Chapter 1

Bioinformatics and Constraints

Rolf Backofen and David Gilbert

Contents

In this chapter we aim to introduce the topic of bioinformatics to an audience of computer scientists, highlight an illustrative selection of those areas in bioinformatics to which constraint techniques have been applied, and suggest where they may be applicable. Bioinformatics is an exciting and rapidly developing field, and we hope that we haven't predicted all the developments in the next few years – indeed we hope that readers of this chapter will contribute to future applicatiions of constraints in bioinformatics!

One of the first issues that needs to be addressed is the what is meant by "bioinformatics" – it is already almost a colloquial word in the scientific community, but its interpre-

tation varies widely. The word bioinformatics has two obvious components – "bio-" and "informatics"; we deal with each of these in turn.

At present the widely accepted interpretation of the "bio" part is *molecular biology*, i.e. the study of the structure and activity of macromolecules essential to life. However are other areas within biology which can be considered to be within the remit of bioinformatics, for example the study of evolution, and genetics.

Informatics is a word which has only recently entered the English language, following the French, German and Russian traditions which broadly agree that its meaning coincides with "computer science". Thus one definition of informatics is "the science of systematic processing of information, using modeling and abstraction of the concrete realization".

Thus, when considering both parts of the word, we consider the proper meaning to be solving problems arising from biology using methodology from computer science, applied mathematics and statistics. We have decided to focus our contribution on work in bioinformatics that involves either the design of a variant of an existing algorithm from the domain of computer science, or the design of a new algorithm.

An alternative term, more or less coinciding with bioinformatics is *computational biology*, used more in North America than in Europe. Waterman[107] considers that there are three interpretations, all of which are valid:

> One, that it is a subset of biology proper and any required mathematics and computer science can be made up on demand; two, that it is a subset of the mathematical sciences and that biology remains a remote but motivating presence; three: that there are genuine interdisciplinary components, with the original motivation from biology suggesting mathematical problems, which suggest biological experiments.

A good overall introduction to Bioinformatics for computer scientists is [22]. Books that concentrate more on the required mathematical/algorithmic basis of bioinformatics are e.g. [107, 21, 59].

The amount and variety of biological data now available, together with techniques developed so far have enabled research in Bioinformatics to move beyond the study of individual biological components (genes, proteins etc) albeit in a genome-wide context to attempt to study how individual parts cooperate in their operation [60]. Bioinformatics as a scientific activity has now moved closer to the area of Systems Biology [65] which seeks to integrate biological data as an attempt to understand how biological systems function. By studying the relationships and interactions between various parts of a biological system it is hoped that an understandable model of the whole system can be developed. For example the determination that some interaction, and its strength, exists between two entities is a first step to determining network structure and is a crucial step in the modelling and analysis of networks such as gene regulation networks, metabolic pathways and signalling networks. The advent of the new high-throughput technologies (for example gene expression arrays, mass spectrometry) has meant more challenges for computer scientists in terms of the type and quantity of data available for analysis.

There are other fields which broadly apply principles from biology to derive novel approaches in computer science, for example biocomputing, neural computing, genetic algorithms, and evolutionary computing. These are not directly part of Bioinformatics, other than being some of the techniques from computer science which can be applied to biological data.

Since it is rare to find researchers who are both computer scientists and biologists, it is generally the case that effective research in bioinformatics requires the joint effort from scientists in both fields. An important corollary is that in order to achieve such cooperation all parties must use a common language and be prepared to learn about issues from the other side. In fact many researchers from the biological and physical sciences working in bioinformatics have acquired significant computing skills, and may have greater specialist knowledge in mathematics and statistics than do many computer scientists. An illustration of this is the heavy use of Hidden Markov Models in bioinformatics, a topic about which most computer scientists know very little. It is the computer scientist's task to apply the approach of problem abstraction together with efficient algorithm design to the problems from the biological domain.

A challenge for computer scientists who are involved in research in bioinformatics is to achieve results that make a contribution to computer science. Of course this is not the main motivation for biologists; moreover there are some exciting projects in bioinformatics which in the short to medium term are unlikely to contribute to computer science.

1.1 What biologists want from bioinformatics

The great aim of research in bioinformatics is to understand the functioning of living organisms in order to "improve the quality of life". This improvement will be achieved by many means including drug design, identification of genetic risk factors, gene therapy, genetic modification of food crops and animals, etc. Some of these, especially the last, are proving to be controversial.

1.2 The Central Dogma

The study of proteins, how they interact with each other, and how genes are regulated is central to the understanding of the basic principles of the functioning of living organisms.

Proteins comprise approximately 60% of the dry mass of a living cell, and are linear heteropolymers that are constructed from a chain or sequence of monomers called amino acids, of which twenty different types are involved in the composition of proteins. It is widely accepted that the function of proteins (and RNA) is determined by their structure, and it is known that in the majority of cases structure is uniquely determined by the sequence of amino acids, or nucleotides in the case of RNA. The case of *prions* is one example of exception to the latter rule where misfolding causes prion disease [56]. More generally, protein folding can be assisted by molecular chaperones and folding catalysts . Folding catalysts accelerate specific steps in folding, whereas the main function of the molecular chaperones seems to be in preventing off-pathway reactions that lead to protein aggregation and possibly misfoilding. [52]

The central dogma of information flow in biology essentially states that the sequence of amino acids making up a protein and hence its structure (folded state) and thus its function, is determined by a two-stage process. The first stage is *transcription* – the process of copying DNA to RNA by an enzyme called RNA polymerase, and the second is that of *translation* – where messenger RNA is decoded to produce polypeptide chains according to the rules specified by the genetic code. This code enables the 20 amino-acids which form proteins to be coded by triples (codons) of the 4 bases of RNA.

> The central dogma states that once 'information' has passed into a protein it cannot get out again. The transfer of information from nucleic acid to nucleic acid, or from nucleic acid to protein, may be possible, but transfer from protein to protein, or from protein to nucleic acid, is impossible. Information here means the precise determination of sequence, either of bases in the nucleic acid or of amino acid residues in the protein.
>
> Francis Crick [25]

Although some proteins, for example transposases, can modify genetic material by inserting DNA sequences, it is not the case that the amino-acid sequences of those proteins is reverse-coded to make sequences of nucleotides.

Thus bioinformatics is concerned in a major way with the elicitation of DNA sequences from genetic material, the annotation of delimited segments (e.g. with information about their function), the control of gene expression (i.e. under what circumstances proteins are transcribed from DNA), and the relationship between the amino acid sequence of proteins and their structure. At present, the only physical methods to determine protein structure are X-ray crystallography and NMR (nucleo-magnetic resonance), both of which are not only very time-consuming, but cannot be applied to all classes of proteins. One of the holy grails of bioinformatics is to develop computational methods to determine protein structure from amino-acid sequence.

1.3 A classification of problem areas

The problem areas in Bioinformatics can be broadly divided into three classes:

Problems specifically related to the Central Dogma: This includes both those related to a specific level of information (i.e., sequence, structure or function), and those that encompass more than one level.

Problems related to data in general: With the exponential growth of knowledge in (molecular) biology, there are rapidly growing problems such as storage, retrieval, and analysis of the data. Hence there are issues of database design for biological resources, representation and visualization of biological knowledge, and the application of data analysis methods such as data mining. A key underlying technique is that of *abstraction* of the data; it is of course imperative that the operations over the abstract data preserve the biological meaning of the operations on the original form of the data.

Simulation of biological processes: This means the prediction of dynamic behavior of a biological system on the basis of its components. Examples include the simulation of protein folding (molecular dynamics) or of metabolic pathways.

In the following we concentrate on the first class of problems, i.e. sequence, structure and function, and select a subset of illustratory examples.

1.4 Sequence related problems

1.4.1 Physical map

In this problem, one has a collection of short, known substrings of the DNA called *probes* with the property that they occur *exactly once* in the DNA, and a set of fragments of the DNA (called *clones*), which (ideally) cover a specific region of interest on the DNA. For both the clones and the probes, the exact location on the DNA and the ordering of the locations are unknown. The goal is to find the ordering of the probes and/or clones in the DNA.

The first step is to check for every probe P_i and every clone C_j, whether C_j contains the substring denoted by P_i. This is done by performing hybridization experiments. Hybridization is the process of forming a (possibly imperfect) double helix out of two DNA or RNA molecules. This can be used to determine which probes occurs in which clones. The result is a matrix $A = (a_{ij})$, where a_{ij} is 1 if probe P_i is hybridizes with clone C_j, otherwise 0. Now if there were no errors in the hybridization experiments, then the ordering of the probes could be found be reordering the rows and columns of the matrix such that the resulting matrix has the *consecutive ones* property. But since the experiments are faulty, the problem of finding the ordering minimizing the errors is NP-complete (see e.g., [48, 20])

The ordering of the probes (denoted by a permutation π on the set of probe indices), usually together with a good bound on the distance between to successive probes, constitutes a physical map, which can be used for different purposes. One is to use this map when sequencing the genome. The reason is that sequencing is done by splitting DNA into fragments, which are sequenced in the sequel. The remaining problem is to generate the original DNA-sequence out of sequenced fragments. This is usually done by searching for overlapping fragments. The problem is that DNA contains so-called *repeats*. This are long fragments of DNA which are repeated several times on the DNA. Clearly, such repeats may not be used for the process of generating the original DNA sequence out of overlapping fragments. One way to check this is to use a physical map.

