts, for the cubic and the face-centered cubic lattice. This allows for a systematic enumeration strategy for predicting the minimal energy and the native structures of a sequence.

The remaining technical challenge of this chapter is mapping a HP-sequence to a core, such that the core forms a hydrophobic core of the resulting structure. We call this process *threading*. The threading problem is to find such a structure for a given sequence and a given core. We describe our threading approach for arbitrary lattices — actually, most of the theory of the chapter is even developed for arbitrary graphs. For the rest of the chapter, we fix a lattice L.

Structure Prediction by Threading

Before going into technical details of the threading algorithm, we recall shortly how threading is used for structure prediction. Given a sequence *seq*, structure prediction should answer two main questions.

- What is the minimal energy of structures of *seq*?
- What are the structures of *seq* that have minimal energy?

The first question is answered constructively by finding one minimal energy structure of *seq*. Therefore, initialize c by the maximal number of contacts in cores of size n_{H}, where n_{H} denotes the number of H-monomers in sequence *seq*. Then, we thread the sequence to the cores in $\mathrm{Cores}(n_{\mathrm{H}}, c)$ modulo symmetry. Since threading does not necessarily succeed, we have two cases. Either, we find a structure of *seq*, where its hydrophobic core is in $\mathrm{Cores}(n_{\mathrm{H}}, c)$. Then, the minimal energy is known by c and we can stop immediately. Or, there is no such structure. In this case, we decrease the number of contacts c by one and iterate the threading to cores in $\mathrm{Cores}(n_{\mathrm{H}}, c)$.
The iteration is guaranteed to terminate with a successful threading. Thus, after this procedure we know the minimal energy and the maximal number of HH-contacts of any structure of *seq*.
For the second question, we will first determine the maximal number of contacts c in any hydrophobic core of *seq*, i.e. we start by answering the first question. Then, the sequence is threaded to each core in $\mathrm{Cores}(n_{\mathrm{H}}, c)$.

Predicting Structures with Unconnected Cores

The above strategy finds all structures with connected cores, since the sets $\mathrm{Cores}(n_{\mathrm{H}}, c)$ contain only connected cores. The restriction to connected cores was introduced during core construction. The main technical reason was that as soon as there is one unconnected point set for a given size and number of contacts, there are infinitely many unconnected point sets of the same size and number of contacts. Of course, this is a consequence of our definition of a point set, where each of its points has a fixed position. In our threading procedure we can only use a single point set of this kind as the hydrophobic core.
However, we can overcome the restriction to connected cores in the threading step. For threading the sequence *seq* to all (connected and unconnected) cores of size n_{H} with at least c contacts, we start by threading the sequence to the connected cores in $\mathrm{Cores}(n_{\mathrm{H}}, c)$. We will then systematically enumerate structures with unconnected hydrophobic cores. Such cores consist of

connected components, which are already precomputed in sets Cores(n'_{H}, c') for $n'_{\mathrm{H}} < n_{\mathrm{H}}$ and $c' \leq c$.

First, we can calculate which (if any) compositions from connected components can yield c or more contacts. Among all unconnected point sets of size n, the most contacts are formed by sets which consist of one connected component of size $n-1$ and one isolated point. Then, the next best cores consist of a component of size $(n-2)$ and a component of size 2 and so on. For enumerating optimal and almost optimal structures on arbitrary cores, it suffices to handle cores that consist of one large component and one or few small components.

We sketch the generation of all requested structures of *seq* with a hydrophobic core with only one isolated point. The presented idea can then be generalized for handling other strongly asymmetric decompositions of the core into its connected components.

For enumerating all structures with only one isolated point H-monomer, we will successively replace one H-monomer by a P-monomer in the sequence *seq* and thread the mutated sequence to the connected cores in Cores($n_{\mathrm{H}}-1, c$). As effect of the substitution, only the remaining H-monomers have to form cores in Cores($n_{\mathrm{H}}-1, c$) and the substituted monomer is unconstrained. Only H-monomers that are isolated from other H-monomers in the chain are replaced, since only such a monomer can be positioned in isolation from the large connected component of the hydrophobic core. Finally, we ensure isolation of the replaced monomer in the resulting structures. Technically, we could filter the results for this aim. More efficiently, we can easily add a constraint to the later developed constraint model.

Usually structures with unconnected hydrophobic cores are not an important issue for structure prediction, since the most compact cores are always connected and the optimal structures are usually found on compact cores. Having in mind the discussed strategy for the treatment of structures with unconnected hydrophobic cores and in the same time their low practical significance, we will not discuss this issue further. For our examples in Chapter 7, we will only predict structures with connected hydrophobic cores.

Chapter Overview

In the rest of the chapter, we develop a constraint-based algorithm for threading a sequence to a core. Section 6.1 recalls the threading problem and gives a constraint model for its solution. The constraint model of this section abstracts from a low-level formulation of the constraints and their actual

propagation. Most notably, we introduce a new global constraint, namely the walk constraint. The subsequent section describes this new constraint and develops a propagator, which makes the constraint hyper-arc consistent. In our constraint model, the walk constraint is used in conjunction with the all-different constraint. Section 6.3 discusses the combination of the two global constraints. A combination of both constraints allows for much more powerful propagation. However, there is no efficient propagator for the combined constraint, which computes hyper-arc consistency. For still improving the propagation, we identify and investigate a class of relaxations of the combined constraint, which can be propagated efficiently. Finally, we discuss the implementation of an efficient propagator and give some results.
Parts of this chapter base upon [BW01a].

6.1 A Constraint Model

The input of the *threading problem* consists of a HP-sequence *seq* and a core C, where its size $|C|$ equals the number of H-monomers in the sequence. It asks for the structures of *seq* that have the hydrophobic core C.
We tackle the threading problem by a constraint based approach. The structure of a protein is modeled by a set of finite domain variables $\mathtt{X}_1, \ldots, \mathtt{X}_{|seq|}$, whose domains consist of points of the lattice L.
The problem is now to find a solution, i.e. an assignment of the monomers to points, subject to the constraints

1. the points $\mathtt{X}_i$, where $seq_i = \mathrm{H}$ and $1 \leq i \leq |seq|$, are elements of C,
2. all the $\mathtt{X}_i$ are different, where $1 \leq i \leq |seq|$, and
3. the points $\mathtt{X}_1, \ldots, \mathtt{X}_{|seq|}$ form a walk (*walk constraint*).

Obviously, the variables which represent H-monomers have finite domains, due to the first constraint. For the P-monomers, note that, the first two constraints imply that P monomers are not in the core, since there are as many H-monomers as core positions. However, due to the finite chain length and the third constraint, the domains for the variables that represent P-monomers are also finite. The second constraint tells that a protein structure is self-avoiding. Finally, the third constraint tells that chain bonds between monomers are preserved in a protein structure, i.e. the monomer positions form a walk through the lattice, where the points $\mathtt{X}_i$ and $\mathtt{X}_{i+1}$ are neighbored $(1 \leq i < |seq|)$. The walk constraint corresponds to the chain constraint, which was formulated for HP-model structures in Chapter 2.

Some attention has to be paid, since many constraint systems do only support integer finite domain variables, whereas in our formulation domains consist of lattice points. As a remedy, we assign unique integers to the points in the domains. There are several straightforward ways for constructing such a mapping from points to integers. For example, due to the walk constraint, we can easily arrange that all points in the domains of variables have coordinates between 0 and $m-1$, where m is large enough (depending on the sequence length n). A suitable mapping is then

$$\vec{p} \leftrightarrow m^2 p_x + m p_y + p_z.$$

6.2 Walk Constraints

For entailing the first constraint, it suffices to get node-consistency of the constraints

- $X_i \in C$ for H-monomers i and
- $X_i \notin C$ for P-monomers i,

which is technically done by assigning (finite) domains to the variables.
Both of the remaining constraints can be propagated globally. The global treatment of the all-different constraint is well described in [Reg94]. Thus, we focus on the walk constraint. We will later discuss how one gets further propagation by combining the two constraints.
For generality, we discuss the constraints on arbitrary finite graphs. Afterwards, we can specialize the results for the cubic lattice and the FCC lattice straightforwardly. For the lattices, the set of graph nodes is a subset of the lattice nodes and the edges are all pairs of graph nodes in minimal lattice distance.
In the following, we fix a graph $G = (V, E)$. A *walk of length* n is a tuple $p = (p_1, \ldots, p_n)$ of length n of elements of V, such that

$$\forall 1 \leq i \leq n-1 : (p_i, p_{i+1}) \in E.$$

Usually, we write a walk $(p_1, \ldots, p_n)$ more conveniently as $p_1 \ldots p_n$.
Denote the set of walks of length n by walks(n). Note that intentionally walks are allowed to contain cycles, i.e. are not necessarily self-avoiding.
We define a walk constraint that states that the nodes assigned to the argument variables form a walk. Specializing a notion of Chapter 2, we call a walk $p \in \text{walks}(n)$ *consistent with variables* $\mathtt{X}_1, \ldots, \mathtt{X}_n$, if and only if

$$\forall 1 \leq i \leq n : p_i \in \text{dom}(\mathtt{X}_i).$$

Definition 6.2.1 (Walk Constraint)
Let $\mathtt{X}_1, \ldots, \mathtt{X}_n$ *be variables. Then, the* walk constraint

$$C = \text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$$

is interpreted by the set of tuples

$$\text{T}(C) = \{p \in \text{walks}(n) | p \textit{ is consistent with } \mathtt{X}_1, \ldots, \mathtt{X}_n\}.$$

Already in Chapter 2, we discussed that a constraint C is defined semantically by its set of tuples $\text{T}(C)$. This set $\text{T}(C)$ consists of all combinations of values for variables of C that satisfy C.
Hyper-arc consistency (cf. Chapter 2) of the walk constraint is a local property. Due to the following theorem, the hyper-arc consistency of the n-ary walk constraint is reduced to the arc consistency of the set of all 2-ary walk constraints.

Theorem 6.2.2
Let $\mathtt{X}_1, \ldots, \mathtt{X}_n$ *be variables, where* $n \geq 2$. *Then,* $\text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ *is hyper-arc consistent, if and only if for* $1 \leq i \leq n-1$ *all constraints* $\text{Walk}(\mathtt{X}_i, \mathtt{X}_{i+1})$ *are arc consistent.*

The theorem is actually a corollary to a more general result of Freuder [Fre82]. By this result, hyper-arc consistency amounts to global consistency in a tree-structured network of binary constraints. Nevertheless, we present a direct proof of Theorem 6.2.2 for better understanding.
Proof of Theorem 6.2.2
The global to local direction is trivial. Let $\text{Walk}(\mathtt{X}_i, \mathtt{X}_{i+1})$ be arc consistent with all $1 \leq i \leq n-1$. We have to show that $C = \text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ is hyper-arc consistent.
We proof the claim by induction on the constraint arity n. The **base case** $n = 2$, is obviously valid.
For the **induction case** $n > 2$, let $1 \leq i \leq n$ and $a \in \text{dom}(\mathtt{X}_i)$. We show that there is a walk p that is consistent with $\mathtt{X}_1, \ldots, \mathtt{X}_n$, such that $p_i = a$.
For the sub-case, where $1 < i < n$, the constraints $\text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_i)$ and $\text{Walk}(\mathtt{X}_i, \ldots, \mathtt{X}_n)$ are arc consistent by induction hypothesis. In consequence, for $p_i = a$ there exists a sub-walk $p_1 \ldots p_i$ that is consistent with $\mathtt{X}_1, \ldots, \mathtt{X}_i$ and a sub-walk $p_i \ldots p_n$ that is consistent with $\mathtt{X}_i, \ldots, \mathtt{X}_n$. Thus, there is the required walk by composition of the two sub-walks. For proving the induction step, we need to prove the remaining sub-cases $i = 1$ and $i = n$.
For the sub-case $i = 1$, there is a walk $p_1 p_2$ that is consistent with $\mathtt{X}_1, \mathtt{X}_2$ with $p_1 = a$ due to the arc consistency of $\text{Walk}(\mathtt{X}_1, \mathtt{X}_2)$. Furthermore, by induction

hypothesis, i.e. by the arc consistency of Walk($\mathtt{X}_2, \ldots, \mathtt{X}_n$), there is a walk $p'_2 \ldots p'_n$ that is consistent with $\mathtt{X}_2, \ldots, \mathtt{X}_n$, where $p'_2 = p_2$. The remaining sub-case $i = n$ is shown analogously. □

6.3 Combining Walk and All-Different

The combination of the walk constraint with the all-different constraint yields a new constraint, which allows only self-avoiding walks.

Definition 6.3.1 (All-different constraint)
For variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$*, we define the* all-different constraint

$$C = \text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$$

by

$$\text{T}(C) = \left\{ \begin{array}{c} (\tau_1, \ldots, \tau_n) \\ \in \text{dom}(\mathtt{X}_1) \times \cdots \times \text{dom}(\mathtt{X}_n) \end{array} \middle| \begin{array}{c} \forall 1 \leq i < j \leq n : \\ \tau_i \neq \tau_j \end{array} \right\}.$$

Definition 6.3.2 (Self-avoiding walk constraint)
We define the self-avoiding walk constraint SAWalk($\mathtt{X}_1, \ldots, \mathtt{X}_n$) *by*

$$\text{T}(\text{SAWalk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)) = \text{T}(\text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n)) \cap \text{T}(\text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)).$$

We are not aware of any efficient arc consistency algorithm for this combined constraint in the literature. Furthermore, it is unlikely that there exists one. It is well known that many problems involving self-avoiding walks, especially counting of such walks, are intrinsically hard and there are no efficient algorithms to solve them [MS96].
However, the treatment of self-avoiding walks is desirable since it promises much better propagation in practice. Therefore, we propose a relaxation of the intractable self-avoiding walk constraints.
One relaxation of the self-avoiding constraint is constraining the walks to be non-reversing. Non-reversing walks are walks which do not turn back immediately. Hence, a non-reversing walk is (only) locally avoiding itself, but more distant parts of the walk can overlap.
In principle, we follow the idea of locally self-avoiding walks. However, we develop the idea in more generality than found in non-reversing walks. For this purpose, we define the following sets of walks.