In practical applications, the major problem is the occurrence of errors in the hybridization experiments. *False positives* are entries $a_{ij} = 1$ although probe P_i is not contained in clone C_j. Vice versa, *false negatives* are entries $a_{ij} = 0$, where P_i is contained in clone C_j. In [20], Christof et al. considered a variant of the problem that uses additional information stemming from *end-probes*. These are probes where it is known that they are stemming from the end of the clones (but we do not know which is the left or right end). Let P_i and P_k be the end probes for C_r, and let P_j be another probe different from P_i and P_k that hybridizes with C_r (i.e., $a_{jr} = 1$). Then we know that in the correct ordering π, the value π_j must be between π_i and π_k (i.e., either $\pi_i < \pi_j < \pi_k$ is true, or $\pi_j < \pi_j < \pi_i$), which gives rise to additional *betweenness* constraints. They presented an integer linear programming approach for the above described problem, where a maximum likelihood model is used as an objective function to model the errors in the matrix A. The idea behind the maximum likelihood model is to search for the corrected matrix B that maximizes the likelihood $P[A|B]$, given probabilities for producing false positive and false negative entries in A.

1.4.2 Comparison and alignment

Overview

The goal of this activity is to compare two sequences, and in addition to return an alignment, i.e. some information regarding those parts which are very similar. When comparing the sequences, additional information e.g. stemming from known structures may be used. In general, sequence alignment is fast, whereas alignment involving structure is slow due to its high complexity.

One of the first fields in bioinformatics was DNA sequence alignment. The reason for the interest in sequence alignment stems from the fact that there are many different proteins which have common ancestors, and that these *homologous* (i.e., related by evolution) proteins have a similar structure and function. In addition, homologous proteins often have similar sequences. Using a reverse reasoning, sequence similarity is used to detect the homology of protein structures.

Clearly, the quality of this approach depends on the similarity measure used, which is determined by a model of evolution. The usual approaches use a model with substitution, deletion or insertion of a single amino-acid (see e.g. [107] for an overview). In this case, sequence alignment can be performed in polynomial time using a dynamic programming approach. There are also new approaches which deal with more complex models of evolution such as [10], who considers in addition duplication of substrings (tandem repeats). A more complex problem is that of multiple sequence alignment [64], which is known to be NP-complete.

On the level of structure comparison, there are many different problems that have been considered. Protein threading extends sequence alignment by incorporating structural information. In this approach an alignment is made between two sequences, one with an unknown structure and the other with a known structure, taking into account the known structure [69]. Again, this problem has been shown to be NP-hard.

Another problem is to compare two different structures by superposing elements using translation and rotation to minimize the atomic coordinate Root Mean Square Deviation (RMSD) [35]. Structures can also be compared at a higher level of abstraction than atomic coordinates by using a topological approach based on secondary structure elements [47] (see Section 1.5.4; this can be performed over topology graphs by detecting maximal cliques [66] or by pattern discovery and structural alignment [44]).

Pairwise sequence alignment

Pairwise sequence alignment is the problem of determining the similarity of two sequences. An alignment of two sequences $a, b \in \Sigma^*$ consists of two sequences u, v of the same length that are generated from a, b via the insertion of gaps. Alignments are evaluated according to scoring functions, which evaluates the number of inserted gaps, and the similarity of different letters u_i and v_i at the same position in the alignment (called substitutions). Multiple sequence alignment is the generalization of the problem to several sequences.

There are different possibilities for constraint-based formalizations of sequence alignment. We will start with a formalization that is commonly used in standard approaches to sequence alignment. We will start with the formal definition of an alignment.

Definition 1 (Alignment and Alignment Distance). *Let Σ be an alphabet with $- \notin \Sigma$. For every $u \in (\Sigma \cup \{-\})^*$ we define $u|_\Sigma$ to be the restriction of u to Σ (by deleting all occurrences of $-$ in u). An* alignment *is a pair $(a^\diamond, b^\diamond)$ with $a^\diamond, b^\diamond \in (\Sigma \cup \{-\})^*$ such that $|a^\diamond| = |b^\diamond|$ and there is no position i such that $a_i^\diamond = - = b_i^\diamond$. An alignment $(a^\diamond, b^\diamond)$ is* an alignment of (a, b) *with $a, b \in \Sigma^*$ if*

1. $a^\diamond|_\Sigma = a$, *and*
2. $b^\diamond|_\Sigma = b$.

Given a cost function w, we define the cost of an alignment by

$$w(a^\diamond, b^\diamond) = \sum_{i=1}^{|a^\diamond|} w(a_i^\diamond, b_i^\diamond).$$

The alignment distance *of a, b is*

$$D(a, b) \quad = \quad \min\{w(a^\diamond, b^\diamond) \mid (a^\diamond, b^\diamond) \text{ alignment of } (a, b)\}.$$

The alignment $(a^\diamond, b^\diamond)$ is optimal *if $D(a, b) = w(a^\diamond, b^\diamond)$.*

Instead of using distance-based scoring function, one can also use a similarity measurement for evaluating alignments. Then, one searches for an alignment that maximizes the similarity between the two sequences. As [93] have shown, one can transform each distance-based (global) scoring scheme into a similarity-based, without changing the optimal alignment. Hence, we will consider only the distance-based scoring scheme in the following.

The standard approach to solve the pairwise sequence alignment problem for two sequences a, b is to use to define a dynamic programming matrix $(D_{i,j})$, which stores the cost of the best alignment between the prefixes $a_1 \dots a_i$ and $b_1 \dots b_j$. I.e., $D_{i,j} = D(a_1 \dots a_i, b_1 \dots b_j)$. This matrix can then be calculated using the following recursion equation:

$$\begin{aligned} D_{0,0} &= 0, \\ D_{0,j} &= \sum_{k=1}^{j} w(-, b_k), \\ D_{i,0} &= \sum_{k=1}^{i} w(a_k, -), \\ \forall i, j > 0 : D_{i,j} &= \min \left\{ \begin{array}{l} D_{i,j-1} + w(-, b_j), \\ D_{i-1,j-1} + w(a_i, b_j), \\ D_{i-1,j} + w(a_i, -) \end{array} \right\}. \end{aligned} \tag{1.1}$$

Thus, the standard sequence alignment problem can be solved in quadratic time and space. This changes if one considers different extensions of the original problem. One extension is to consider *parametric sequence alignment*, where the cost parameter for deletion $\delta = w(\sigma, -)$ and substitution $\mu = w(\sigma, \sigma')$ are variable. The reason for considering this parametric version is that it is hard to determine these parameter (especially the cost

for deletion). Hence, one is interested in checking whether a given alignment is the same for a complete range of parameters. Yap [109] considered a constraint-based approach for this problem, where he directly encodes the entries of the dynamic programming table $D_{i,j}$ as variables, and the recursion equations as constraints. He considered then different possibilities of pruning in the case that δ and μ are not known (i.e., are not ground).

Other variants extend sequence alignment by considering additional conditions that stem from information on the secondary or ternary structure of the associated molecule. By and large one can say that the difference to sequence alignment is that the scoring function evaluates not single positions in the alignment, but pairs of positions that are related (or close) in the structure. This is especially useful when comparing two RNA-sequences, where it is known that the structure is more conserved than the sequence. Both global [86, 23, 73, 57, 55] and local [6] versions of the RNA sequence/structure alignment have been considered. The multiple RNA sequence/structure alignment problem is even harder than the multiple sequence alignment problem, since successful heuristic approaches like progressive alignment can only be applied either in special cases (like the PMMulti system [55]), or via the combination of sequence/structure and sequence alignment (like the MARNA-system [92]).

There many are other problems that extend sequence alignment (or related problems) using additional information. Examples are the alignment methods used for the detection of alternative spliceforms of proteins [41, 49, 54, 39], or the design of similar protein sequence whose mRNA form a specific RNA-structure [3]. Currently, most of these problems are solved via special dynamic programming approaches. A first approach to apply a general technique for sequence alignment under additional constraints has been presented in [**?**], where cluster tree elimination was used to efficiently solve pairwise sequence alignment problems with additional constraints.

Multiple sequence alignment

The problem of multiple sequence alignment is to align not only two different sequences, but any number of sequence. This is required to detect biologically important motifs. Formally, a *multiple alignment* for n sequences $S_1, \ldots, S_n$ is given by a character matrix

$$A = (A_{ij})_{1 \leq i \leq n, 1 \leq j \leq K}$$

over the alphabet $\Sigma = \Sigma \cup \{\text{-}\}$ with the property that S_i can be obtained from $A_{i1} \ldots A_{iK}$ by removing the gaps. In the general formalization, the jth column $A_{1j}, \ldots, A_{nj}$ of the alignment is evaluated using an n-ary function $w(A_{1j}, \ldots, A_{nj})$, and the distance $D(A)$ of an alignment A is given by

$$D(A) = \sum_{1 \leq j \leq K} w(A_{1j}, \ldots, A_{nj}).$$

There is a special formalization of the scoring function that is used in most practical applications, namely the sum-of-pairs score. The basic idea of this score is to evaluate an alignment by the sum of all pairwise alignments, which was introduced by Carrillo and Lipman [17]. Here, the distance of an alignment $D(A)$ is given by given by

$$D(A) = \sum_{i < i'} \sum_{1 \leq j \leq K} w_p(A_{ij}, A_{i'j}),$$

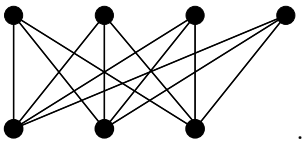
where $w_p : \Sigma' \times \Sigma' \to \mathbb{R}$ is a usual pairwise cost function. Of course, this is equivalent to

$$D(A) = \sum_{1 \le j \le K} \sum_{i < i'} w_p(A_{ij}, A_{i'j}),$$

and is hence a special case of the general multiple sequence alignment problem, where the cost for a column is given by

$$w(a_1, \ldots, a_n) = \sum_{i < i'} w_p(A_{ij}, A_{i'j})$$

Kececioglu [63, 64] introduced a graph-based formalization of multiple sequence alignment with sum-of-pairs cost function, the *complete maximum-weight trace (CMWT)* formalization. An ILP (integer linear programming) solution for this problem was presented in [84, 62]. In CMWT, the letters of the strings $S_i = s_{i1} \ldots s_{in_i}$ are considered to be the set of vertices $V = V_1 \uplus \ldots \uplus V_n$[1] of a complete n-partite graph $G = (V, E)$ (i.e., G satisfies that for every $s_{ij} \in V_i$ and $s_{i'j'} \in V_{i'}$, we have $e = (s_{ij}, s_{i'j'}) \in E$ if and only if $i \neq i'$). G is called the *complete alignment graph* for the sequences $S_1, \ldots, S_n$. An *alignment graph* G' is a subgraph of the complete alignment graph. Alignment graphs can be used to restrict the search for a multiple sequence alignment to a subset of all possible alignments to reduce the search space. For example, let S_1 be AACG and S_2 be AGG. Then the complete alignment graph for AACG and AGG is the 2-partite graph



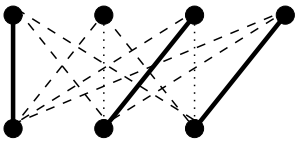
.

With every edge $e \in E$, there is a positive weight $w(e)$ associated. An alignment A for the sequences $S_1, \ldots, S_n$ *realizes* an edge $e = (s_{ij}, s_{i'j'}) \in E$ of an alignment graph $G = (V, E)$ for the sequences $S_1, \ldots, S_n$ if the jth character of S_i and the j'th character of $S_{i'}$ are aligned in A. For example, consider the alignment

```
A  A  C  G
A  -  G  G
```

.

Then this alignment realizes three edges, indicated by straight lines:

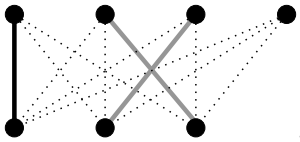


Given an alignment A, the set of all edges realized by A is called the *trace* of A. A set $T \subset E$ of edges is called a *trace* if it is the trace of some alignment A. Given the weight function w, the *weight* of a trace T is $\sum_{e \in T} w(e)$.

[1] Where $\uplus$ is the disjoint union, and V_i is $\{s_{i1}, \ldots, s_{in_i}\}$.

Definition 2 ((Complete) Maximum-Weight Trace). *Let $S_1, \ldots, S_n$ be sequences, let $G = (V, E)$ be the complete alignment graph for $S_1, \ldots, S_n$, and let w be a weight function. The* complete maximum-weight trace problem *is to find a trace $T \subset E$ that has maximal weight (under w). The* maximum-weight trace problem *is defined analogously for an alignment graph $G = (V, E)$ for $S_1, \ldots, S_n$.*

A remaining problem is that not any subset of edges is a trace (i.e., not every subset of E corresponds to a real alignment). Consider again the two sequences AACG and AGG, and consider the following subset of edges indicated by straight lines:



By the definition of a realized edge, this set of edges would correspond to the alignment

```
A  C  A  G
A  G  G  -
```

which is an alignment for the sequences ACAG and AGG instead of AACG and AGG. Hence, this subset of edges is not a trace. The problem are the two crossing edges indicated in grey above.

An ILP-formalization for the pairwise alignment characterizing traces was given in [73][2], which is a follows. Let $G = (V, E)$ be an alignment graph, and let $e_1, \ldots, e_n$ be an enumeration of all alignment edges in E. We say that e_k is *in conflict with* e_l iff e_k and e_l are crossing edges, i.e., $e_k = (s_{1i}, s_{2j})$, $e_l = (s_{1i'}, s_{2j'})$ with neither $i < i' \wedge j < j'$ nor $i' < i \wedge j' < j$. Then one introduces for every edge e_i a boolean variable x_i, where $x_j = 1$ implies that e_i is contained in the trace. Furthermore, let $w_i = w(e_i)$. Then the constraint problem is

$$\text{maximize} \quad \sum_{e_i \in E} w_i \cdot x_i$$

subject to the following constraints:

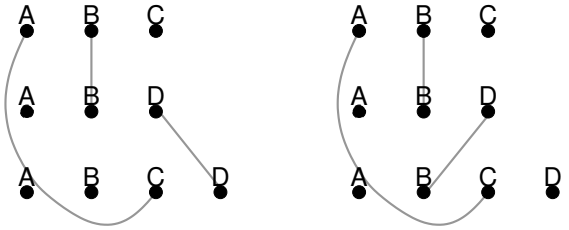
$$x_i \in \{0, 1\} \tag{1.2}$$

$$x_k + x_l \leq 1 \qquad \forall e_k, e_l \in E \text{ s.t. } e_k \text{ is in conflict with } e_l \tag{1.3}$$

For the multiple sequence alignment step, the condition of non-crossing edges is not so simple. Whether a pair of edges is conflicting might depend on other edges contained in the trace. Consider the following two set of edges for three sequences ABC, ABD and

[2] In this work, structural condition where formulated in addition to the pure sequence alignment problem

$ABCD$:



The the first represents for example the following valid alignment:

```
- - A - B C -
- - - A B - D
A B C - - - D
```

.

The second one does not represent a valid alignment, but we cannot identify pairs of conflicting edges.

Hence, we have to extend the definition for the multiple case. For the pairwise case, a trace is nothing else than a set of edges which are strictly ordered *in both components*. I.e., a trace is an ordered set of edges $e_1 = (s_{1i_1}, s_{2j_1}), \ldots, e_m = (s_{1i_m}, s_{2j_m})$ with the property that $\forall 1 \leq k < m : i_k < i_{k+1} \wedge j_k < j_{k+1}$. The corresponding definition for the multiple alignment case is as follows.

Given sequences $S_1, \ldots, S_n$ with $S_i = s_{i1} \ldots s_{in_i}$, one defines the *extended alignment graph* $G = (V, E, \prec)$ for $S_1, \ldots, S_n$ to be a triple such that (V, E) is an alignment graph for $S_1, \ldots, S_n$, and $\prec$ is defined by

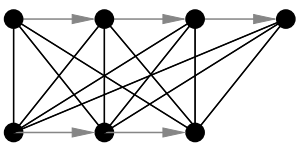
$$\prec = \{(s_{ij}, s_{ij+1}) \mid 1 \leq i \leq n \wedge 1 \leq j < n_i\}.$$

With $\prec^*$, we denote the transitive closure of $\prec$. Note that $\prec^*$ is a strict partial order of V.

Using the extended alignment graph, one can characterize traces. A *connected component* of a graph $G = (V, E)$ is a $\subseteq$-maximal set $V' \subseteq V$ such that for all vertices $v, v' \in V'$ there is a path of edges in E connecting v and v'. For any two subsets $X, Y \subseteq V$, we define

$$X \lhd Y \text{ if and only if } \quad \exists v \in X \; \exists v' \in Y : \; v \prec v'.$$

We define $\lhd^*$ to be the transitive closure of $\lhd$. For the sequences `AACG` and `AGG` the extended complete alignment graph is



where we have indicated the edges for $\prec$ by arrows.