Definition 6.3.3 (Locally self-avoiding walks)
Let $1 \leq k \leq n$*. A* k-avoiding walk $p = p_1 \ldots p_n$ of length n *is a walk* $p \in \text{walks}(n)$*, where for all* $1 \leq i \leq n - k + 1$*, the nodes* $p_i, \ldots, p_{i+k-1}$ *are all*

different to each other. We define that for $k > n$, k-avoiding is equivalent to n-avoiding. Denote the set of k-avoiding walks of length n by walks$[k](n)$. *We call k-avoiding walks also* locally self-avoiding walks.

Note that general walks (resp. self-avoiding walks) of length n are special cases of k-avoiding walks namely 1-avoiding walks (resp. n-avoiding walks) of length n. For graphs with symmetric and non-reflexive edges, the property non-reversing is equivalent to 3-avoiding. Furthermore, note that by definition,

$$\text{walks}[k](n) \supseteq \text{walks}[k'](n)$$

holds for all $1 \leq k \leq k' \leq n$.

Definition 6.3.4 (Locally self-avoiding walk constraints)
Let $\mathtt{X}_1, \ldots, \mathtt{X}_n$ denote variables. Then, we define the set of k-avoiding walks that are consistent with $\mathtt{X}_1, \ldots, \mathtt{X}_n$ *as*

$$\text{cwalks}[k](\mathtt{X}_1, \ldots, \mathtt{X}_n).$$

*The corresponding constraints constrain their variables to form k-avoiding walks. Therefore, we define the k-*avoiding walk constraint $\text{Walk}[k](\mathtt{X}_1, \ldots, \mathtt{X}_n)$ *by*

$$\text{T}(\text{Walk}[k](\mathtt{X}_1, \ldots, \mathtt{X}_n)) = \text{cwalks}[k](\mathtt{X}_1, \ldots, \mathtt{X}_n).$$

Such a constraint is also called locally self-avoiding walk constraint.

The interplay of $\text{Walk}[k](\mathtt{X}_1, \ldots, \mathtt{X}_n)$ and $\text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ provides much better propagation than the one of the two constraints $\text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ and $\text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$. This is demonstrated by the following example. The HP-model of the cubic lattice has a special property, which restricts the placing of certain monomers. Therefore, we look at a HPH sub-sequence and the corresponding sub-structure. Then, the P-monomer cannot be positioned outside of the minimal cuboid that surrounds all H-monomers (see Figure 6.1). This property is detect via propagation of the 3-avoiding constraint without actually placing the three monomers. It cannot be detected, if we use only separate propagation of the constraints $\text{AllDiff}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$ and $\text{Walk}(\mathtt{X}_1, \ldots, \mathtt{X}_n)$, since an arbitrary walk can place the two H-monomers on the same position and the P-monomer arbitrarily in their neighborhood.

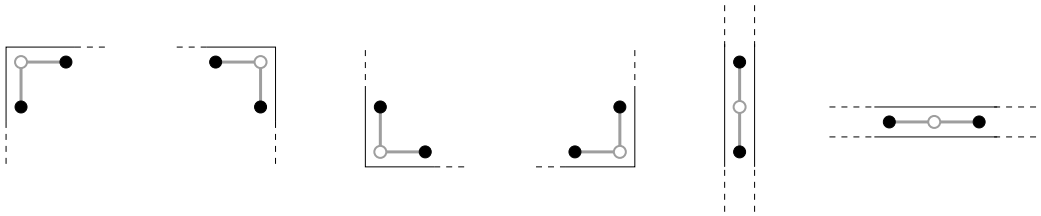


Figure 6.1: All possible sub-structures formed by a HPH sub-sequence in the cubic lattice. The P must be placed in the minimal cuboid that surrounds the two H-monomers and thus in the minimal cuboid surrounding all H-monomers.

6.4 Propagating Locally Self-Avoiding Walk Constraints

In the last section, we developed the notion of locally self-avoiding walk constraints. Here, we discuss a strategy for propagating such constraints efficiently. We develop a efficient approach to the propagation of locally self-avoiding walk constraints, which makes use of dynamic programming.

6.4.1 Computing Hyper-Arc-Consistency

The key to our algorithm is the counting of walks. For the hyper-arc consistency of a k-avoiding walk constraint, we need to know for each variable, whether a value v in its domain supported by a k-avoiding walk or not. For checking this support, we will count the walks, where the i-th monomer is placed on node v, for all monomer indices i and nodes v. Then, in order to get consistency, we remove all unsupported domain values.
Counting the total number of consistent k-avoiding walks is a good starting point. In the following, we denote the cardinality of a set S by $\#S$. For computing the number of consistent walks $\#\,\mathrm{cwalks}[k](\mathtt{X}_1, \ldots, \mathtt{X}_n)$, we will first define walks with given suffixes. For our purpose, the suffixes for computing $(k+1)$-avoiding walks always have a length of k. Exactly this length is required and sufficient for checking $(k+1)$-avoiding.

Definition 6.4.1 (Consistent suffix walks)
We define the set of $(k+1)$-avoiding walks that are consistent with $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$ with suffix $q = q_1 \ldots q_k$ *for* $k+1 \leq n$ *as*

$$\mathrm{csuffwalks}[k+1](x)[q] = \left\{\ p \in \mathrm{cwalks}[k+1](x) \;\middle|\; \forall 1 \leq i \leq k : p_{n-k+i} = q_i \ \right\}$$

In the following lemma, we make use of a recursively defined helper function $\#\mathrm{csw}[k+1](x)[q]$. As we will show later, $\#\mathrm{csw}[k+1](x)[q]$ is equal to $\#\,\mathrm{csuffwalks}[k+1](x)[q]$ for all values, where q is a self-avoiding walk that is consistent with the variables $\mathtt{X}_{n-k+1}, \ldots, \mathtt{X}_n$. Otherwise, $\#\mathrm{csw}[k+1](x)[q]$ remains undefined. By this definition, we resemble an efficient implementation more closely. Due to the use of the helper function, we can avoid to check whether q is a consistent self-avoiding walk in each recursion step.

Lemma 6.4.2
Let $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$ be variables, $0 < k \leq n$.
First, the number $\#\,\mathrm{cwalks}[k+1](x)$ is equal to the sum

$$\sum_{q \in \mathrm{walks}(k)} \#\,\mathrm{csuffwalks}[k+1](x)[q].$$

Second, for $q = q_1 \ldots q_k \in \mathrm{walks}(k)$, the number of consistent $(k+1)$-avoiding walks with suffix q can be computed recursively due to the following recursion equation.

$$\#\,\mathrm{csuffwalks}[k+1](x)[q] = \begin{cases} \#\mathrm{csw}[k+1](x)[q] & q \in \mathrm{cwalks}[k](\mathtt{X}_{n-k+1}, \ldots, \mathtt{X}_n) \\ 0 & \textit{otherwise,} \end{cases}$$

where we define for $q \in \mathrm{cwalks}[k](\mathtt{X}_{n-k+1}, \ldots, \mathtt{X}_n)$,

$$\#\mathrm{csw}[k+1](x)[q] = \begin{cases} 1 & n = k \\ \displaystyle\sum_{\substack{(q_0,q_1) \in E, \\ q_0 \notin \{q_1,\ldots,q_k\}, \\ q_0 \in \mathrm{dom}(\mathtt{X}_{n-k})}} \#\mathrm{csw}[k+1](\mathtt{X}_1, \ldots, \mathtt{X}_{n-1})[q_0 \ldots q_{k-1}] & n > k. \end{cases}$$

Proof of Lemma 6.4.2
Let $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$ be variables, $0 < k \leq n$. The first claim is satisfied, since clearly

$$\mathrm{cwalks}[k+1](x) \;=\; \biguplus_{q \in \mathrm{walks}(k)} \mathrm{csuffwalks}[q](k+1)[x],$$

where $\biguplus$ denotes disjoint union.
For the second claim, let

$$q = q_1 \ldots q_k \in \mathrm{walks}(k).$$

We will show that under the condition

$$q \in \text{cwalks}[k](\mathtt{X}_{n-k+1}, \ldots \mathtt{X}_n),$$

$$\#\,\text{csuffwalks}[k+1](x)[q] = \#\text{csw}[k+1](x)[q]. \tag{6.1}$$

If otherwise q is not self-avoiding or q is not consistent with $\bar{x} = \mathtt{X}_{n-k+1}, \ldots \mathtt{X}_n$, then the set csuffwalks$[k+1](x)[q]$ is empty by definition.
Now, let $q \in \text{cwalks}[k](\bar{x})$. Then, we show (6.1) by induction on $k \leq n$. For the **base case** $n = k$, note that $\bar{x} = x$. Clearly, q is the single element of csuffwalks$[k+1](x)[q]$.
The **induction case** is $n > k$. For any q_0 such that

$$(q_0, q_1) \in E, q_0 \notin \{q_1, \ldots, q_k\}, q_0 \in \text{dom}(\mathtt{X}_{n-k})$$

there is satisfied

$$\#\text{csw}[k+1](x')[q'_{q_0}] = \#\,\text{csuffwalks}[k+1](x')[q'_{q_0}],$$

where $q'_{q_0} = q_0 \ldots q_{k-1}$ and $x' = \mathtt{X}_1, \ldots, \mathtt{X}_{n-1}$. Note that q'_{q_0} depends on q_0, which is introduced for technical convenience. Consequently, it suffices to show that

$$\#\,\text{csuffwalks}[k+1](x)[q] = \left| \biguplus_{c_{q_0}} \text{csuffwalks}[k+1](x')[q'_{q_0}] \right|,$$

where we define c_{q_0} as the condition $(q_0, q_1) \in E \wedge q_0 \notin \{q_1, \ldots, q_k\} \wedge q_0 \in \text{dom}(\mathtt{X}_{n-k})$ in dependency of the node q_0.
First note that the sets csuffwalks$[k+1](x')[q'_{q_0}]$, where c_{q_0} holds, are disjoint, for different q_0. For the equation we have to show both inclusions.

- $\leq$. Let $p \in \text{csuffwalks}[k+1](x)[q]$. $q_0 = p_{n-k}$ satisfies c_{q_0}, and for $p' = p_1 \ldots p_{n-1}$ holds $p' \in \text{csuffwalks}[k+1](x')[q'_{q_0}]$.

- $\geq$. Let q_0 such c_{q_0} holds. Let

  $$p \in \text{csuffwalks}[k+1](x')[q'_{q_0}].$$

  Then, we get that

  $$p' \in \text{csuffwalks}[k+1](x)[q]$$

  holds for $p' = p_1 \ldots p_{n-1} q_k$.

□

Since the recursion equation furnishes a dynamic programming algorithm, the numbers of walks with suffixes can be computed efficiently.
This algorithm to compute the numbers of k-avoiding walks of maximal length n, where $2 \leq k \leq n$, has a polynomial complexity in n and the maximal number of nodes in the domains of the variables.
Note that the lemma handles only the case of k-avoiding walks, where $k \geq 2$. The reason is that for the walk property itself we have to remember a history of minimal length 1. Hence, the number of 1-avoiding walks can not be computed more efficiently than the number of 2-avoiding walks. Obviously the lemma could be slightly modified (by dropping the condition $q_0 \notin \{q_1, \ldots, q_k\}$ in the sum of the recursion step) to compute the number of 1-avoiding, i.e. general walks.
Analogously to walks with suffixes, we can treat walks with prefixes. Hence, define the *set of k-avoiding walks with prefix $q = q_1 \ldots q_m$ that are consistent with $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$* as

$$\text{cprefwalks}[k][q](x) = \left\{\ p \in \text{cwalks}[k](x) \;\middle|\; \forall 1 \leq i \leq \min(m,n) : p_i = q_i\ \right\}.$$

Due to symmetry of the definitions, the walks with prefixes can be treated analogously to walks with suffixes.
We can now express the number of k-avoiding walks that are consistent with $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$, where the i-th monomer occupies the position v, in terms of suffix and prefix walk numbers.
For preparation, define the set of these walks as $\text{cwalks}[k](x|i \mapsto v)$. In the case of general walks, the number of walks that map $\mathtt{X}_i$ to position v is the number of prefixes of length i that end in v times the number of suffixes of length $n - i$ starting in v. For k-avoiding walks, this does not suffice, since the composition of a k-avoiding prefix and suffix will not generate a k-avoiding walk in general. To guarantee this, the prefix and suffix has to overlap at least by $k-1$ positions. Note that the i can be located arbitrarily in this overlapping region. These considerations are summarized by the next lemma.

Lemma 6.4.3
Let $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$ be variables, $1 \leq i \leq n$, and $v \in V$. We choose values k and j such that $1 \leq k+1 \leq n$, $1 \leq j \leq i \leq j+k-1 \leq n$.

$$\#\,\text{cwalks}[k+1](x|i \mapsto v) = \sum_{\substack{q \in \text{walks}[k](k) \\ q_{i-j+1} = v}} \left(\begin{array}{c} \#\,\text{csuffwalks}[k+1](\mathtt{X}_1, \ldots, \mathtt{X}_{j+k-1})[q] \\ \cdot\ \#\,\text{cprefwalks}[k+1][q](\mathtt{X}_j, \ldots, \mathtt{X}_n) \end{array} \right).$$

Proof
Let k, x, i, j, and v be as in the lemma. It suffices to show that the sets $A = \text{cwalks}[k+1](x|i \mapsto v)$ and

$$B = \biguplus_{\substack{q \in \text{walks}[k](k) \\ q_{i-j+1}=v}} \left(\begin{array}{c} \text{csuffwalks}[k+1](\mathtt{X}_1, \ldots, \mathtt{X}_{j+k-1})[q] \\ \times \ \text{cprefwalks}[k+1][q](\mathtt{X}_j, \ldots, \mathtt{X}_n) \end{array} \right)$$

have equal cardinality. Note that the union is disjoint, since the suffixes and prefixes partition the sets of all consistent $k+1$-avoiding walks. Define $f : A \to B$ by $f(p) = (p_1 \ldots p_{j+k-1}, p_j \ldots p_n)$.

$\leq$. Let $p \in \text{cwalks}[k+1](x|i \mapsto v)$. $f(p) \in B$, since for $q = p_j \ldots p_j + k - 1$, q is in $\text{cwalks}[k](k)$, where $q_{i-j+1} = p_i = k$, and

$$p \in \text{csuffwalks}[k+1](\mathtt{X}_1, \ldots, \mathtt{X}_{j+k-1})[q] \times \text{cprefwalks}[k+1][q](\mathtt{X}_j, \ldots, \mathtt{X}_n)$$

holds. f is obviously injective.