Theorem 3. *Let $S_1, \ldots, S_n$ be sequences, and let $G = (V, E, \prec)$ be the extended alignment graph for $S_1, \ldots, S_n$. Then a subset $T \subseteq E$ is a trace if and only if it does not contain two edges sharing the same node, and $\lhd^*$ is a strict partial order on the connected components of $G' = (V, T)$.*

The question is of course how to enforce the above stated condition in a constraint-based or ILP formalization. For the pairwise case, this is achieved by excluding all conflicting edges with the constraint given in (1.3), thus forcing a strict partial order on the edges. Following [62], then every pair of conflicting edges for the pairwise case corresponds in the extended alignment graph to a *mixed cycle*. This is a cycle in the extended alignment graph that uses at least one alignment edge and at least one edge from the $\prec$-order. Such a mixed cycle is called *critical* if in every sequence, all the nodes used by the cycle occur consecutively in the sequence. Then condition (1.3) is replaced in [62] by excluding all critical mixed cycles, which then gives an ILP-formalization for multiple alignment.

The main problem with the above formulation is that in general, one has to add exponentially many cycle constraints. For this reason, Prestwich et al. proposed in [83] an alternative ILP model, which is transformed to linear pseudo-Boolean (PB, a generalization of SAT which significantly improves expressiveness) form. The model is of polynomial size, and therefore better suited to a generic SAT solver.

Pairwise alignment with conditions: example protein threading

The previous formalization is based on a graph based model of sequence alignment, where one has Boolean variables for every possible alignment edge. The major drawback of this approach is that it uses a huge number of variables. E.g., for pairwise sequence alignment, this model requires quadratically many variables.

Another possible formalization for pairwise sequence alignment that requires less variables has one variable X_i for each position $1 \leq i \leq |S_1|$ of the first sequence S_1. The domain of each variable is the set $\{1, \ldots, |S_2|\}$ of positions in the second sequence. In principle, $X_i = j$ is interpreted as "*position i of the first sequence is aligned with position j of the second one*".

The next step is to encode gaps. An unaligned position j in the second sequence, which correspond to a gap in the first one, are already encoded by the fact that there is no i with $X_i = j$. In addition, one has to encode that a position i in the first sequence is aligned with a gap in the second sequence. One possible way to encode this is by allowing $X_{i-1} = X_i$, which is then interpreted as position i is aligned with a gap. On the other hand, position i is matched (i.e., aligned with some position j in the second sequence) if and only if $X_i = j$ *and* $X_i > X_{i-1}$. Note that this kind of encoding was considered in [**?**].

In the following, we will consider a special instance of pairwise sequence alignment with additional conditions using a formalization similar to the one described above, namely protein threading. The additional conditions stem from information about the structure of one sequence. For protein threading, we have a sequence s with known structure, and we want to determine the structure of a sequence s' that is homologous (i.e. related via evolution) to s via an appropriate pairwise alignment. The idea is to use the known structure of s to guide structure prediction for s' by simultaneously aligning s' with s and with the known structure of s.

The basic approach for protein threading is to identify first parts of the structure of s that are more likely to be conserved. This is called a *core model for* s, and consists usually of secondary structure elements. The *secondary structure* of a sequence consist of structural elements of high local order. There are two main elements considered for protein threading, namely α-helices (a helical structure) and β-sheets (two or more strands

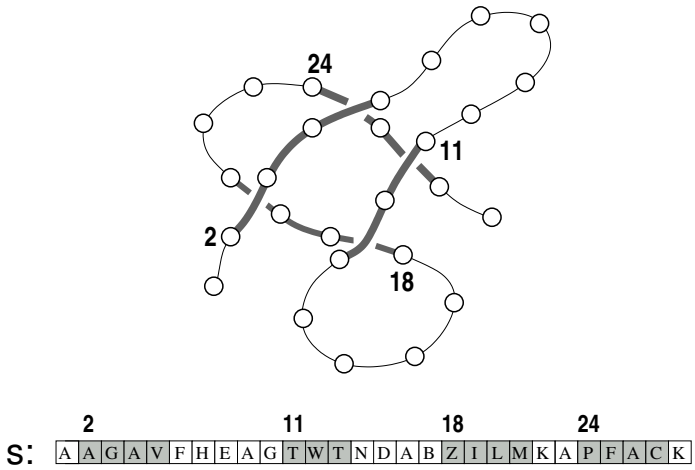


Figure 1.1: Core model. It is supposed that the grey parts of the given structure are the conserved regions. The define the core of the structure. This leads to the definition of 4 core elements.

of the protein sequence that are regularly connected). It is assumed that the core models are highly conserved in their length as well as in their interactions. The stretches between two core elements are called *loops*, and the lengths of these loops can vary in the homologous sequence s'. This is captured by the definition of a core model.

Definition 4 (Core Model). *Let s be a sequence. A* core model *for s is a tuple $(m, \vec{c}, \vec{\lambda}, \vec{l}_{\min})$, where $\vec{c} = (c_1, \ldots, c_m)$ is the sequence of lengths for the core elements in s, and $\vec{\lambda} = (\lambda_0, \ldots, \lambda_m)$ is the sequence of lengths for the loops between the core elements such that*

$$|s| = \lambda_0 + \sum_{1 \le i \le m} (c_i + \lambda_i).$$

The sequence $\vec{l}_{\min} = (l_0^{\min}, \ldots, l_m^{\min})$, consists of the minimal length required to connect the corresponding ends of the core elements (i.e., the minimal length of the loop regions) with $\forall 1 \le i \le m : \; l_i^{\min} \le \lambda_i$.

Note that the value λ_0 is the length of the initial loop (i.e., the N-terminal loop), while λ_m is the length of the final loop (i.e., the C-terminal loop).

Given a core model $(m, \vec{c}, \vec{\lambda}, \vec{l}_{\min})$ for s, we define the ith *core region* of s to be the set of positions

$$C_i = \left\{ \lambda_0 + \sum_{1 \le j < i} (c_j + \lambda_j) + k \;\middle|\; 1 \le k \le c_i \right\}.$$

The jth position of the ith core is denoted by $C_{i,j}$. Figure 1.1 illustrates a core model with 4 core regions, where the lengths of the core regions is given by the vector $(4, 3, 4, 3)$.

In the following, we will define a threading of sequence s' through the core model for s to be a mapping of the core positions to consecutive positions of s'. Since we are using consecutive regions, a threading is uniquely determined by the mapping of the first position of every core region. Furthermore, this implies that there are no gaps allowed in core regions. All gaps in the alignment must occur in the loop regions. In inserting and deleting positions in the loop regions, one must obey the length restrictions imposed by the core model. Recall here that $l_i^{\min}$ is the minimal length needed to connect C_i and C_{i+1} according to stereochemical restrictions (depending, e.g., on the distance between the last position in C_i and the first position of C_{i+1} in the structural model of s).

Definition 5 (Threading). *Let $\mathbb{M}_C = (m, \vec{c}, \vec{\lambda}, \vec{l}_{\min})$ be a core model for a sequence s. Let s' be a sequence. A* threading of s' through the core model $\mathbb{M}_C$ for s *is a vector*

$$\vec{t} = (t_1, \ldots, t_m) \in \mathbb{N}^m$$

such that

$$1 + l_0^{\min} \leq t_1 \tag{1.4}$$

$$\forall 1 \leq i < m : (t_i + c_i + l_i^{\min}) \leq t_{i+1} \tag{1.5}$$

and

$$t_m + c_m + l_m^{\min} \leq |s'| + 1 \tag{1.6}$$

In the following, we set $c_0 = 0$ for convenience. The conditions (1.4)–(1.6) are called *ordering constraints.* These constraints imply so-called *spacing constraints*, which constitute a domain for the ith value of an arbitrary threading

$$\forall 1 \leq i \leq m : \left[1 + \sum_{j<i} (c_j + l_j^{\min}) \leq t_i \leq |s'| + 1 - \sum_{j \geq i} (c_j + l_j^{\min}) \right] \tag{1.7}$$

The next part is to score the threading. The first step is to define the interactions that are determined by the given structure. Thus, the *interaction graph* describes which core regions contain core positions that are 'neighbors' in some biochemical sense, i.e. that are core positions that interact in the folded structure. Albeit the interactions have to be defined on the level of amino acids, one usually combines all interactions between two core regions into one single interaction. How this complex interaction is evaluated is then hidden in the scoring function.

Definition 6 (Interaction Graph). *Let s be a sequence with a core model $\mathbb{M}_C = (m, \vec{c}, \vec{\lambda}, \vec{l}_{\min})$. An* interaction graph $\mathbb{I}$ *for $\mathbb{M}_C$ is a graph (V, E), where $E \subseteq V^2$ and V is the set of all core regions, i.e.*

$$V = \{C_i \mid 1 \leq i \leq m\}.$$

Definition 7 (Scoring Function). *Let s be a sequence with core model $\mathbb{M}_C = (m, \vec{c}, \vec{\lambda}, \vec{l}_{\min})$ and interaction graph $\mathbb{I}$. A scoring function g for s, $\mathbb{M}_C$, and $\mathbb{I}$ consists of two functions $g_1 \in \mathbb{N}^2$ and $g_2 \in \mathbb{N}^4$ with the property that*

$$g_2(i, j, k, l) \neq 0 \Leftrightarrow (C_i, C_j) \in \mathbb{I}. \tag{1.8}$$

Given a threading $\vec{t}$ of s' to s under the core model $\mathbb{M}_C$, the score *of $f(\vec{t})$ of $\vec{t}$ is defined by*

$$f(\vec{t}) = \sum_{i=1}^{m} g_1(i, t_i) + \sum_{i=1}^{m} \sum_{j>i}^{m} g_2(i, j, t_i, t_j).$$

This is the form of scoring function that is most often used in protein threading, where only pairwise interactions are considered. But in general, higher-order interactions could be admitted. To include these, one must extend the definition of interaction graph to that of an interaction hypergraph. Furthermore, one must introduce $2n$-ary functions g_n in order to implement n-ary interactions. If the core model has m regions, then $n \leq m$. Hence, the fully general form of scoring function is

$$\begin{aligned} f(\vec{t}) \quad = \quad & \sum_{i_1} g_1(i_1, t_{i_1}) + \sum_{i_1} \sum_{i_2 > i_1} g_2(i_1, i_2, t_{i_1}, t_{i_2}) + \dots \\ & + \sum_{i_1} \sum_{i_2 > i_1} \dots \sum_{i_{m-1} > i_m} g_m(i_1, i_2, \dots, i_m, t_{i_1}, t_{i_2}, \dots, t_{i_m}). \end{aligned}$$

Lathrop and Smith [69] solved the threading problem using a branch-and-bound approach, working on sets of possible threadings $\mathbb{T}$, which are described by finite domains for the t_i's. Reformulating the scoring function slightly, we get the following tight bound using the scoring function itself:

$$\text{lb}(\mathbb{T}) = \min_{\vec{t} \in \mathbb{T}} \sum_{i=1}^{m} \left[g_1(i, t_i) + g_2(i-1, i, t_{i-1}, t_i) + \sum_{|i-j|>1}^{m} \frac{1}{2} g_2(i, j, t_i, t_j) \right].$$

Of course, it is NP-hard to calculate $\text{lb}(\mathbb{T})$. For that reason, they introduced the following following relaxed scoring function for a set of threadings $\mathbb{T}$:

$$\text{lb}_{\text{poly}}(\mathbb{T}) \quad = \quad \min_{\vec{t} \in \mathbb{T}} \sum_{i=1}^{m} \left[\begin{array}{ll} & g_1(i, t_i) \\ + & g_2(i-1, i, t_{i-1}, t_i) \\ + & \min\limits_{\substack{\vec{u} \in \mathbb{T} \\ u_i = t_i}} \sum\limits_{|i-j|>1} \frac{1}{2} g_2(i, j, t_i, u_j) \end{array} \right],$$

The relaxation is given by the fact that for the calculation of g_2, for every i a different threading $\vec{u}$ can be used. Thus, there is no dependencies anymore for the calculation of the g_2 terms (with the exceptions of the terms $g_2(i-1, i, t_{i-1}, t_i)$), which implies that the bound can be calculated in polynomial time using dynamic programming.

1.4.3 Search and pattern discovery

In both sequences (DNA and RNA) and structure (RNA and protein), there are functionally significant regions that are repeated in different entities; these regions can be often described by patterns. A need has arisen to be able to search through genome or protein databases (which may be very large), and identify entries which match the pattern. Obviously, this has a parallel in formal language theory, see for example Searls' excellent discussion in [87]. In reality, biological data is noisy, and in the case of string languages, stochastic approaches have been developed using for example Hidden Markov Models [33]

and stochastic context-free grammars [71]. It is of interest to note here that although Dynamic Bayesian Networks [43] can represent Hidden Markov Models, the use of DBNs in bioinformatics for sequence analysis remains an under-exploited area.

Although patterns can be constructed by hand, its preferable to use a mechanized (machine learning) approach, i.e., *pattern discovery* [15, 85]. Finding gene expression sites in DNA may require context sensitive patterns.

One active research field is to design appropriate pattern languages and associated discovery mechanisms which are able to express significant properties of structures as opposed to strings [47, 58].

Pattern discovery can also be performed over protein structures [46] and metabolic pathways.

Sequence pattern matching

The basic biochemical properties of DNA and RNA permit some constraints to be exploited in pattern matching over nucleotide sequences:

- The first property is that of the total ordering of the nucleotides in a sequence, by convention from the $5'$ to the $3'$ end, which can be exploited in pattern matching algorithms.

- The second property is the name associated with a nucleotide. A DNA nucleotide consists of a base – adenine, cytosine guanine, or thymine — plus a molecule of sugar and one of phosphoric acid; such nucleotides are often known by the initial letter of the base that they contain, a, c, g or t. In the case of RNA, thymine is replaced with uracil (u). Thus the names of nucleotides are drawn from a restricted alphabet of size 4: a, c, g, t in the case of DNA, and a, c, g, u in the case of RNA, and patterns can be defined with characters drawn from (a subset of) the alphabet.

- Thirdly, two nucleotides can interact due to the Watson-Crick base pairs: in the case of DNA, a-t and c-g, with both pairs being of roughly equal strength. RNA pairs are a-u, c-g, as well as the weaker g-u, and some other even weaker pairs. This base paring can cause nucleotide sequences to adopt particular conformations due to long-range interactions. This pairing can be exploited both in formal models of the conformations, and also associated techniques to compute over these models.

Protein sequences comprise amino-acids which have properties corresponding to the first two above: firstly that they are totally ordered (in this case from the N terminus to the C terminus), and secondly that there are 20 amino acids, i.e. the names are drawn from an alphabet of 20 names (or corresponding letters).

Eidhammer et al. [34] have defined patterns in sequences as consisting of a logical expression on components, where a component is a description of a string of symbols, and a set of constraints. An input string S matches a pattern P if every component in P is matched by some substring of S, such that all the constraints are satisfied and the logical expression evaluates to *True*.

Sequential patterns can be defined using the following constraints:

(1) *length* of a substring to match a specific component;

Sequential Patterns		
Tandem repeat	$\alpha\alpha$	$\underline{acg}\,\underline{acg}$
Simple repeat	$\alpha\beta\alpha$	$\underline{acg}\,\underline{aaa}\,\underline{acg}$
Multiple repeat	$\alpha\beta\alpha\beta_1\alpha$	$\underline{acg}\,\underline{aa}\,\underline{acg}\,\underline{uu}\,\underline{acg}$
Structural Patterns		
Stem loop	$\alpha\beta\alpha^{rc}$	$\underline{acg}\,\underline{aa}\,\underline{cgu}$
Attenuator	$\alpha\beta\alpha^{rc}\beta_1\alpha$	$\underline{acg}\,\underline{aa}\,\underline{cgu}\,\underline{au}\,\underline{acg}$
Palindrome, even	$\alpha\alpha^{r}$	$\underline{acg}\,\underline{gca}$
Palindrome, odd	$\alpha x\alpha^{r}$	$\underline{acg}a\underline{gca}$
Pseudoknot	$\alpha_1\beta\alpha_2\beta_1\ \alpha_1^{rc}\beta_2\alpha_2^{rc}$	$\underline{acg}\,\underline{aa}\,\underline{ucu}\,\underline{gc}\,\underline{cgu}\,\underline{aua}\,\underline{aga}$

Table 1.1: Patterns in nucleotide sequences, from [34]

(2) *distance* in the input string between substrings to match the different components of a pattern;

(3) *contents* of a substring to match a component;

(4) *positions* on the input string where a particular component can match;

The patterns in the PROSITE data base [9] are examples of the sequential class; thus *[AC]-x(2,3)-D* describes a pattern comprising three components, the first being an *A* or a *C*, the second of length 2 or 3 and the last being a *D*.

Structural patterns have in addition at least one *correlation constraint*, between two substrings matching different components, e.g. the substrings should be identical, or the reverse of each other. Examples are repetitions or palindromes, and can correspond to conformations that the sequence can adopt.

Example patterns in nucleotide sequences identified from the literature by Eidhammer et al [34] are given in Table 1.1 below. Pattern components (strings) are indicated by letters from the Greek alphabet: $\alpha, \beta, \ldots$ (with or without indices) and x is a wildcard. The reverse of a component α is indicated by α^r, and α^c is the complement of α. These annotations can be combined: α^{rc} is the reverse complement of α. Strings corresponding to pattern components are underlined.

The CLP version of the Eidhammer et al. system is now no longer available for general use. from the paper describing it. A related but more sophisticated, and faster approach by Thebault e al. is described in [100]. They use a CSP approach, representing structured RNA motifs which interact with other molecules. These motifs occur on more than one sequence and which are related together by possible hybridization. Together with pattern matching algorithms, constraint satisfaction techniques have been implemented in a prototype software system called "MilPat" (http://carlit.toulouse.inra.fr/MilPaT/MilPat.pl) and can be applied to search for tRNA and snoRNA genes on genomic sequences.