$\geq$. We have to show f is surjective. Let $(p', p'') \in B$. This implies by definition

$$q = p'_{|p'|-k+1} \ldots p'_{|p'|} = p''_1 \ldots p''_k \in \text{walks}[k](k).$$

Let

$$p = p'_1 \ldots p'_{|p'|} p''_k \ldots p''_{|p''|}.$$

Then, by construction $f(p) = (p', p'')$. Due to the length k overlapping of p' and p'', p is a walk and p is $k+1$-avoiding. Further p is obviously consistent with x. Finally

$$p_i = q_{j-i+1} = k,$$

thus $p \in A$.

□

Based on the computation of these numbers we develop an hyper-arc consistency algorithm for the k-avoiding walk constraints.

Theorem 6.4.4
Let $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$ be variables with non-empty domains. The constraint $C = \text{Walk}[k](x)$ is hyper-arc consistent, if and only if for every $1 \leq i \leq n$ and $v \in V$, where $\#\,\text{cwalks}[k](x|i \mapsto v) = 0$, it holds that $v \notin \text{dom}(\mathtt{X}_i)$.

Proof
Let x and C be defined as in the theorem.
First, let C be hyper-arc consistent. Let $1 \leq i \leq n$ and $v \in V$, such that the set $\text{cwalks}[k](x|i \mapsto v)$ is empty. Then, there is no walk $p \in \text{cwalks}[k](x)$, where $p_i = v$. Hence there is no such walk in $\text{T}(C)$. We get $v \notin \text{dom}(\mathtt{X}_i)$, due to the hyper-arc consistency of C.
Second, let C be not hyper-arc consistent. Then, we show that there is a $1 \leq i \leq n$ and $v \in V$ such that $v \in \text{dom}(\mathtt{X}_i)$ and

$$\#\,\text{cwalks}[k](x|i \mapsto v) = 0.$$

The hyper-arc consistency of C has to be violated by at least one pair $1 \leq i \leq n$ and $v \in V$, where $v \in \text{dom}(\mathtt{X}_i)$. Choose such i and v. Since there is no walk p in $\text{T}(C)$, where $p_i = k$, there is no such walk in $\text{cwalks}[k](x)$. This implies

$$\text{cwalks}[k](x|i \mapsto v) = \emptyset.$$

□

6.4.2 A Global Propagator

Based on the considerations of the previous sections, we sketch an implementation of the k-avoiding walk constraint propagator.
Let $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$ be finite domain variables. The general strategy of the propagator for $\text{Walk}[k](x)$ is as follows

1. For all
$$q \in \text{walks}[k](k) \text{ and indices } i, \text{ where } k \leq i \leq n,$$
compute
$$\#\,\text{csuffwalks}[k](\mathtt{X}_1, \ldots, \mathtt{X}_i)[q] \text{ and } \#\,\text{cprefwalks}[k][q](\mathtt{X}_{n-i+1}, \ldots, \mathtt{X}_n).$$

2. Compute from this the numbers
$$\#\,\text{cwalks}[k](x|i \mapsto v)$$
for all $1 \leq i \leq n$ and $v \in V$. Whenever such a value equals 0, remove v from the domain of $\mathtt{X}_i$.

3. If at least one domain of the $\mathtt{X}_1, \ldots, \mathtt{X}_n$ changes repeat from step 1.

Even since we have presented efficient algorithms to compute the above numbers and thus get hyper-arc consistency of the walk constraint, there remain some details. Most demanding are incremental computation and the saving of copying time.
At the first invocation, the computation of the walk numbers can be done by dynamic programming algorithms. If domains are narrowed, the previously computed walk numbers can be updated. For this aim, one can devise an efficient update algorithm, which works destructively on the data structures. However, then the incremental computation comes at the price of copying the data structures, whenever the tree branches.

6.4.3 Further Propagation

The global handling of the locally self-avoiding walk constraint, enables further propagation, when the constraint is used in the context of an all-different constraint.
Assume that the variables in a set X are constrained as all different. Then, whenever we derive that in every solution one of the variables in $Y \subseteq X$ is assigned to a node v, we can introduce the basic constraints $v \notin \mathrm{dom}(x)$ for all $x \in X - Y$. The following theorem tells how to derive this.

THEOREM 6.4.5
Let $x = \mathtt{X}_1, \ldots, \mathtt{X}_n$ be variables, $1 \leq k \leq n$, and $\tau \in \mathrm{T}(\mathrm{Walk}[k](x))$. Furthermore, $S \subseteq \{1, \ldots, n\}$ such that $\max S - \min S \leq k$,[1] and $v \in V$.
Then,

$$\sum_{j \in S} \#\,\mathrm{cwalks}[k](x|j \mapsto v) = \#\,\mathrm{cwalks}[k](x)$$

implies that $\tau_j = v$ for exactly one $j \in S$.

PROOF
Let n, x, k, τ, S, and v be defined as in the theorem.
Let $j \in S$ and $p \in \mathrm{cwalks}[k](y|j \mapsto v)$. Since $\max S - \min S \leq k$, we know that $p_{j'} = v$ if and only if $j = j'$ for all $j' \in S$. Hence, the sets $\mathrm{cwalks}[k](y|j \mapsto v)$ are disjoint for $j \in S$. Thus,

$$\sum_{j \in S} \#\,\mathrm{cwalks}[k](y|j \mapsto v) = \#\,\mathrm{cwalks}[k](y)$$

[1] $\max S$ ($\min S$) denotes the largest (smallest) element in S due to the canonical ordering of natural numbers, respectively.

implies

$$\biguplus_{j \in S} \text{cwalks}[k](y|j \mapsto v) = \text{cwalks}[k](y),$$

i.e., for every walk $p \in \text{cwalks}[k](y)$, $p_j = v$ for exactly one $j \in S$.
Finally, since

$$\tau_r \dots \tau_{r+m-1} \in \text{cwalks}[k](y),$$

we get $\tau_j = v$ for exactly one $j \in S$. □

Since for our purpose, the k-avoiding walk propagator always works in presence of an all-different constraints, the k-avoiding walk propagator could handle further propagation due to the combination with this constraint. The justification to do this is given by Theorem 6.4.5.
For tractability one can restrict the subsets S, e.g. to all subsets of successive numbers up to size k.

6.4.4 Employing Distances

Here, we discuss a possible reduction of the value k of a k-avoiding walk constraint in cases, where the beginning and end point is (at least partially) known.

Definition 6.4.6 (Distance)
For $s, t \in V$, we define a walk from s to t *as a walk $p = p_1 \dots p_n$, where $p_1 = s$ and $p_n = t$. Furthermore, we define a distance on nodes by*

$$\text{dist}(s,t) = \min \left\{ \; n > 0 \;\middle|\; p \in \text{walks}(n), s = p_1, p_n = t \; \right\}.$$

Since V is finite, the defined distance can be computed by Dijkstra's shortest path algorithm. Note that $\text{dist}(s,t)$ is neither a metric nor total.
Depending on the distance of first and last nodes of a walk, self-avoidingness can be already guaranteed by k-avoidingness.

Theorem 6.4.7
Let $s, t \in V$ such that $d = \text{dist}(s,t)$ is defined. Let $n > 0$ and $1 \leq k \leq n$ such that

$$d + k \geq n.$$

Then, a walk p of length n from s to t is k-avoiding if and only if it is self-avoiding.

Proof
We fix $s, t \in V$ such that there exists at least on walk from s to t. Then, $d = \text{dist}(s, t)$ is defined. We choose $1 \leq k \leq n$, where $d + k \geq n$.

First, let $p \in \text{walks}[k](n)$ be a walk from s to t. Assume that p is not self-avoiding. Then, exist $1 \leq i \leq j \leq n$, where $j - i > k$ and $p_i = p_j$. Then, $p_1 \dots p_i p_j + 1 \dots p_n$ is a walk of length $n - (j - i)$ from s to t. Now, by the minimality of d, $d \leq n - (j - i)$ holds, i.e. $d < n - k$, which contradicts $d + k \geq n$.

The second direction is obvious, since each self-avoiding walk is k-avoiding. □

In a constraint search, Theorem 6.4.7 allows to replace k'-avoiding walk constraints by more efficiently computed, but semantically equivalent k-avoiding walk constraints, where $k < k'$, whenever k-avoiding implies self-avoiding, due to the theorem. Inversely, if we derive that k-avoiding walks are self-avoiding this allows stronger propagation due to theorem 6.4.5.

Analogously, in a constraint search, one can simplify a k'-avoiding walk propagator by a more efficient k-avoiding one, in situation described by Theorem 6.4.7, while preserving semantical equivalence.

6.4.5 Improving Propagation by Short Self-Avoiding Walk Constraints

In the absence of a k-avoiding walk propagator, the propagation may be improved by adding a number of short self-avoiding walk constraints. For example, one can introduce $n-k+1$ many constraints $\text{SAWalk}(\mathtt{X}_i, \dots, \mathtt{X}_{i+k-1})$ for all $1 \leq i \leq n - k + 1$, i.e. a k-avoiding walk constraint for each run of k variables.

However, note that the hyper-arc consistency of these self-avoiding walk constraints of length k can not guarantee the hyper-arc consistency of a global k-avoiding walk constraint. A contradictory example is provided by Figure 6.2.

Notably, in the special case $k = 3$, the consistency of the 3-ary constraints $\text{SAWalk}(\mathtt{X}_i, \mathtt{X}_{i+1}, \mathtt{X}_{i+2})$ is computed in only time $O(2b \cdot d)$, where d is the domain size of the variable $\mathtt{X}_{i+1}$ and b is the constant number of neighbors of each lattice point, namely $b = 6$ for the cubic lattice and $b = 12$ for the face-centered cubic lattice.

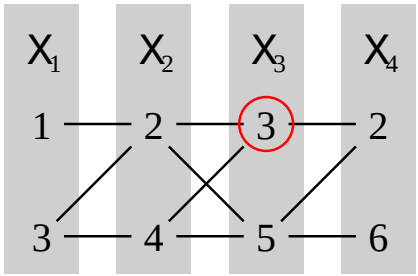


Figure 6.2: A CSP with 4 variables, where $\mathrm{dom}(\mathtt{X}_1) = \{1, 3\}$, $\mathrm{dom}(\mathtt{X}_2) = \{2, 4\}$, $\mathrm{dom}(\mathtt{X}_3) = \{3, 5\}$, and $\mathrm{dom}(\mathtt{X}_4) = \{2, 6\}$. The neighborship of the points 1,...,6 is indicated by connecting lines. The constraints $\mathrm{SAWalk}(\mathtt{X}_1, \mathtt{X}_2, \mathtt{X}_3)$ and $\mathrm{SAWalk}(\mathtt{X}_2, \mathtt{X}_3, \mathtt{X}_4)$ are hyper-arc consistent. However, the constraint $\mathrm{Walk}[3](\mathtt{X}_1, \mathtt{X}_2, \mathtt{X}_3, \mathtt{X}_4)$ is not hyper-arc consistent. Its hyper-arc consistency is violated by $3 \in \mathrm{dom}(X_3)$.

6.5 Results

In the following, we report results of a threading study on the face-centered cubic lattice, which was originally done for [BW01a]. We implemented two threading algorithms. For the first algorithm, we implemented a propagator that handles general walks by reduction to binary walk constraint propagators. For the second algorithm, an experimental, non-optimized version of a propagator for 3-avoiding walks is implemented. The constraint-propagators are implemented as extension to Mozart (Oz 3.0) [Smo95]. For this purpose, Mozart provides a convenient interface for extending the language by propagators written in C++ [MW96].

For benchmarking of the two threading algorithms, the following experiment was performed. Random HP-sequences were threaded to cores of sizes n =25, 50, and 75. Therefore, for each core 50 sequences were randomly generated with n H-monomers and $0.8 \cdot n$ P-monomers. Additionally, we threaded 50 random sequences of length 160 to a core of size 100. We also threaded some randomly generated sequences of length 180 to this core. For each sequence, the threading is performed by both algorithms.

Both algorithms thread the very majority of the test sequences successfully. The results show that the combination of the walk constraint with the all-different constraint yields significantly better propagation even for the strong relaxation of only 3-avoiding walks. Both algorithms successfully threaded all of the 50 sequences to the core of size 25 (which means a sequence length

of 45). For longer sequences, the second algorithm succeeds for significantly more sequences than the first one. Furthermore, it often finds a solution in less search nodes (in our study, up to a factor of 303). The results are summarized in Table 6.1.

input size		algorithm 1		algorithm 2	
core size	seq. length	fails	avg. nodes	fails	avg. nodes
25	45	0%	36	0%	36
50	90	12%	970	2%	103
75	135	20%	586	8%	513
100	160	60%	1468	50%	598

Table 6.1: Threading of random sequences to cores of four different sizes. The table shows size of the core, the length of the sequences, the percentage of sequences which could not be threaded successfully within the given time limit by the two algorithms, and the average number of nodes in successfull runs by both algorithms. We choose a time limit of 5 minutes for the first algorithm. The second algorithm is given a longer time limit of 15 minutes, since the walk propagator is experimental and non-optimized.

For additional demonstration, we applied our approach to predict optimal structures for some sequences of length 200. Table 6.2 gives the run-times for threading these sequences.

Sequence	length	runtime
PPPHPHHPHHPPPPHPHPPPPHPHHPPHPHHHHHPPHHPPHHHHHHPPHPP HHPPHPHPHHHHHPHHPHHHPPPHHHPHHPPHPHPPHPPPHPPHPPHPPH HHPHHHPHPPHPHHPHHHHPHPHHHPHHHPPPPPPPHHHHHHPPPPPPPPH HHPPHPHPPPHPHPHPHHPPHHPPPPHHHHHHPPPHHPPPPPHPPPHHPP	200	6.1 s
HPHHPPHPPPPPHHPHPHPHHPPHPPPPPHHHHHHPPPHPPHHHPPHPPPP HHPPHHHPHPHHHPPHPHHPPHPHHPPPPHHPPHPPHHHHPPPPPHHHPP PPHPPPPPPHPPHHPHHHHPHHHHHHHHPPHHPPPHPHHHPHHHHHPHHP HHHPHPHHPPPPHPHHPHHHPHPPPPHPPPPPPHPHHHHHPHHPPPHPPH	200	52.6 s
HPHHHPHHPHPHPPPHHHHHPHPHPHHHHPPPHHPPPPPPHHPPPPHPHH HPPPPHPPPHHPHHPPPHPPHPPPHHHHPHHPHPPPPHHPPPHHPPHPPP HPPHHHPHHHPHPPHPHHHHPPHHPPPPHHHPHHPPHPPHHHHPPHPHPP HPHPPPPPHPHPHHHHHHHPHPHHHHHHPHHPPPPHPPPPHPPPHHHPHH	200	17.6 s

Table 6.2: Example runtimes for the threading of 3 random sequences of length 200 onto a hydrophobic core of size 100 in the FCC lattice (on a Pentium 4 at 2.4Ghz).