Another related approach using CSP is by Morgante et al [82]; the software system SMaRTFinder can be downloaded from http://bioinf.dimi.uniud.it/software/software/smartfinder. The algorithm locates structured models which are sequences of simple motifs and distance constraints. It combines standard pattern matching procedures with a constraint satisfaction solver, and can search for partial matches. A significant feature of their approach is that the (potentially) exponentially many solutions are represented in compact form as a

graph. The time and space necessary to build the graph are linear in the number of occurrences of the component patterns.

Staden's program [97] is an early system which permits search for structural patterns in sequences. A pattern comprises elements, called motifs. There are nine classes of motifs, the simplest of which is just a string of characters. Two other classes include structures: inverted repeat or stem-loop and (direct) repeat. Logical operators AND, OR and NOT can be used to specify whether each motif must be present, is an alternative to another, or must be absent. Constraints can be specified on the length of a motif, the distance between two motifs and the contents of a motif; for the structure classes, constraints can be given on an individual part of the structure, e.g. on the loop of a stem-loop. Percentage match and scoring matrices can be used in searching. In Staden's system there is no possibility to define *general* correlations or relations between parts.

An example of a pattern which can be decribed in Staden's language is

$$tata(\langle(at \text{ OR } cg), -5, -2\rangle \text{ AND } \langle tt(\langle\neg ga, -3, 3\rangle), 2, 6\rangle)$$

which describes a pattern whose 'root' motif is the string $tata$, with two further required motifs. The first of these is between 5 and 2 bases upstream of the $tata$ motif, and can be either at or cg. The second is a tt motif located between 2 and 6 bases downstream of the $tata$ motif, and there must not be a ga motif within 3 bases upstream or downstream of the tt.

As can be seen, the language permits motifs to be overlaid on each other. Although this may seem counter-intuitive when describing biochemical sequences, there are situations when such overlays occur during processing of nucleotides, for example 'cassette genes' [50].

Other languages and associated systems are: *SCRUTINEER* [90], *RNAmot* [40], *RNAMotif* [75] (http://www.scripps.edu/mb/case/casegr-sh-3.5.html), *OVERSEER* [91], *Palingol* [12], *PatScan* [32] (http://www-unix.mcs.anl.gov/compbio/PatScan/HTML/patscan.html), and *PALM* [53]. However *GENLANG* (Searls [89, 88]) is the most general system which has been implemented for searching for structural patterns in nucleotide sequences. It uses an indexed language which has an expressive power between context-free and context-sensitive languages. *String variables* are used to define structures and constraints on the length and contents of the string variables can be specified.

Eidhammer et al. [34] have defined a constraint-based structure description language for biosequences, and give an algorithm plus associated program to solve the structure searching problem as a CSP as well as an implementation in the constraint logic programming language clp(FD) [27]. The language is able to describe two-dimensional structure of biosequences, such as tandem repeats, stem loops, palindromes and pseudo-knots.

1.4.4 Phylogenetic trees

If we have any set of species that are related, then the relationship between these species (resp. entities) is called a *phylogeny*. When constructing a phylogenetic tree, the task is to set up a tree to show how the different species have evolved from a common ancestor. In addition, the trees generated are often labelled. The labels indicate the time when the species evolved from a common ancestor, or any other measure of the distance between the different species. Note that the construction phylogenetic trees is not necessarily applied to

species, but to any kind of entities where we can set up some sort of distance information (e.g., phylogenetic trees can be constructed for languages). In this case the tree constructed may not be rooted.

The problem of constructing phylogenetic trees can be formulated in different ways. The first one is to have a finite set of species or entities $S = \{e_1, \ldots, e_n\}$, and a distance matrix $(d_{ij})_{i,j\in[1..n]}$ containing the pairwise distances between the entities. The problem is to construct a tree, where the edge are labelled by distances and the nodes are labelled entities (using new entities for the inner nodes). The tree is correct if for each two entities e_j, e_k from S, the distance in the tree (by summing up the edges distances along the path connecting them) out of the ordinal set in the tree is d_jk. Trees can be constructed from pairwise distances by variety of methods, including UPGMA (unweighted pair group method using arithmetic averages) [96].

Another formulation of the phylogenetic tree construction problem is *parsimony* [38]. Here, one has a set S of sequences (DNA or protein), and a method for calculating costs for relating any two sequences (not restricted to S). The task is then to find a tree, where the leafs are labelled by elements of S and the inner nodes are labelled by other sequences. Furthermore, the tree should have minimal costs according to the given method (i.e., the sum of distances between any two sequences that are directly connected in the tree should be minimal).

Since one, or in the case of parsimony several, optimal trees can be generated by tree building algorithms, an approach such as the bootstrap method [37] is commonly used to assess the significance of some phylogenetic feature and thus give some measure of confidence for the tree.

Although the concept of 'constraints' is widely used in the phylogentic literature, for example in the application to parsimony and maximum likelihood in terms of constraints over edge parameters between substitution sites, [98], almost no work has been done by the computational constraint community. However, related work certainly exists, for example the work by Gent et al [42] on the application of constraint programming to supertrees.

1.5 Structure related problems

1.5.1 Structure prediction

Here one is concerned about the relation between sequence and structure. The sequence can either be from a protein, in which case the problem is sometimes referred as the *protein folding problem*; a more simple variant is that of RNA folding.

Now for natural protein sequences, the protein folds into one stable structure (which is believed to be a structure where the free energy has a global minima), which is completely determined by its amino acids sequence. This native structure determines the function of a protein. Since it is very easy to determine the sequence of a protein, the structure prediction problem consists of determining the structure from a given sequence. This is one of the holy grail of bioinformatics, since protein structure prediction is a very important but notoriously hard problem. It is subject of many ongoing attempts to solved this problem by a variety of methods (see for example the CASP competitions [18] [99]) Note that for artificial sequences, the sequence usually does not determine the structure (i.e., the artificially designed protein will not fold to a stable structure in general).

Proteins have a high level of local organisation (called secondary structure), which consist of α-helices, β-strands and turns). For that reason, there are approaches for predicting secondary structure first, before the overall tertiary structure is determined, as well as approaches with try to predict tertiary structure directly. It is presently believed that protein structure prediction cannot be done purely on the level of secondary structure alone.

A problem related to the protein folding problem is the *inverse protein folding*, which consists of the following. Given a three-dimensional structure, generate a sequence that will fold uniquely into the given structure. Naively, this can be solved using structure prediction (generate a sequence, then predict the structure, and compare the result with the given structure). Clearly, this problem is of interest for drug design, although inverse protein folding is not used in drug design yet. The reason simply that the problem is unsolved (see e.g. [51], where this problem is treatment for lattice proteins).

For RNA, secondary structure is usually related to base pair bonding, and structure prediction is possible on this level (under some restrictions) taken into account thermodynamical energies [112].

1.5.2 Structure-prediction for lattice models of proteins

Introduction

To tackle protein structure prediction and related problems simplified protein models have been introduced. These simplified models have been successfully used by several groups in the international contest on automated structure prediction. The most important class of simplified models are the so-called lattice models. The simplifications commonly used in this class of models are: 1.) monomers (or residues) are represented using a unified size 2.) bond length is unified 3.) the positions of the monomers are restricted to lattice positions, and 4.) a simplified energy function.

Apart from their use in structure prediction, they have became a major tool for investigating general properties of protein folding. They constitute a genotype (protein sequence) versus phenotype (protein conformation) mapping that can be dealt with using computational methods. Thus, they can be used to investigate evolutionary processes. An example is [14], where so-called neutral networks have been investigated. The edges of the network are pairs of sequences which differ only in one sequence positions, but have the same minimal energy conformation. Thus, a neutral network represents all protein sequences encoding the same protein conformation. The question is whether one can switch between two different neutral networks using only a small number of amino-acid substitutions. If this is the case, then this suggest a way evolution could have produced the diversity of protein conformations found in nature.

The simplest model is the HP-model, which is an important representative of lattice models. It has been introduced by Lau and Dill in [70]. In this model, the 20 letter alphabet of amino acids is reduced to a two letter alphabet, namely H and P. H represents *hydrophobic* amino acids, whereas P represent *polar* or hydrophilic amino acids. In natural proteins, the hydrophobic amino-acids tend to be in the middle of the protein (forming a compact hydrophobic core), whereas the hydrophilic ones tend to be on the surface of the protein, thus interacting with the surrounding water. This is modeled in the energy function for the HP-model, which is given by the matrix as shown in Figure 1.2(a). It simply states that the energy contribution of a contact between two monomers is -1 if both are H-

(a)

	H	P
H	-1	0
P	0	0

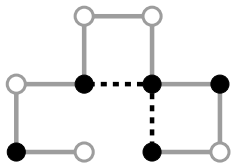
(b) 

Figure 1.2: Energy matrix and sample conformation for the HP-model

monomers, and 0 otherwise. Two monomers form a *contact* in some specific conformation if they are not connected via a bond, and the euclidian distance of the positions is 1. A conformation with *minimal energy* (also called *optimal conformation*) is just a conformation with the maximal number of contacts between H-monomers. Just recently, the structure prediction problem has been shown to be NP-complete even for the HP-model [11, 24].

A sample conformation for the sequence PHPHPPHHPH in the two-dimensional lattice with energy -2 is shown in Figure 1.2(b). The white beads represent P, the black ones H monomers. The two contacts are indicated via dashed lines.

So far, most of the existing approaches are heuristic methods like the hydrophobic zipper [28], the genetic algorithm by Unger and Moult [103], the chain growth algorithm by Bornberg-Bauer [13], or monte-carlo approaches with simulating annealing like [8], which is a monte-carlo method applicable for any regular lattice. There are only two approaches available that are able to prove optimality of the found conformations, namely the constraint-hydrophobic core construction (CHCC) [111], and the constraint-based protein folding method [5], which we will describe here in more detail. It is the first methods that is applicable to two different lattices (the cubic lattice, and the face-centered-cubic lattice), and to different energy functions (namely the HP-model and its extension HPNX, which also encodes charged amino acids). Using this constraint-based approach, we were able to find minimal energy conformations (and prove their optimality) for sequences up to length 300. In contrast, the CHCC method, which is not based on constraint programming, was only applied to sequences up to length 86. In the following, we will handle only the cubic lattice, albeit the face-centered-cubic lattice (FCC) is more suited for modeling protein conformations, but is also more complex.

A simple constraint-based formalization

A sequence is an element in $\{H, P\}^*$. With s_i we denote the i^{th} element of a sequence s. We say that a monomer with number i in s is even (resp. odd) if i is even (resp. odd). A conformation c of a sequence s is a function

$$c : [1..|s|] \rightarrow \mathbb{Z}^d$$

(where $d = 2$ or $d = 3$ depending on whether we consider a 2-dimensional or a 3-dimensional lattice) such that

1. $\forall 1 \leq i < |s| : ||c(i) - c(i+1)|| = 1$ (where $|| \cdot ||$ is the euclidian norm on $\mathbb{Z}^d$)
2. and $\forall i \neq j : c(i) \neq c(j)$.

The first condition is imposed by the lattice constraint and implies that the distance vector between two successive elements must be a unit-vector (or a negative unit-vector) in every

admissible conformation. The second condition is the constraint that the conformation must be self-avoiding.

Given a conformation c of a sequence s, the number of contacts $Contact_s(c)$ in c is defined as the number of pairs (i, j) with $i + 1 < j$ such that

$$s_i = H \wedge s_j = H \wedge ||c(i) - c(j)|| = 1$$

(in other words, the number of pairs of H-monomers that have distance 1 in the conformation c, but are not successive in the sequence s). The energy of c is just $-Contact_s(c)$. With $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$ we denote the unit vectors $(1, 0, 0)$, $(0, 1, 0)$ or $(0, 0, 1)$, respectively. We say that two points $\vec{p}, \vec{p}' \in \mathbb{Z}^3$ are *neighbors* if $\|\vec{p} - \vec{p}'\| = 1$. This is equivalent to the proposition that $\vec{p} = \vec{p}' \pm \vec{e}$ with $\vec{e} \in \{\vec{e}_x, \vec{e}_y, \vec{e}_z\}$.

This can now be directly encoded as a constraint problem. Our constraint problem consists of finite domain variables. We use also Boolean constraint and reified constraints. With *reified constraints* we mean a constraint $x =: (\phi)$, where ϕ is a finite domain constraint. x is a Boolean variable which is 1 if and only if ϕ holds. Technically, this can be achieved via setting x to 1 if the constraint store entails ϕ, and to 0 if the constraint store disentails ϕ. A constraint store *entails* a constraint ϕ if every valuation that makes the constraint store valid also makes ϕ valid. 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. We have implemented the problem using the language Oz [94], which supports finite domain variables, Boolean constraints, reified constraints, entailment constraints and a programmable search module.

Now we can encode the space of all possible conformations for a given sequence as a constraint problem as follows. We introduce for every monomer i new variables X_i, Y_i and Z_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. But we can even restrict the possible values of these variables to the finite domain $[1..2n]$.[3] This is expressed by introducing the constraints

$$\mathrm{X}_i \in [1..(2 \cdot length(s)] \wedge \mathrm{Y}_i \in [1..(2 \cdot length(s)] \wedge \mathrm{Z}_i \in [1..(2 \cdot length(s)]$$

for every $1 \leq i \leq n$. The self-avoidingness is just $(\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i) \neq (\mathrm{X}_j, \mathrm{Y}_j, \mathrm{Z}_j)$ for $i \neq j$.[4]

For expressing that the distance between two successive monomers is 1, we introduce for every monomer i with $1 \leq i < length(s)$ three variables Xdiff_i, Ydiff_i and Zdiff_i. The value range of these variables is $[0..1]$. Then we can express the unit-vector distance constraint by

$$\begin{array}{ll} \mathrm{Xdiff}_i =: |\mathrm{X}_i - \mathrm{X}_{i+1}| & \mathrm{Zdiff}_i =: |\mathrm{Z}_i - \mathrm{Z}_{i+1}| \\ \mathrm{Ydiff}_i =: |\mathrm{Y}_i - \mathrm{Y}_{i+1}| & 1 =: \mathrm{Xdiff}_i + \mathrm{Ydiff}_i + \mathrm{Zdiff}_i. \end{array}$$

The constraints described above span the space of all possible conformations. I.e., every valuation of $\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i$ satisfying the constraints introduced above is an *admissible* conformation for the sequence s, i.e. a self-avoiding walk of s. Given partial information

[3] We even could have used $[1..n]$. But the domain $[1..2n]$ 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.

[4] This cannot be directly encoded in Oz [94], but we reduce these constraints to difference constraints on integers.

about $\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i$ (expressed by additional constraints as introduced by the search algorithm), we call a conformation c *compatible* with these constraints on $\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i$ if c is admissible and c satisfies the additional constraints.

The most simplest way to search for conformations with maximal number of contacts would be to add constraints for counting the number of contacts. Then one can directly enumerate the variables X_i, Y_i and Z_i. For HP-type models, we have to count contacts which are always generated between two neighboring H-monomers. For this purpose, one introduces a variable $Contact_{i,j}$ that is 1 if i and j have a contact in every conformation which is compatible with the valuations of $\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i$, and 0 otherwise. Then

$$\begin{aligned} \mathrm{Xdiff}_{i,j} &= |\mathrm{X}_i - \mathrm{X}_j| \qquad \mathrm{Zdiff}_{i,j} = |\mathrm{Z}_i - \mathrm{Z}_j| \\ \mathrm{Ydiff}_{i,j} &= |\mathrm{Y}_i - \mathrm{Y}_j| \qquad Contact_{i,j} \in \{0,1\} \\ (Contact_{i,j} = 1) &\leftrightarrow (\mathrm{Xdiff}_i + \mathrm{Ydiff}_i + \mathrm{Zdiff}_i = 1) \end{aligned} \tag{1.9}$$

where $\mathrm{Xdiff}_{i,j}$, $\mathrm{Xdiff}_{i,j}$ and $\mathrm{Zdiff}_{i,j}$ are new variables. The variable $HHContacts$ counts the number of contacts between H-monomers, and is defined by

$$HHContacts = \sum_{\substack{i+1<j\wedge \\ s(i)=H\wedge s(j)=H}} Contact_{i,j}. \tag{1.10}$$

Now we could start to apply constraint-based enumeration on $\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i$ searching for a conformation with maximal number of contacts.

The main problem using this approach alone is that it is very difficult to define good bounds and to find a search heuristic for enumerating low-energy conformation first. Nevertheless, this formulation is in part required for lattice models with an extended alphabet like the HPNX-model [7], which models also electrostatic contacts in addition to hydrophobicity.

Dal Palù et al. [80] considered an extension of the above problem for a much more sophisticated energy function. Since it is not possible to solve the problem optimally or near-optimally in the case of extended energy functions, they integrated additional biological knowledge to achieve good predictions. Starting from a formulation of the protein folding problem for the face-centered cubic lattice similar to the one described in Eq (1.9), they integrated secondary structure information in the prediction process. In a later work, Dal Palù et al. [79] extended the simple formulation Eq (1.9) by introducing variable that have three-dimensional domains (called *Box-domains*) associated, and described a constraint system and propagation techniques for this kind of variables. A similar approach of using variables with three-dimensional domains was successfully used in the PSICO-system [67] for the prediction of protein structure from Nucleic-Magnetic-Resonance (NMR) data. NMR is an experimental technique for determining protein structure. The result is a set of *distance constraint* that give an estimate of the pairwise distances between the atoms of the proteins. Here, the task is to find a structure minimizing an energy function that is compatible with the distance constraints (or to be more precise, that minimizes the violation of distance constraints).