Finally, we give some examples of native structures for sequences with 100 H-monomers in Figure 6.3, which are obtained by threading the sequences to

cores of size 100 with maximally many contacts. Figure 6.5 shows a three-dimensional view of the given native structure for sequence S4 in Figure 6.3.

S1 HHHHPHHHHHHPPHHPHHHHHHHHPHHPHHHHHHHHHHPPHHPPPPPHHPHHHHHHHHPHPPHHPP
PHHHHHHHHPHHHHHHPPHHHHHHHPHPPHHHHHHHHHPPHHPPPHHHHHHHPHHPHHHHHHPP
HHHH
$N_e^+ES_w^+N_e^+N_e^-S_e^-S_w^+ES_w^-WS_w^-NN_w^-SS_w^+N_w^+ES_e^+N_e^+S_e^+N_e^-ES_w^+EEN_w^-N_w^-N_w^-N_w^-S_w^-S_w^+SN_w^-S_w^-S_w^+S_w^+E$
$N_e^-S_w^-N_e^-EN_w^+NS_e^+N_e^+ES_w^+S_w^+WS_w^+WS_w^-N_e^-ES_e^+N_e^-N_e^-N_e^-S_e^-WS_w^+S_w^+S_w^+EN_e^+N_w^+S_w^+SWN_w^-N_e^-S_w^-S_w^+$
$S_w^-N_e^-N_w^-EEN_e^-EN_e^+N_e^+N_e^+S_w^+S_e^+WS_w^+SSS_w^-WNNS_w^-SS_w^-NN_e^-S_w^-N_e^-EN_e^+N_e^+N_e^+ES_w^+S_e^+WWSN_w^-$
$S_w^-S_w^-N_e^-N_e^+EN_e^+N_e^-S_e^-WS_w^-SS_w^+NS_w^+N_e^+N_e^+ES_w^-SN_w^-N_e^-N_e^+N_e^+$

S2 HHPPHPHHHHHHHHHHHPHPPPPHHHPPPHHHHHPPHHHHHPPHHHHHPPHHHHHPPHHHHHPHHHHHPHHHHHP
PPHHPPPHHHHHHHHHPHPPHHHPPPHHHHHPPHHHHHHHPPPHHPPHHHHHPPPHHHHHHHHHPHP
PHHHHHHHPPPHHHPPHHP
$S_w^-EN_e^-WSSN_w^+WN_e^+S_e^+N_e^-S_e^+N_e^-EN_e^-WS_w^-N_w^-WS_w^+S_w^-S_e^+N_e^-S_e^-N_e^+EN_e^+S_w^+WS_w^+S_w^+WN_e^-S_e^-EN_e^+$
$N_e^+S_w^+S_w^+S_w^-N_e^-S_w^-WN_w^-S_w^-EEN_e^-N_e^+EN_e^+N_e^+N_e^-S_e^-WS_w^-S_w^-S_w^+N_w^-S_w^-WS_w^+N_e^+S_e^-S_w^+S_w^-S_w^+N_e^+N_w^+EE$
$S_e^-NN_e^+S_e^+N_w^+N_e^+S_w^+SS_w^-NES_e^+NN_e^-N_w^-S_w^-EN_e^-S_e^-WS_w^+SWN_w^+S_w^+SS_w^-NS_w^-N_e^-S_w^-SN_e^-NN_e^+N_e^-ES_e^-$
$N_e^+N_w^+S_w^+SN_w^-S_w^-SS_w^+NS_w^+N_e^+N_e^+N_e^+S_e^+ENN_w^-S_e^-S_w^+SEN_e^-NN_w^-S_e^-WS_w^-S_w^-S_w^+S_w^+N_e^+N_e^+EN_e^-S_w^-SN_w^+$
$SS_w^-NN_e^-$

S3 HHHPPPHHPHHPPPPPHHHHHHHHHPHPPHHPHHPHHHHHHPPPHHHHHHHHHHPPHPHPPHPHPPHHH
PHPPHPHPPPHHHHHHPHHHHPPPHHHPPPPHHPPPHHHPPHHHHPHHHHHPPHHHHHHPPPHHHH
HHPPPHPPHHHHHPHHHHHHHPPHHPPHHH
$N_w^-N_w^-S_w^+N_w^-N_e^-S_w^-S_w^-S_e^-EN_w^+S_w^+N_e^+S_e^-EN_w^-NS_w^+N_e^+ES_w^+N_e^+N_e^-S_e^-N_e^-WWSS_w^+S_w^+ES_e^+WWN_w^-S_w^-E$
$EES_w^-WWWN_e^-ES_e^-N_e^+N_e^-N_e^+NS_e^+S_w^+N_e^+EN_w^+S_w^+S_w^-S_e^-N_e^+N_e^+S_w^+SN_w^-S_e^-N_e^+N_e^-N_e^-N_w^-S_w^-S_w^+ES_e^-$
$N_w^-S_w^-N_w^+S_w^+S_w^-N_e^-S_w^-S_w^+S_w^+S_w^+N_e^+N_e^+S_w^+S_w^+S_w^-NN_e^-WS_e^-S_w^+S_w^-N_e^-N_w^-ES_e^-S_w^+S_w^+S_e^+NS_e^+NS_e^+N_e^+NS_e^+$
$N_e^-EN_e^-N_w^-S_w^-S_w^-S_w^-S_w^+N_e^+EN_w^+S_w^+SSN_w^-NS_w^-S_w^-N_e^-N_e^-N_e^-N_e^+N_e^+EN_w^+S_e^+ES_w^+WS_e^+WS_w^-S_w^-N_e^-N_e^-$
$S_w^-S_w^+S_w^+S_e^-EENNN_w^+S_e^+SN_w^+S_w^-SSN_w^-NN_e^-N_e^+$

S4 HHPPHPHHHHHHHHPPHPHPPHPHPPPPHHHPPPHHPHPHHPPHHHHHPPHHHHPPHHHHPPHHHHH
HPHHHHPPPHHPPPHHHHHHHHHPHPPHHHPPPHHHHHPPHHHHHHHPHPPPHHPPHHHHPHHPPPHPH
HHHHHHPHPPHPPHHHHHHHPHPPPHHHPPHHP
$N_e^-S_w^-N_e^-N_e^+S_e^+S_w^+S_w^-S_w^+S_w^-S_w^-EN_w^-N_e^+N_e^-S_e^+NEN_e^+N_e^+S_w^+S_w^-N_w^+S_w^+S_e^-WS_e^-N_e^-ES_w^+S_w^-S_w^-N_w^+S_w^+E$
$S_w^+S_e^+NN_e^-N_e^+NS_e^-N_e^-N_e^-N_e^-S_w^-S_w^+S_w^-N_e^-S_w^-N_w^+S_w^+N_w^-SS_e^+N_e^+EN_e^+N_e^+N_e^-S_e^-WS_w^-S_w^-S_w^+N_w^+S_w^-WS_w^+$
$N_e^+S_e^-S_w^+S_w^-S_w^+N_e^+N_w^+EES_e^-NN_e^+S_e^+N_w^+N_e^+S_w^+SS_w^-NES_e^+NN_e^-N_w^-S_w^-EN_e^-S_e^-WS_w^+SWN_w^+S_w^+SS_w^-NS_w^-$
$N_e^-S_w^-SN_e^-NN_e^+N_e^-ES_e^-N_e^+N_w^+S_w^+SN_w^-S_w^-SS_w^+NS_w^+S_e^+NS_e^+NN_e^+S_e^+ENN_w^-S_e^-S_w^+SEN_e^-NN_w^-S_e^-WN_e^-$
$SWS_e^-WSS_w^+NS_w^+N_e^+N_e^+EN_e^-S_w^-SN_w^+SS_w^-NN_e^-$

Figure 6.3: Example sequences with 100 H-monomers together with absolute walks of one optimal conformation in the FCC for each sequence. There, the steps of the walk are given by points of the compass. The + and − indices indicate an additional 45° walk out of the plane.

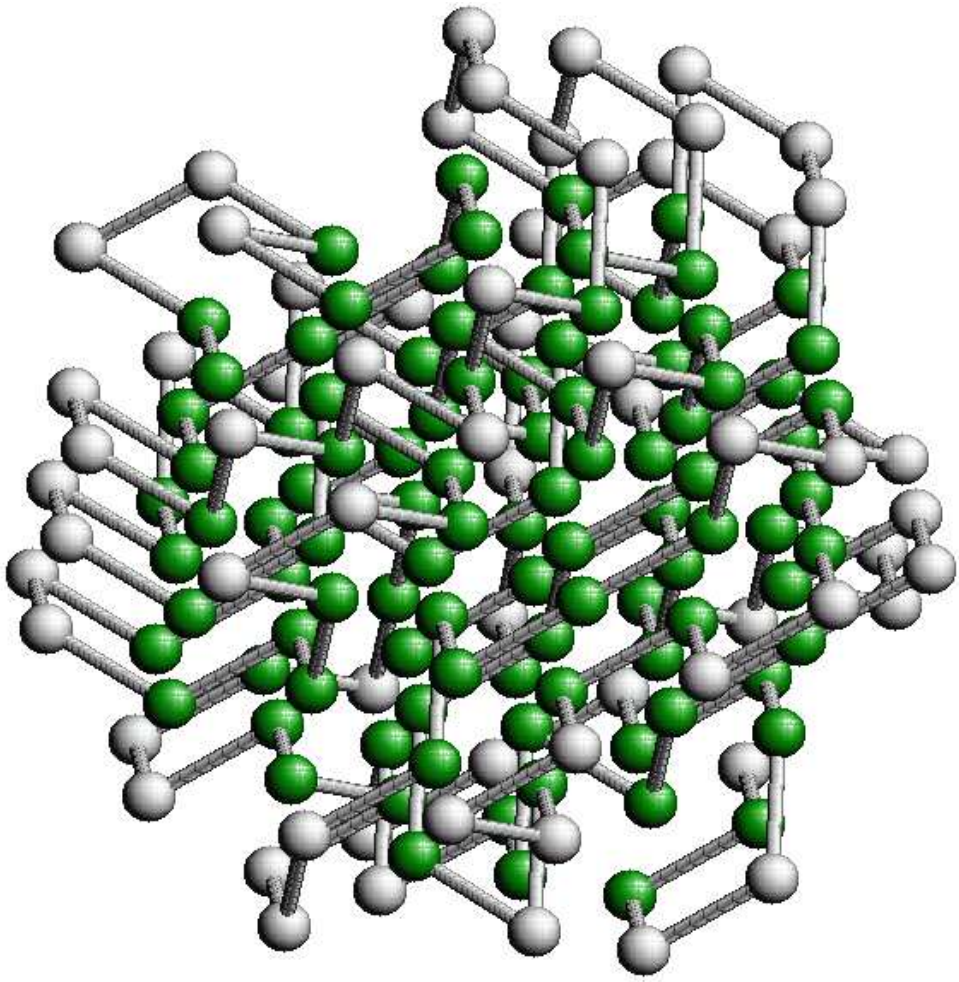


Figure 6.4: An optimal structure for sequence S4 in the FCC. The H-monomers are colored, whereas P-monomers are light grey.

Chapter 7

Applications and Results

In this thesis, we developed the theoretical background for predicting structures in the HP-models of the cubic lattice and the cubic face-centered lattice. This chapter discusses practical applications of the resulting approach CPSP (**C**onstraint-based **P**rotein **S**tructure **P**rediction).
The chapter targets at the following questions:

- How does the approach compare to related methods?
- What is the application range of CPSP?
- Which kind of biological question can be investigated using HP-models?

In particular the third question is topic of ongoing research. Here, the chapter will mainly contribute by giving examples.
Before, we will compare CPSP to existing structure prediction methods by some key characteristics. Then, we will discuss and recall the available implementations of CPSP. For assessing the application range of CPSP in its current implementation, we perform a series of CPSP runs on random sequences. The results will show

1. the run-time behaviour of our implementation in dependency of the degeneracy of the input sequence, i.e. the number of its optimal structures, and
2. the distribution of degeneracy in the three-dimensional HP-models, which is quantified in this thesis for the first time.

We shortly discuss implications and strategies for the further application of CPSP.

Finally, we give two application examples of CPSP that explore protein-like sequences in the cubic HP-model.
Naturally occuring proteins usually have a stable native structure. However, it is a well-known problem in sequence design that artificially synthesized chains of amino acids commonly do not fold into a single native state. The situation is rather similar in the HP-model. There, most sequences in the HP-model behave like artificially designed sequences, since they do not have a unique ground state. Therefore, only HP-sequences with a single optimal structure are considered protein-like.

7.1 Comparison to Related Work

In Table 7.1, we give a comparison to other structure prediction approaches in simplified protein models. Usually, there are no search times reported in the literature. For that reason, we have listed the maximal sequence length that is handled according to the references. Beside the HP model on the two-dimensional square lattice, three-dimensional cubic lattice, and FCC lattice, there are models that distinguish a larger number of amino acids and thus more types of interactions than just the hydrophobic/hydrophobic interactions in the HP-model. These models are commonly called "Hetero"models. Due to the complex energy function, exact structure prediction is not possible in these models. Usually one has to introduce artificial restrictions for computational feasibility, e.g. the restriction to compact conformations. However, this counteracts the benefits of a more sophisticated energy function.
In [SG90, SSK94], the interactions were even generated by a random model resulting in one specific type of interaction for *every* pair of amino acids. Models of this kind are used to make prediction about general properties of the protein folding problem.
When comparing the sequence lengths in Table 7.1, it is important to keep in mind the type of the algorithm, which is specified in the last two columns. All kinds of approaches are represented, including complete enumeration, which all are necessarily restricted. The enumeration approaches either can be applied only to small sequence lengths (≤ 18), or to models, where the search space has been restricted artificially. An example here is the approach by [SG90, SSK94], where only maximally compact conformations are investigated. Namely, only conformations on a $3 \times 3 \times 3$ cube are taken into account, which drastically reduces the search space. This implies, that they consider only sequences of length 27, which equals the number of positions in a $3 \times 3 \times 3$ cube. For this model, one has to enumerate only all 103,346

Structure Prediction Algorithms

Authors	Model	Dim.	maxlen	Algorithm	Comment
Shakhnovich et al. [SG90] and Sali et al. [SSK94]	cubic Hetero (max. compact)	3	27	compl. enum	fixed shape
Dinner et al. [DSK96]	cubic Hetero (max. compact)	3	125	compl. enum	fixed shape
Yue&Dill [YD93]	cubic HP	3	36	b&b	proves optimum
Yue&Dill [YD95]	cubic HP	3	88	b&b	proves optimum
Xia et al. [XHLS00]	tetrahedral Hetero	3	?	enumeration	restricted shape
Kaya&Chan [KC00]	cubic Hetero	3	55	monte carlo	
Cui et al. [CWBBC02]	square HP	2	18	compl. enum	

Approximation Algorithms

Authors	Model	Dim.	maxlen	Algorithm	Comment
Hart&Istrail [HI96]	cubic HP	3	—	approx.	$\frac{3}{8}$ of optimum
Hart&Istrail [HI97a]	FCC-HP side chain	3	—	approx.	86% of optimum
Agarwala et al. [ABD$^+$97]	FCC-HP	3	—	approx.	$\frac{3}{5}$ of optimum

Table 7.1: Results for different lattice models by other groups.

maximally compact conformations [SG90]. In a later work, this was extended to $5 \times 5 \times 5$ cube for sequence length 125 [DSK96].

The protein structure prediction problem is reported to be difficult also for inexact, heuristic methods. For example, in [YFT$^+$95] a Monte Carlo optimization procedure failed to predict the minimal HP-energy of all but one of the ten test HP-sequences of length 48. [BFG$^+$98] reports on an advanced Monte Carlo strategy, called PERM, that reached the ground state for all of the same ten sequences. However, the approach is slow and can not prove the global optimality of found local optima. Of course, it is also not possible to completely enumerate all optimal solutions by stochastic search methods. In Table 7.2 we compare the run-times of PERM and CPSP for predicting the minimal energy of the ten sequences of 48 monomers, which were originally given in [YFT$^+$95] and termed "Harvard sequences".

Finally, we compare our work with the CHCC-algorithm [YD93, YD95], which is the only other approach that can find optimal conformations in the cubic lattice HP-model and in the same time prove their optimality. The HP-model is not designed to generate one *single* minimal energy conformation for each sequence. Instead, commonly there are a lot of minimal energy conformations, suggesting possible topologies for a protein. The number of this minimal energy conformations for a specific sequences *seq* is called the *degeneracy* of *seq*. In [YFT$^+$95], Yue et al. have given a lower bound on the degeneracy of the ten "Harvard sequences". We have largely improved

No.	Sequence	CPSP	PERM
1	$HPH_2P_2H_4PH_3P_2H_2P_2HPH_3PHPH_2P_2H_2P_3HP_8H_2$	0.1 s	6.9 min
2	$H_4PH_2PH_5P_2HP_2H_2P_2HP_6HP_2HP_3HP_2H_2P_2H_3PH$	0.1 s	40.5 min
3	$PHPH_2PH_6P_2HPHP_2HPH_2PHPHP_3HP_2H_2P_2H_2P_2HPHP_2HP$	4.5 s	100.2 min
4	$PHPH_2P_2HPH_3P_2H_2PH_2P_3H_5P_2HPH_2PHPHP_4HP_2HPHP$	7.3 s	284.0 min
5	$P_2HP_3HPH_4P_2H_4PH_2PH_3P_2HPHPHP_2HP_6H_2PH_2PH$	1.8 s	74.7 min
6	$H_3P_3H_2PHPH_2PH_2PH_2PHP_7HPHP_2HP_3HP_2H_6PH$	1.7 s	59.2 min
7	$PHP_4HPH_3PHPH_4PH_2PH_2P_3HPHP_3H_3P_2H_2P_2H_2P_3H$	12.1 s	144.7 min
8	$PH_2PH_3PH_4P_2H_3P_6HPH_2P_2H_2PHP_3H_2PHPHPH_2P_3$	1.5 s	26.6 min
9	$PHPHP_4HPHPHP_2HPH_6P_2H_3PHP_2HPH_2P_2HPH_3P_4H$	0.3 s	1420.0 min
10	$PH_2P_6H_2P_3H_3PHP_2HPH_2P_2HP_2HP_2H_2P_2H_7P_2H_2$	0.1 s	18.3 min

Table 7.2: The ten "Harvard sequences" of [YFT+95]. Time to find and prove minimal energy in the cubic HP-model by CPSP vs. time to only find the same energy by PERM, which is reported in [BFG+98].

these bounds (see Table 7.3). Only for one sequence, CHCC can find as many structures as CPSP. In the remaining cases, CHCC clearly fails to predict the correct degeneracy and only yields rather loose bounds. For an explanation of this behaviour, compare Chapter 4, where we review parts of the CHCC review.
Note that we tested the validity of our results by an independent program. This program takes the list of predicted optimal structures and then checks

- that all structures have the proposed optimal energy and
- that all structures differ from each other.

The first test simply counts the HH-contacts of each structure. For the second test we normalize each structure, which is given as absolute walks, to the lexicographically minimal symmetric walk in order to handle geometrical symmetry. Then, the list of normalized walks is tested for uniqueness using string comparison. This proves independently that the given numbers of structures are indeed a lower bounds on the degeneracy, since in particular the optimal energies are confirmed in the literature.

7.2 Implementation and Application Range

Here, we shortly recall and discuss the currently available implementations of the CPSP method. For assessing the applicability of this implementation, we investigate its run-time behaviour.

Seq. No.	Degeneracy		Run-time
	CHCC	our approach	CPSP
1	1500×10^3	$10,677,113$	138 min
2	14×10^3	$28,180$	37 s
3	5×10^3	$5,090$	33 s
4	62×10^3	$49,442$	104 s
5	54×10^3	$1,954,172$	37 min
6	52×10^3	$1,868,150$	28 min
7	59×10^3	$106,582$	156 s
8	306×10^3	$15,926,554$	166 min
9	10^3	$2,614$	4.5 s
10	188×10^3	$580,751$	11 min

Table 7.3: Degeneracy values for the ten "Harvard sequences"of [YFT+95] by our algorithm compared to lower degeneracy bounds that were computed using CHCC [YFT+95]. Additionally, we show the run-times of CPSP for enumerating the optimal structures. [YFT+95] does not report the run-times for CHCC.

We implemented separate programs for the three steps of our approach. First, there is a stand-alone applications for computing number sequences for the cubic lattice and one for computing frame sequences for the face-centered cubic lattice. The implementation for the FCC was already discussed in Chapter 4, where we also report on its good performance.

For core construction, we implemented two separate constraint programs — one for each lattice. Both programs were written in Oz 3.0 using the programming system Mozart 1.3. Some results are already given in Chapter 5.

The two steps computation of frame/number sequences and core construction are considered precomputation steps for CPSP. The programs for these steps are only used to generate a library of cores.

When this library is generated up to a core size n_{H}, structures of arbitrary sequences with up to n_{H} H-monomers can be predicted by the final step of our approach, namely threading. Only the run-time of this final step is important for many applications of our method, where we predict structures for many differrent sequences of only a few different core sizes.

Since optimal structures can have sub-optimally compact cores, our library has to contain also many cores that are not optimally compact. It is possible to precompute and store sufficiently many sub-optimal cores such that almost all sequences can be threaded to cores in the library. For the few sequences

that do not fit to the available cores, one could generate the missing cores on demand. However, in many applications it seems tolerable to simply ignore such sequences. In this case, the run-time for predicting the structures of a given sequence depends only on the threading step.
The final prediction step is again implemented in Oz. The programm is always given a single sequence *seq* and then predicts structures of *seq*. Controlled by options, it performs various structure prediction tasks, like counting all structures or only finding the best energy. The application systematically threads the given sequence to a series of cores, much like it was described in Figure 3.5. We handle both lattices by a single program. This reflects that changing the lattice in the threading step only requires changing the core library and adapting the definition of neighborship.
Besides our main implementation, there is a web-application, which demonstrates the structure prediction approach via the internet. For a given HP-sequence, it computes its maximal HH-contacts and one optimal structure per hydrophobic core that fits the sequence. The server can be accessed by its address `http://www.bio.inf.uni-jena.de/Software/Prediction`.
Our implementations predict only structures with connected hydrophobic cores. The issue of structures with unconnected hydrophobic cores was discusssed before in Chapter 6. In principle, those structures can be handled by a slight extension of the implemented algorithm.

7.3 Degeneracy of HP-Models

An interesting use of CPSP is determining the number of optimal structures for a sequence. For a sequence *seq*, we denote the number of its optimal structures by $g^{nat}(seq)$. This number is known as the degeneracy of *seq*. When enumerating only structures with connected cores, we do not determine $g^{nat}(seq)$, but $g^{con}(seq)$, which denotes the number of optimal structures of *seq* with connected hydrophobic cores. Usually $g^{con}(seq)$ is a very good bound on $g^{nat}(seq)$. In many cases, this bound is tight, since unconnected hydrophobic cores have less contacts than connected ones. Thus unconnected cores are disfavoured, if not impossible, for optimal structures of most sequences. Due to these considerations we will not distinguish between the two kinds of degeneracy in the following and also omit the superindex in $g^{con}(seq)$.
For practical application, we need to gain insight into the typical run-time behaviour of our implementation. It turns out that the time for predicting structures of a sequence is roughly linear to the number of enumerated

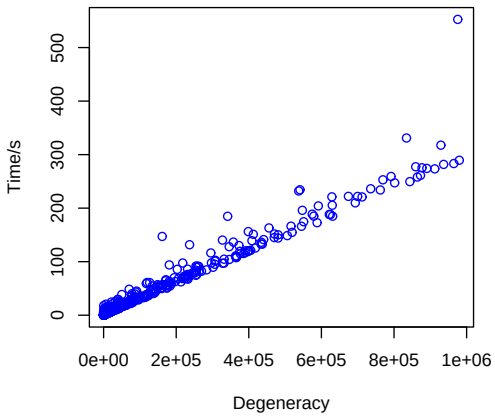


Figure 7.1: Run-time grows only linear with degeneracy. Plot of degneracy in the cubic HP-model vs. time in seconds from a sampling of 1000 sequences of length 27. Only sequences with a degeneracy below 10^6 are plotted.

structures (cf. Figure 7.1). In consequence, the enumeration of all optimal structures of a sequence depends on the number of those structures, i.e. the degeneracy of the sequence. Clues towards the average run-time are provided by a study on the distribution of degeneracy in our HP-models.

For the study of degeneracy, as well as for our later application examples, we use a "trick" to speed up the computation. Instead of determining $\mathrm{g}(seq)$, we will always compute a modification $\mathrm{g}^{\leq t}(seq) = \min(\mathrm{g}(seq), t)$ for a threshold $t \in \mathbb{N}$, where t can be kept rather small in practice. The run-time for computing $\mathrm{g}^{\leq t}$ is roughly linear in t. For determining $\mathrm{g}^{\leq t}(seq)$, we use the feature of our implementation that allows to stop the enumeration after t structures.

Note that by definition, if $\mathrm{g}^{\leq t}(seq) < t$ for a sequence seq and a threshold t, then we know $\mathrm{g}(seq)$ exactly, namely $\mathrm{g}(seq) = \mathrm{g}^{\leq t}(seq)$. Otherwise, if $\mathrm{g}^{\leq t}(seq) = t$, then we know that the degeneracy of seq is at least t.

7.3.1 Degeneracy in the Cubic Lattice

We determined the degeneracies in the cubic lattice for a sample of sequences of size 27 in order to estimate the distribution. Instead of determining the exact degeneracy for each sequence seq, we determined $\mathrm{g}^{\leq 10^6}(seq)$ in order

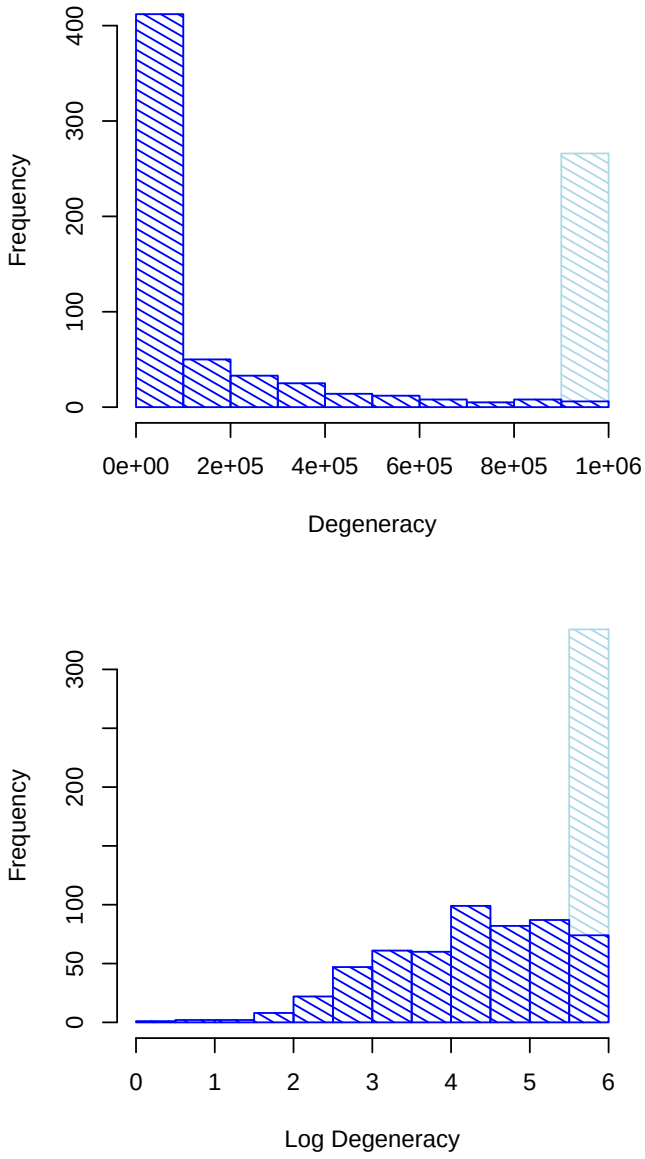


Figure 7.2: Distribution of degeneracy for sequences of length 27 in the cubic HP model. We show a histogram of degeneracy and a histogram of log degeneracy (decadic logarithm). Degeneracies are experimentally determined by sampling 1000 sequences. Only degeneracies below 10^6 are determined exactly. The last column includes all sequences with degeneracy $g \geq 10^6$.

to save computation time. Figure 7.2 shows the resulting histogram for sequences of length 27. In addition to the histogram of degeneracy, we show the histogram of log degeneracy. This histogram is shown for this and the following degeneracy distributions, since it allows a better by-eye comparison of different degeneracy distributions. Also, log degeneracies are related to the stability of a native conformation as we discuss in the next section. The last column of the histograms includes the sequences with higher degeneracy ($> 10^6$). Despite the distribution clearly favours low degeneracies, there are about 30% such sequences of length 27.

7.3.2 Degeneracy in the FCC Lattice

For the face-centered cubic lattice, we performed the same study as in the cubic case before. This time, the sequences have only length 12, which was chosen due to the much higher degeneracy of the FCC and our limited computation time. Figure 7.3 shows the corresponding histogram of degeneracy for the FCC and again in addition the histogram of log degeneracy.
We also compare our results for the FCC, to sequences of length 12 in the cubic HP model. Figure 7.4 compares the two distributions.

7.4 Protein-Like HP-Sequences

It is known that most sequences in the cubic HP-model have many optimal structures. This fact, which is known as *degeneracy* of the HP-model, was shown and made more precise in the last section.
For real proteins it is known that most randomly generated sequences fail to fold into a stable native structure. In particular, this is an important issue in the sequence design problem. The property of native structure stability is reflected in the HP-model. Thermodynamic stability in the HP-model can be computed from the complete *density of states (DOS)*. Here, the DOS $\mathrm{g}_E(seq)$ of a sequence *seq* gives the number of its structures that have energy E. The free energy of the native structure (i.e. one of the optimal structures of *seq*) is then calculated as

$$\Delta G = E_{\min} - k_B T \ln[e^{-E_{\min}/(k_B T)} - \sum_E \mathrm{g}_E(seq) e^{-E/(k_B T)}],$$

where $E_{\min}$ denotes the energy of the native structure, k_B is the Boltzmann constant and T the temperature (see [BBC99]).

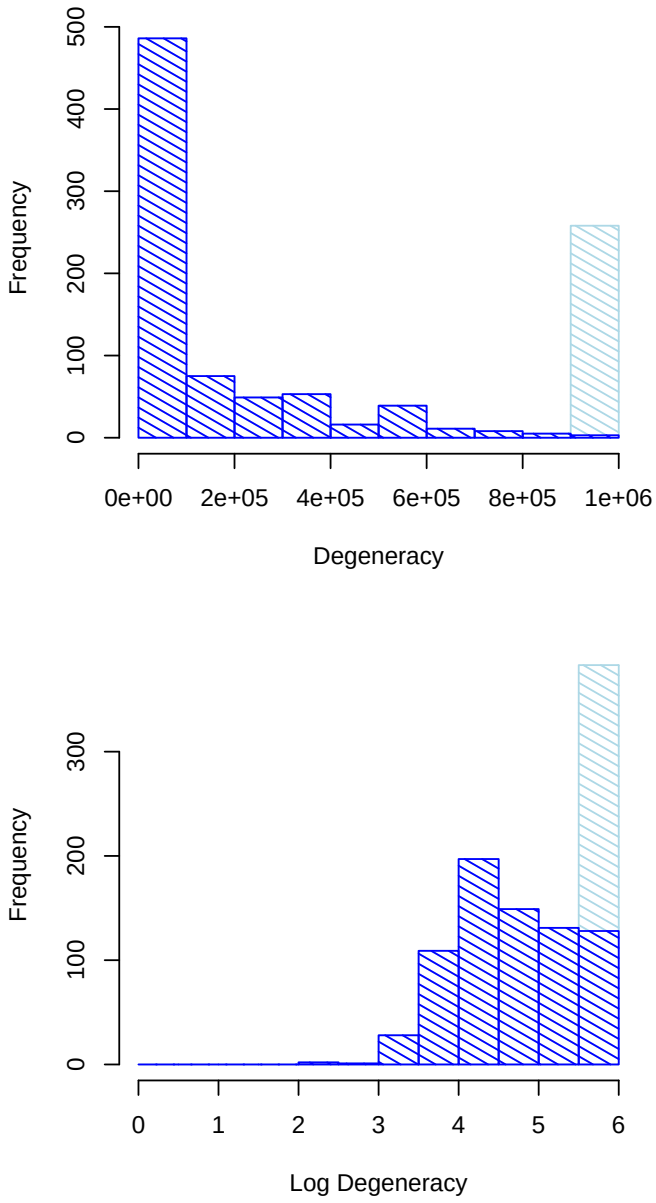
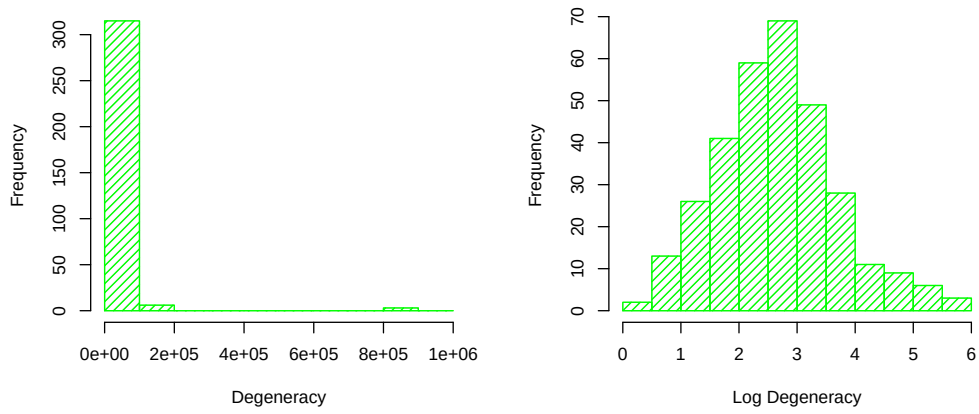


Figure 7.3: Histogram of degeneracies and log degeneracies for sequences of length 12 in the FCC HP-model from sampling 1000 sequences. Only degeneracies below 10^6 are determined exactly. The last column includes all sequences with degeneracy $g \geq 10^6$.

Cubic



FCC

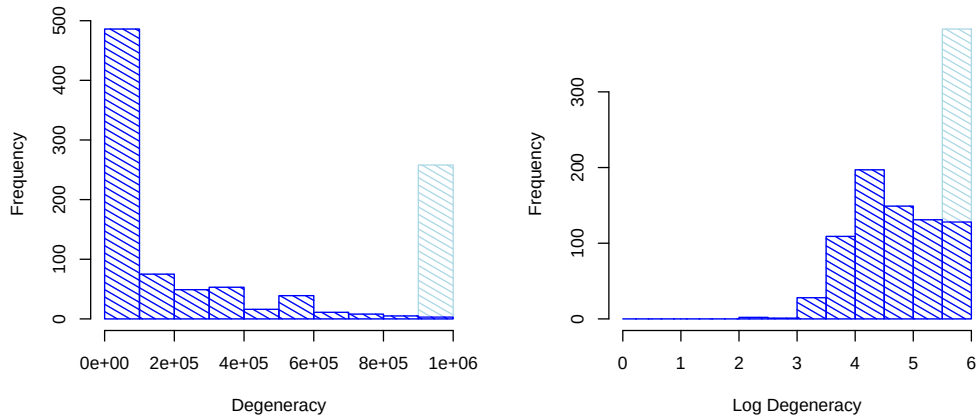


Figure 7.4: Comparison of degeneracy distributions and log degeneracy distribution for cubic and FCC HP-model at length 12.

The more negative ΔG of a sequence *seq*, the more stable is its native structure. The free energy term is strongly dominated by the degeneracy of *seq*. Especially, ΔG is always positive if the degeneracy is greater than one, since then the probability of one native structure falls below 50%.
Due to these considerations, only sequences where $g(seq) = 1$ have a stable native structure. It is reasonable to assume that only those HP-sequences correspond to the sequences in real proteins, which were selected for stability by evolution.
In [YFT+95], the authors found some sequences of length 60-80 with fewer than five optimal structures in the cubic lattice HP-model, but do not know of HP-sequences with a unique ground state. It was unknown, whether such sequences exist in three-dimensional HP-models.
We present an application of CPSP for answering this question. In the same time we devise a method for constructing such protein-like sequences, which shows a way for rational protein design in simplified protein models.
Our approach simulates protein evolution by point mutations of the HP-sequences, where the only driving force is towards lower degeneracy. For evolving a protein-like sequence, we apply an optimization strategy, which relies on exact protein structure prediction for computing degeneracies.
The good performance of this strategy points to a super-funnel-like arrangement of the sequence space (cf. [BBC99]). Due to the *super-funnel hypothesis*, the sequence space is structured into neutral nets, where a *neutral net* consists of sequences that are related by point mutations and encode for the same common structure. The sequences of a neutral net are centered around a prototype sequence that encodes the common structure in the most stable way. Finally, the stability of the common structure decreases with the hamming distance to the prototype sequence. [BBC99] verifies this funnel-like arrangement for the 2D-cubic HP-model and a further two-dimensional lattice model.
The next sub-section describes the simulation of protein evolution for generating low degenerated sequences. After this, we show a second application of CPSP, which points toward the computation of neutral nets in three-dimensional HP-models.

7.4.1 Finding Protein-Like Sequences

The sequences with degeneracy one are computed using a modified **M**onte **C**arlo (MC) optimization procedure for minimizing $\ln(g(seq))$. This method relies on CPSP for computing the degeneracies of sequences in each step.

Note that minimizing the logarithm of degeneracies also minimizes the degeneracy. However, minimizing the logarithms turned out to be more efficient then directly optimizing degeneracy. Presumably this can be attributed to the well suited distribution of log degeneracy.
A search for a protein-like sequence, in the following also called simulation run, starts with guessing a low degenerated sequence, e.g. a sequence *seq* with $\mathrm{g}(seq) < t$ for $t = 5000$. Therefore, we iteratedly choose a random sequence *seq*, where we choose for each position H or P with equal probability, and compute $\mathrm{g}^{\leq \mathrm{t}}(seq)$ until $\mathrm{g}^{\leq \mathrm{t}}(seq) < t$. This last sequence *seq* is chosen as start sequence s_0 for the MC optimization. Then, we apply MC steps to this sequence, which are described below. Thereby, we (virtually) produce a series of sequences $s_0, \ldots, s_n$ until $\mathrm{g}(s_n) = 1$ or n reaches a certain limit on the number of steps. We fix a temperature $T \in \mathbb{R}$ and a cut-off value $0 < c \ll 1$ for all simulation runs. The cut-off value c is introduced for efficiency; as explained soon, c is the minimal probability for accepting a change for the worse in a MC step. For each single MC step, i.e. going from s_i to s_{i+1}, we proceed as follows.

1. Apply a random point mutation to s_i producing s_i^m, where each position for the mutation is equally probable.

2. Compute $\mathrm{g}^{\leq \mathrm{t}}(s_i^m)$, where

$$t = g(s_i) \cdot \exp(-T \cdot \ln(c)). \tag{7.1}$$

   Note that $g(s_i)$ is already known exactly.

3. If $\mathrm{g}^{\leq \mathrm{t}}(s_i^m) < g(s_i)$, then $s_{i+1} = s_i^m$. Else if $\mathrm{g}^{\leq \mathrm{t}}(s_i^m) < t$ then we choose $s_{i+1} = s_i^m$ with a probability

$$\exp \frac{\ln g(s_i^m) - \ln g(s_i)}{-T}. \qquad \text{(Metropolis criterion)}$$

   Otherwise, the sequence is not changed in this step, i.e. $s_{i+1} = s_i$.

The above schema is a modification of the standard Monte Carlo optimization procedure. Instead of computing $g(s_i^m)$ for the mutated sequence, which is required for the standard approach, we compute only $\mathrm{g}^{\leq \mathrm{t}}(s_i^m)$. The threshold t is chosen such that there is a controlled and very limited deviation to the standard MC procedure. Namely, for $g(s_i^m) \geq t$ the mutated sequence is always declined, but in the standard approach it is accepted with a probability less or equal c.

This claim is shown by the calculation

$$\exp \frac{\ln g(s_i^m) - \ln g(s_i)}{-T} \leq_{(7.1)} exp\frac{\ln[g(s_i) \cdot \exp(-T \cdot \ln(c))] - \ln g(s_i)}{-T}$$
$$= exp\frac{\ln g(s_i) - T \cdot \ln(c)) - \ln g(s_i)}{-T} = c.$$

For efficiency, all values $g^{\leq t}(seq)$ that are computed during one simulation run are cached. If later in the simulation, a degeneracy $g^{\leq t'}(seq)$ of the same sequence *seq* is requested, there are two possibilities. Either $t' \leq t$, then $g^{\leq t'}(seq)$ is easily determined from $g^{\leq t}(seq)$, or $t' > t$, then the value $g^{\leq t'}(seq)$ is computed from the scratch.
Figure 7.5 shows the unique ground states of some sequences of size 64 and 80 that were found following the above strategy.

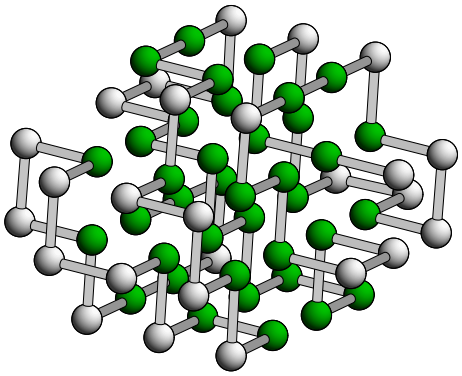
7.4.2 Neutral Environments

Since we know protein-like sequences due to the last sub-section, we can now explore their surrounding in the sequence space. In this sub-section, we describe how to expand a neutral environment around a given sequence with degeneracy one.
The *neutral environment* of a sequence *seq* is a graph of sequences

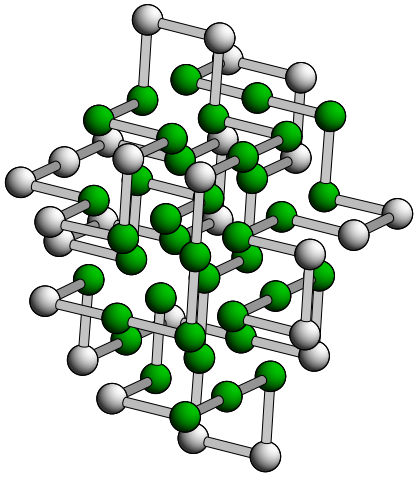
- that have a limited hamming distance d to *seq* (d denotes the *size of the environment*)
- that are connected by point mutations and
- have the unique structure of *seq* in their ground state.

Obviously, the neutral environment of size 0 consists of *seq* alone, i.e. set $E_0 = \{seq\}$. The neutral environment E_i of size i can be generated by adding to the neutral environment E_{i-1} of size $i-1$ all sequences which are related by neutral point mutations to sequences in E_{i-1}. Noteworthy, the added sequences are always related by (neutral) point mutations to sequences in $E_{i-1} - E_{i-2}$ for $i \geq 2$ (respectively, E_0, for $i = 1$), which is used in our efficient implementation.
Here, the crucial step is to decide whether a point mutation, which relates sequences *seq* and *seq*′, is neutral. For our purposes, it is always known that *str* is an optimal structure for one of the sequences. Hence, we test for the other sequence if *str* is also an optimal structure. Instead of enumerating all optimal structures of the sequence, it suffices to determine the energy of the

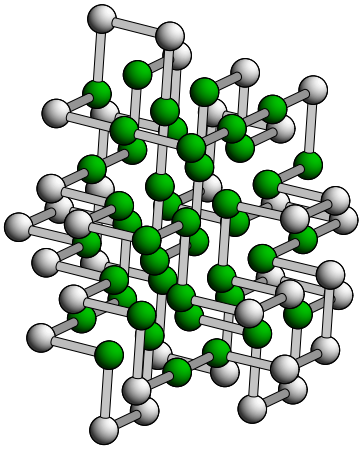
1)



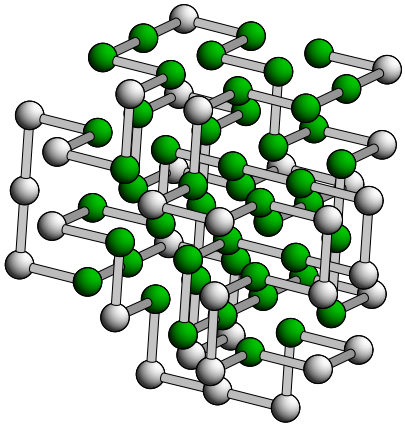
2)



3)



4)



seq_1:HHHPHHPHPPHPPPHPHHPHPPPHPHHHPHPPHHHHHHHHHHPPHHHPHHPHPPHPPHHPHHHP

str_1:UUFDDDBRBLDFLUFULDBDBRUBURUUFRDLDLFUFULDBLDRBDBUUUFLFRURDBBDFLL

seq_2:HHHHPPHPPHPPPHPHPHHHPPHHHPHHHPHPHPHPHHHPPHHPPHHHHPPPHHHHPPHHHPHH

str_2:URFRULUBDRDDLDLBURUBLFLDDFFRRUFUULLBDDLBRULURBRRULFDFDFFLBDDRUB

seq_3:HPPHPHHHPPHPHPHPHHPPPHHPPHHPPPHPHHHHHHPPHPHHHHPHHPPHHPPHHHPHPHHHPHPPPHPPHPPHPPHH

str_3:FDBRUUBRDLDLDLFUULUBRUURDRUBLDBDFFDBBBRFDLLUUUFDLDRFLDRBDRURUUBUFRFFLFLBULDFDBR

seq_4:HPHHHPHPPHPPHPPHPPHPPPHPHPHHHHHPPHHHHHHHPHHHPHHHHHPHHHPHPHPPPHHPHPPHHHHPHHPHPPHH

str_4:LDDRDFDLULFRDFULURFRRBDBRULBLLLFRRDFUUULDBBURDBRURDDLDFRURFFLUULFDBBBURDFULULDD

Figure 7.5: Some HP-sequences of 64 and 80 monomers with unique ground state. Besides the graphical representation, we give the HP-sequences seq_i and the absolute walks of the structures str_i (**F**orward, **B**ackward, **L**eft, **R**ight, **U**p, **D**own).

optimal structures, which can be done much faster. Then, we test whether a HP-protein with this sequence and structure *str* has the same energy. An example of a neutral environment is shown in Figure 7.6.

Conclusion

The given results and application examples demonstrate the applicability of the introduced protein structure prediction approach CPSP in the ongoing research on proteins. Due to its speed and flexibility, which is superior to all previous approaches to exact structure prediction, CPSP provides new research opportunities. As we have shown, CPSP promotes our understanding of widely-used protein models. By the ability of CPSP to predict structures in the FCC lattice, the deployment of exact models for the structure prediction of real proteins seems to come into reach. Finally, CPSP enables the use of unrestricted, three-dimensional protein models for the exploration of protein evolution and kinetics.

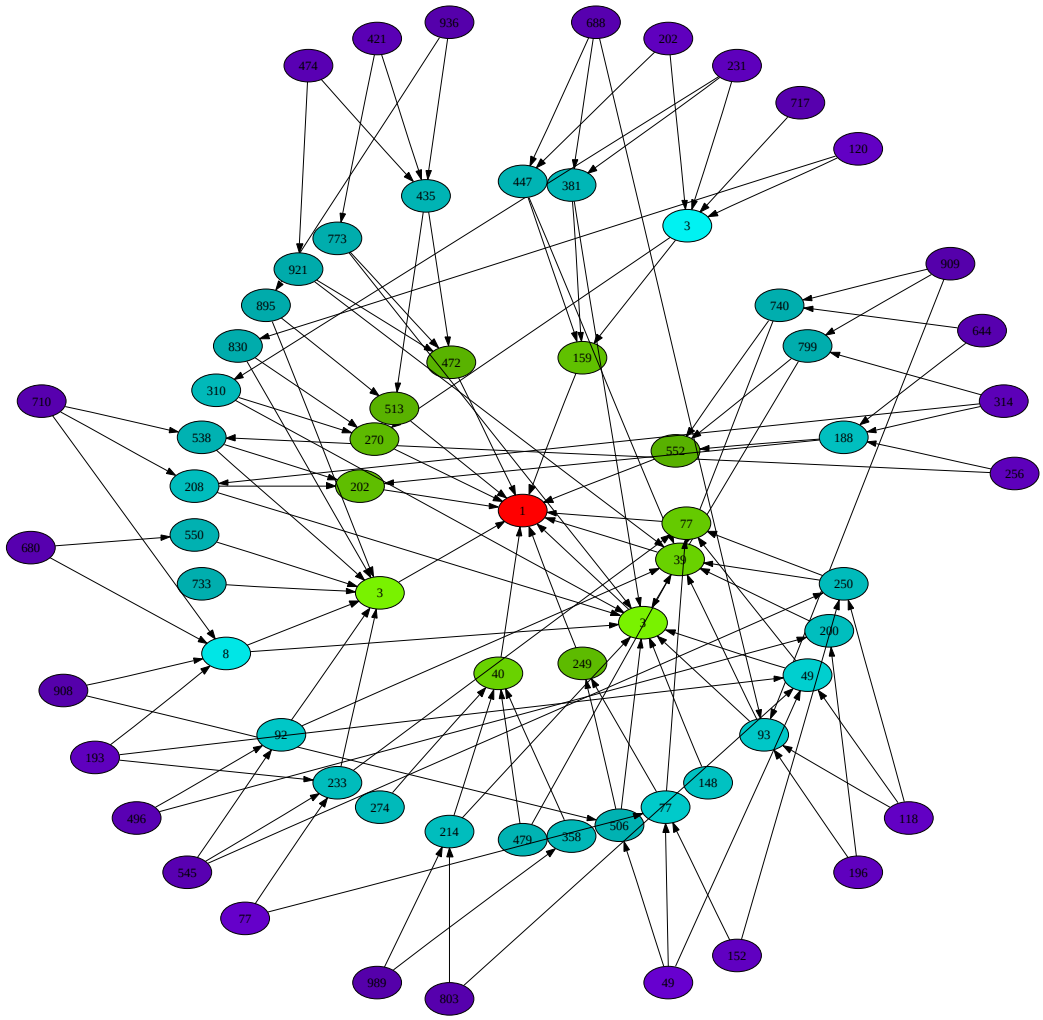


Figure 7.6: Neutral environment of a sequence with unique ground state in the cubic lattice. Each node represents one sequence and is labeled by its degeneracy. Edges are drawn between sequences that are related by exactly one point mutation. The figure shows only sequences *seq* of the environment of size 3 with $g(seq) \leq 1000$.

Bibliography

[ABD+97] R. Agarwala, S. Batzoglou, V. Dancik, S. E. Decatur, S. Hannenhalli, M. Farach, S. Muthukrishnan, and S. Skiena. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model. *Journal of Computational Biology*, 4(3):275–96, 1997.

[AGS95] V. I Abkevich, A. M Gutin, and E. I Shakhnovich. Impact of local and non-local interactions on thermodynamics and kinetics of protein folding. *Journal of Molecular Biology*, 252(4):460–71, 1995.

[AGS97] V. I. Abkevich, A. M. Gutin, and E. I. Shakhnovich. Computer simulations of prebiotic evolution. In *Proc. of the Pacific Symposium on Biocomputing 1997 (PSB 1997)*, pages 27–38, 1997.

[Bac98a] Rolf Backofen. Constraint techniques for solving the protein structure prediction problem. In Michael Maher and Jean-Francois Puget, editors, *Proceedings of* 4^{th} *International Conference on Principle and Practice of Constraint Programming (CP'98)*, volume 1520 of *Lecture Notes in Computer Science*, pages 72–86. Springer Verlag, 1998.

[Bac98b] Rolf Backofen. Using constraint programming for lattice protein folding. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing (PSB'98)*, volume 3, pages 387–398, 1998.

[Bac00a] Rolf Backofen. *Optimization Techniques for the Protein Structure Prediction Problem.* Habilitation, Ludiwg-Maximilians-Universität München, 2000.

[Bac00b] Rolf Backofen. An upper bound for number of contacts in the HP-model on the Face-Centered-Cubic Lattice (FCC). In R. Giancarlo and D. Sankoff, editors, *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM 2000)*, number 1848 in LNCS, pages 277–292, Montréal, Canada, 2000. Springer-Verlag, Berlin.

[Bac01] Rolf Backofen. The protein structure prediction problem: A constraint optimisation approach using a new lower bound. *Constraints*, 6:223–255, 2001.

[Bac04] Rolf Backofen. A polynomial time upper bound for the number of contacts in the hp-model on the face-centered-cubic lattice (fcc). *Journal of Discrete Algorithms*, 2(2):161–206, 2004.

[BB97] Erich Bornberg-Bauer. Chain growth algorithms for HP-type lattice proteins. In *Proc. of the* 1^{st} *Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 47 – 55. ACM Press, 1997.

[BBC99] E. Bornberg-Bauer and H. S. Chan. Modeling evolutionary landscapes: mutational stability, topology, and superfunnels in sequence space. *Proc. Natl. Acad. Sci. USA*, 96(19):10689–94, 1999.

[BFG+98] U Bastolla, H Frauenkron, E Gerstner, P Grassberger, and W Nadler. Testing a new Monte Carlo algorithm for protein folding. *Proteins*, 32(1):52–66, 1998.

[BJB02a] Zerrin Bagci, Robert L. Jernigan, and Ivet Bahar. Residue coordination in proteins conforms to the closest packing of spheres. *Polymer*, 43:451–459, 2002.

[BJB02b] Zerrin Bagci, Robert L. Jernigan, and Ivet Bahar. Residue packing in proteins: Uniform distribution on a coarse-grained scale. *Journal of Chemical Physics*, 116:2269–2276, 2002.

[BL98] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 5(1):27–40, 1998.

[BW99] Rolf Backofen and Sebastian Will. Excluding symmetries in constraint-based search. In Joxan Jaffar, editor, *Proceedings of*

5^{th} *International Conference on Principle and Practice of Constraint Programming (CP'99)*, volume 1713 of *Lecture Notes in Computer Science*, pages 73–87, Berlin, 1999. Springer–Verlag.

[BW01a] Rolf Backofen and Sebastian Will. Fast, constraint-based threading of HP-sequences to hydrophobic cores. In *Proceedings of* 7^{th} *International Conference on Principle and Practice of Constraint Programming (CP'2001)*, volume 2239 of *Lecture Notes in Computer Science*, pages 494–508, Berlin, 2001. Springer–Verlag.

[BW01b] Rolf Backofen and Sebastian Will. Optimally compact finite sphere packings — hydrophobic cores in the FCC. In *Proc. of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM2001)*, volume 2089 of *Lecture Notes in Computer Science*, pages 257–272, Berlin, 2001. Springer–Verlag.

[BWBB99] Rolf Backofen, Sebastian Will, and Erich Bornberg-Bauer. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics*, 15(3):234–242, 1999.

[BWC00] Rolf Backofen, Sebastian Will, and Peter Clote. Algorithmic approach to quantifying the hydrophobic force contribution in protein folding. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing (PSB 2000)*, volume 5, pages 92–103, 2000.

[CBB02] H.S. Chan and E. Bornberg-Bauer. Perspectives on protein evolution from simple exact models. *Applied Bioinformatics*, 1:121–144, 2002.

[CGP+98] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3):423–65, 1998.

[Cip98] Barry Cipra. Packing challenge mastered at last. *Science*, 281:1267, 1998.

[CWBBC02] Yan Cui, Wing Hung Wong, Erich Bornberg-Bauer, and Hue Sun Chan. Recombinatoric exploration of novel folded

structures: a heteropolymer-based model of protein evolutionary landscapes. *Proc. Natl. Acad. Sci. USA*, 99(2):809–14, 2002.

[DBF02] A. Dovier, M. Burato, and F. Fogolari. Using secondary structure information for protein folding in CLP(FD). In *Proc. of Workshop on Functional and Constraint Logic Programming*, volume ENTCS vol. 76, 2002.

[DBY+95] K.A. Dill, S. Bromberg, K. Yue, K.M. Fiebig, D.P. Yee, P.D. Thomas, and H.S. Chan. Principles of protein folding – a perspective of simple exact models. *Protein Science*, 4:561–602, 1995.

[DFC93] K. A. Dill, K. M. Fiebig, and H. S. Chan. Cooperativity in protein-folding kinetics. *Proc. Natl. Acad. Sci. USA*, 90:1942 – 1946, 1993.

[DSK96] A. R Dinner, A Sali, and M Karplus. The folding mechanism of larger model proteins: role of native structure. *Proc. Natl. Acad. Sci. USA*, 93(16):8356–61, 1996.

[FFHS00] C. Flamm, W. Fontana, I. L. Hofacker, and P. Schuster. RNA folding at elementary step resolution. *RNA*, 6(3):325–38, 2000.

[FHSW02] Christoph Flamm, Ivo L. Hofacker, Peter F. Stadler, and Michael T. Wolfinger. Barrier trees of degenerate landscapes. *Z.Phys.Chem*, 216:155–173, 2002.

[Fre82] Eugene C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29:24–32, 1982.

[GG96] S Govindarajan and R. A Goldstein. Why are some proteins structures so common? *Proc. Natl. Acad. Sci. USA*, 93(8):3341–5, 1996.

[GG97] S. Govindarajan and R. A. Goldstein. The foldability landscape of model proteins. *Biopolymers*, 42(4):427–438, 1997.

[Gra04] Peter Grassberger. Sequential Monte Carlo methods for protein folding. In *NIC Symposium 2004*, Juelich, oct 2004.

[GSA+98] A. Gutin, A. Sali, V. Abkevich, M. Karplus, and E.I. Shakhnovich. Temperature dependence of the folding rate in a

simple protein model: Search for a "glass transition. *J. Chem. Phys.*, 108:6466–6483, 1998.

[Heu03] Volker Heun. Approximate protein folding in the HP side chain model on extended cubic lattices. *Discrete Appl. Math.*, 127(1):163–177, 2003.

[HI96] W. E. Hart and S. C. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal. *Journal of Computational Biology*, 3(1):53–96, 1996.

[HI97a] W. E. Hart and S. Istrail. Lattice and off-lattice side chain models of protein folding: linear time structure prediction better than 86% of optimal. *Journal of Computational Biology*, 4(3):241–59, 1997.

[HI97b] W. E. Hart and S. Istrail. Robust proofs of NP-hardness for protein folding: general lattices and energy potentials. *Journal of Computational Biology*, 4(1):1–22, 1997.

[HL96] D. A. Hinds and M. Levitt. From structure to sequence and back again. *Journal of Molecular Biology*, 258(1):201–9, 1996.

[IT02] Anders Irbäck and Carl Troein. Enumerating designing sequences in the HP model. *J. Biol. Phys.*, 28, 2002.

[JCSM03] Tianzi Jiang, Qinghua Cui, Guihua Shi, and Songde Ma. Protein folding simulations of the hydrophobic-hydrophilic model by combining tabu search with genetic algorithms. *Journal of Chemical Physics*, 119(8):4592–4596, 2003.

[KC00] H. Kaya and H. S. Chan. Energetic components of cooperative protein folding. *Physical Review Letters*, 85(22):4823–6, 2000.

[KL99] P. Koehl and M. Levitt. A brighter future for protein structure prediction. *Nat. Struct. Biol.*, 6(2):108–11, 1999.

[LD89] Kit Fun Lau and Ken A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *American Chemical Society*, 22:3986 – 3997, 1989.

[Lev69] Cyrus Levinthal. How to fold graciously. In J. T. P. DeBrunner and E. Munck, editors, *Mossbauer Spectroscopy in Biological Systems: Proceedings of a meeting held at Allerton House*,

pages 22–24, Monticello, Illinois, 1969. University of Illinois Press.

[LTW02] H. Li, C. Tang, and N. Wingreen. The designability of protein structures: A lattice-model study using the Miyazawa-Jernigan matrix. *Proteins*, 49:403, 2002.

[LW01] Faming Liang and Wing Hung Wong. Evolutionary Monte Carlo for protein folding simulations. *Journal of Chemical Physics*, 115(7):3374–3380, August 15 2001.

[MJ96] S. Miyazawa and R. L. Jernigan. Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading. *Journal of Molecular Biology*, 256(3):623–44, 1996.

[MJH+00] D. MacDonald, S. Joseph, D. L. Hunter, L. L. Moseley, N. Jan, and A. J. Guttmann. Self-avoiding walks on the simple cubic lattice. *J. Phys. A: Math. Gen.*, 33:5973–5983, 2000.

[MS96] Neil Madras and Gordon Slade. *The Self-Avoiding Walk*. Probability and Its Applications. Birkhäuser Boston, 1996.

[MW96] Tobias Müller and Jörg Würtz. Interfacing propagators with a concurrent constraint language. In *JICSLP96 Post-conference workshop and Compulog Net Meeting on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages*, pages 195–206, 1996.

[PL95] B. H. Park and M. Levitt. The complexity and accuracy of discrete state models of protein structure. *Journal of Molecular Biology*, 249(2):493–507, 1995.

[Reg94] Jean-Charles Regin. A filtering algorithm for constraints of difference. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, pages 362–367, 1994.

[SD03] Jack Schonbrun and Ken A. Dill. Fast protein folding kinetics. *Proc. Natl. Acad. Sci. USA*, 100(22):12678–82, 2003.

[SFSH94] 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. Royal Society London B*, 255(1344):279–84, 1994.

[SG90] E. I. Shakhnovich and A. M. Gutin. Enumeration of all compact conformations of copolymers with random sequence of links. *Journal Chemical Physics*, 8:5967–5971, 1990.

[Slo98] Neil J. A. Sloane. Kepler's conjecture confirmed. *Nature*, 395(6701):435–6, 1998.

[Smo95] Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.

[SR01] A. Sikorski and P. Romiszowski. Monte Carlo simulations of protein-like heteropolymers. *Acta Biochimica Polonica*, 48(1):77–81, 2001.

[SSG+00] C. T. Shih, Z. Y. Su, J. F. Gwan, B. L. Hao, C. H. Hsieh, and H. C. Lee. Mean-field HP model, designability and alpha-helices in proteinstructures. *Physical Reviews Letters*, 84(2):386–9, 2000.

[SSK94] A. Sali, E. Shakhnovich, and M. Karplus. Kinetics of protein folding. A lattice model study of the requirements for folding to the native state. *Journal of Molecular Biology*, 235(5):1614–36, 1994.

[TG00a] D. M. Taverna and R. A. Goldstein. The distribution of structures in evolving protein populations. *Biopolymers*, 53(1):1–8, 2000.

[TG00b] D.M. Taverna and R.M. Goldstein. The evolution of duplicated genes considering protein stability constraints. In *Proc. of the Pacific Symposium on Biocomputing (PSB 2000)*, volume 5, pages 66–77, 2000.

[UM93a] R. Unger and J. Moult. A genetic algorithm for 3D protein folding simulations. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 581–588, San Mateo, CA, 1993. Morgan Kaufmann.

[UM93b] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231(1):75–81, 1993.

[UM96] R Unger and J Moult. Local interactions dominate folding in a simple protein model. *Journal of Molecular Biology*, 259(5):988–94, 1996.

[WBBC04] R. Wroe, E. Bornberg-Bauer, and H. S. Chan. Comparing folding codes in simple heteropolymer models of protein evolutionary landscape: Robustness of the superfunnel paradigm. *Biophysical Journal*, 2004.

[WFHS99] S. Wuchty, W. Fontana, I. L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49(2):145–65, 1999.

[Wil02] Sebastian Will. Constraint-based hydrophobic core construction for protein structure prediction in the face-centered-cubic lattice. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, and Teri E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing 2002 (PSB 2002)*, pages 661–672, Singapore, 2002. World Scientific Publishing Co. Pte. Ltd.

[XHLS00] Y. Xia, E. S. Huang, M. Levitt, and R. Samudrala. Ab initio construction of protein tertiary structures using a hierarchical approach. *Journal of Molecular Biology*, 300(1):171–85, 2000.

[YD93] Kaizhi Yue and Ken A. Dill. Sequence-structure relationships in proteins and copolymers. *Phys.Rev. E*, pages 2267–2278, 1993.

[YD95] K Yue and KA Dill. Forces of tertiary structural organization in globular proteins. *Proc. Natl. Acad. Sci. USA*, 92(1):146–50, 1995.

[YFT+95] K. Yue, K. M. Fiebig, P. D. Thomas, H. S. Chan, E. I. Shakhnovich, and K. A. Dill. A test of lattice protein folding algorithms. *Proc. Natl. Acad. Sci. USA*, 92(1):325–9, 1995.

[ZS81] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–48, 1981.

Index

Curriculum Vitae

Diplom-Informatiker (Univ.) Sebastian Will
geb. 13.04.1974 in München

1980-1984	Grundschule: Montessori Schule, München
1984-1993	Gymnasium: Theresiengymnasium, München
1993	Abitur
1993-1996	Studium der Biologie, Ludwig-Maximilians Universität (LMU) München
1995	Vordiplom in Biologie
1996-2000	Studium der Informatik, LMU München
2000	Abschluss Diplom-Informatiker (Univ.) (Auszeichnung für das beste Informatik-Diplom des Jahrgangs an der Fakultät) Diplomarbeit am Lehrstuhl für Theoretische Informatik der LMU München bei Prof. Rolf Backofen und Prof. Peter Clote *Symmetry Exclusion in Constraint Search with an Application to Lattice Protein Folding*
2000-2002	Stipendium des Graduiertenkolleg ,,Logik in der Informatik" (GKLI), München (Betreuer: Rolf Backofen)
ab 2002	wissenschaftlicher Mitarbeiter bei Prof. Rolf Backofen am Lehrstuhl für Bioinformatik, Friedrich-Schiller Universtität Jena

List of Publications

[1] Rolf Backofen and Sebastian Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2 no. 4 pp. 681-698, 2004.

[2] Alessandro Dal Palù, Sebastian Will, Rolf Backofen, and Agostino Dovier. Constraint-based protein structure prediction exploiting secondary structure information. In *Convegno Italiano di Logica Computazionale 2004 (CILC 2004)*, 2004.

[3] Rolf Backofen and Sebastian Will. A constraint-based approach to structure prediction for simplified protein models that outperforms other existing methods. In *Proceedings of the 19th International Conference on Logic Programming (ICLP 2003)*, pages 49-71, 2003.

[4] Sebastian Will and Rolf Backofen. Breaking of partial symmetries in the photo and alignment problem. In *Proceedings of the Third International Workshop on Symmetry in Constraint Satisfaction Problems (SymCon 2003)*, pages 187-194, 2003.

[5] Rolf Backofen and Sebastian Will. Excluding symmetries in constraint-based search. *Constraints*, 7 no. 3 pp. 333-349, 2002. Kluwer Academic Publishers.

[6] Sebastian Will. Constraint-based hydrophobic core construction for protein structure prediction in the face-centered-cubic lattice. In *Proceedings of the Pacific Symposium on Biocomputing 2002 (PSB 2002)*, pages 661-672, Singapore, 2002. World Scientific Publishing Co. Pte. Ltd.

[7] Rolf Backofen and Sebastian Will. Fast, constraint-based threading of HP-sequences to hydrophobic cores. In *Proceedings of 7 th International Conference on Principle and Practice of Constraint Programming (CP'2001)*, volume 2239 of Lecture Notes in Computer Science, pages 494-508, Berlin, 2001. Springer-Verlag.

[8] Rolf Backofen and Sebastian Will. Optimally compact finite sphere packings – hydrophobic cores in the FCC. In *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM2001)*, volume 2089 of Lecture Notes in Computer Science, pages 257-272, Berlin, 2001. Springer-Verlag.

[9] Slim Abdennadher, Matthias Saft, and Sebastian Will. Classroom assignment using constraint logic programming. In *Proceedings of the Second International Conference and Exhibition on The Practical Application of Constraint Technologies and Logic Programming (PACLP 2000)*, 2000.

[10] Rolf Backofen, Sebastian Will, and Peter Clote. Algorithmic approach to quantifying the hydrophobic force contribution in protein folding. In *Pacific Symposium on Biocomputing (PSB 2000)*, volume 5, pages 92-103, 2000.

[11] Rolf Backofen, Sebastian Will, and Erich Bornberg-Bauer. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics*, 15 no. 3 pp. 234-242, 1999.

[12] Rolf Backofen and Sebastian Will. Excluding symmetries in constraint-based search. In *Proceedings of 5 th International Conference on Principle and Practice of Constraint Programming (CP'99)*, volume 1713 of Lecture Notes in Computer Science, pages 73-87, Berlin, 1999. Springer-Verlag.

[13] Rolf Backofen, Sebastian Will, and Peter Clote. Algorithmic approach to quantifying the hydrophobic force contribution in protein folding. In *Proceedings of the German Conference on Bioinformatics (GCB'99)*, pages 93-106, 1999. http://www.bioinfo.de/isb/gcb99.

[14] Rolf Backofen and Sebastian Will. Structure prediction in an HP-type lattice with an extended alphabet. In *Proceedings of the German Conference on Bioinformatics (GCB'98)*, 1998.

[15] Rolf Backofen and Sebastian Will. Excluding symmetries in concurrent constraint programming. In *Workshop on Modeling and Computing with Concurrent Constraint Programming*, 1998.

Ehrenwörtliche Erklärung

zur Eröffnung des Promotionsverfahrens

Hiermit erkläre ich,

- dass mir die Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angefertigt und alle benutzten Hilfsmittel, persönlichen Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation nicht schon zuvor als Prüfungsarbeit für eine staatliche unde/oder wissenschaftliche Prüfung eingereicht habe.

Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskriptes haben mich folgende Personen unterstützt:

Prof. Dr. Rolf Backofen

Ich habe die gleiche, eine in wesentlichen Teilen ähnliche bzw. eine andere Abhandlung nicht bereits an einer anderen Hochschule als Dissertation eingereicht.

Jena, den .....

Unterschrift

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die Dissertation selbst angefertigt und alle benutzten Hilfsmittel, persönlichen Mitteilungen und Quellen in meiner Arbeit angegeben habe.

Jena, den .....

Unterschrift