A more sophisticated approach

To overcome the problem of finding good bounds and search heuristics, the set of all conformations was restricted in [5] to a subset of conformations that contains provably all minimal energy conformations. For this purpose, the hydrophobic core, which consists of the the positions occupied by H-monomers, is calculated first. Then, in a second step, a conformation of the HP-sequence is searched that has exactly the hydrophobic core calculated before. Since this problem is a strongly constrained, conformation can be found in relatively short time. Of course, all possible maximal compact hydrophobic cores have to be considered.[5] Formally, a *hydrophobic core* $\mathbb{C}$ is just a set of positions. The number of contacts in a hydrophobic core $\mathbb{C}$ is defined by

$$Contact(\mathbb{C}) = \frac{1}{2}|\{(\vec{p}, \vec{p}') \mid \vec{p}, \vec{p} \in \mathbb{C} \wedge \vec{p} \text{ and } \vec{p} \text{ are neighbors}\}|$$

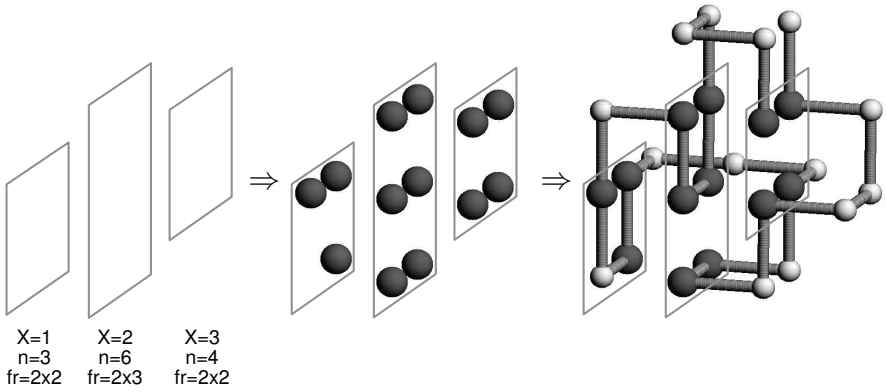
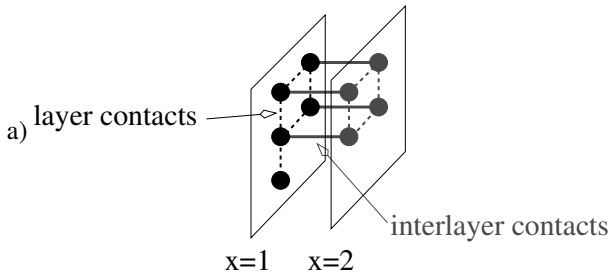


Figure 1.3: The overall approach

Finding all maximally compact hydrophobic cores is an optimization problem itself, which was solved in [5] again in a two-level step. First, the distribution of H-monomers to layers of the form $X = i$ is calculated. Such a distribution is called a frame sequence, and consists of the number of H-monomer in each layer, as well as the minimal rectangle around these monomers. As we will see later, this information can be used to calculate an upper bound on the number of contacts for a specific frame sequence, which allows one to discard many frame sequences. Then, for a given frame sequence, all possible maximally compact hydrophobic cores having the corresponding frame sequence are generated. Thus, we have the overall 3-level approach depicted in Figure 1.3.

Enumeration of Frame sequences The basic idea for the upper bound on the number of contacts is to classify the contacts into contacts between positions in the same layer (called *layer contacts*), and contacts between positions in successive layers (called *inter-layer contacts*). To give an upper bound for a specific frame sequence, one gives separate

[5] If there is no conformation found for the maximally compact core, then sub-optimal cores have to be considered as well, which is not very often the case.



b)

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

Figure 1.4: a) Layer and Interlayer Contacts b) Corresponding Frame Sequence

bounds for the number of layer *and* interlayer contacts for hydrophobic cores have this frame sequence. In Figure 1.4a), a the hydrophobic core for the cubic lattice is shown together with the layer and interlayer contacts. The corresponding frame sequence is given in Figure 1.4b).

For the layer contacts, consider a frame of size $a \times b$ with n H-monomers. For finding the maximal number of layer contacts that any hydrophobic core with this frame can have, Yue and Dill [110] observed that it is much simpler to calculate the surface instead of the number of layer contacts. The *layer surface* of an hydrophobic core $\mathbb{C}$ in layer $x = k$ is the number of positions $\vec{p}$ in layer $x = k$ that are not in $\mathbb{C}$, but neighbors of some position $\vec{p}' \in \mathbb{C}$ ($\vec{p}$ is called a *surface point*). Since every position in the core has 4 neighbors, which are filled by another member of the core or by a surface point, it is clear that surface and contacts are related via the equation

$$4n = 2Contact + 2a + 2b.$$

Hence, minimizing the surface maximizes the number of contacts.

Now whenever we have a layer where a surface point is buried between two position from the core, this core cannot be maximal. We can achieve a more compact one by resorting the core positions in this layer in a way that the gap generated by this surface point is closed (recall that a hydrophobic core is just a set of positions, with no other conditions imposed on them). Under the condition of a maximal compact layer core, this implies that every horizontal and vertical line that goes through the core in some layer must generate 2 surface points. Hence, a frame of size $a \times b$ must generate at least $2a+2b$ surface points. Furthermore, one can conclude that an optimal frame for n points must minimize $a + b$, which is the case for a nearly quadratic frame. I.e., the best possible adaption of a quadratic frame with $a = \lceil \sqrt{n} \rceil$ and $b = \lceil \frac{n}{a} \rceil$ will have minimal surface, which used as a bound when enumerating number sequences.

For the interlayer contacts in the cubic lattice, one simply observes that every monomer in one layer can have at most one contact in the following layer. Thus, the maximal number of interlayer contacts for two successive layers $X = i$ and $X = i + 1$ having n_i and n_{i+1} monomers is $\min(n_i, n_{i+1})$. Using this upper bounds for layer and interlayer contacts, one can calculate the optimal frame sequence using a dynamic programming approach.

For the face-centered cubic lattice, the calculation of the bound is more complicated. Albeit one can uses also a splitting of the core into successive layers and apply the same

bound for layer contacts, the bound for the interlayer contacts is more difficult. The reason is that every H-monomer in one layer can have up to 4 contact in the successive layer. For details the reader is referred to [2].

Construction of Hydrophobic cores Once we have a frame sequence a_k, b_k, n_k for $k = 1 \ldots m$, one has to enumerate the possible hydrophobic cores for this frame sequence. The first step is to fix the frame positions in each layer. That is, we have finite domain variables sy_k and sz_k for the lower left corner of the frame in layer $x = k$. We can choose $sy_1 = sz_1 = 0$ for the first frame. For the remaining frames, we have to enumerate in principle all possible starting positions. But again, we can use bounds to discard combination of values for sy_k, sz_k that may not result in a maximal compact hydrophobic core.

An example of such a bound is the following. A combination is unfavorable if a frame does not completely overlap with the previous frame. Then only the part of the two frames that do overlap can generate interlayer contacts. Hence, we can use the bounds on the interlayer contacts described in the last section to calculate the number of interlayer contacts for the overlapping sub-frames.

Once we have fixed the frames (via determining their lower left corners), we start be enumerating the positions that are actually contained in the core. This can be done by inserting for every position a Boolean variable $c_{\vec{p}}$ for every position $\vec{p}$ that is in one of the fixed frames. Then

$$c_{\vec{p}} = 1 \quad \text{iff} \quad \vec{p} \text{ is in the core.}$$

Clearly, we have

$$\left(\sum_{\vec{p} \text{ is in Layer } x = k} c_{\vec{p}} \right) \quad = \quad n_k.$$

Since a frame is usually tightly filled, this constraint provides good propagation. Finally, we have to encode contacts by using a Boolean variable $Contact_{\vec{p},\vec{p}'}$ for each pair of neighbors $\vec{p}, \vec{p}'$. Then

$$Contact_{\vec{p},\vec{p}'} = 1 \Leftrightarrow (c_{\vec{p}} = 1 \wedge c_{\vec{p}'} = 1).$$

Counting $Contact_{\vec{p},\vec{p}'}$ will gives us the total number of contacts for the core.

We can improve propagation by the following consideration. Usually, hydrophobic cores do not have too many caveats. A *caveat* is a P-monomer which is part of the hydrophobic core and thus buried by H-monomers. This usually produces a non-optimal core, but must be considered in the case that the optimal cores do not correspond to a valid sequence conformation. If a frame does not contain any caveat, then we know that for any line through the frame, the H-monomers must be consecutive on this line. Now suppose we have two positions $\vec{p}$ and $\vec{p'}$ in the same frame with the property that $\vec{p}$ is the left neighbor of $\vec{p'}$, $c_{\vec{p}} = 0$ (P-position) and $c_{\vec{p}} = 1$ (H-position). Then all positions to the left of $\vec{p}$ on the line through $\vec{p}, \vec{p'}$ must be P-positions as well (see Figure 1.5). For a given pair of left neighbors $\vec{p} = (k, s, t)$ and $\vec{p}' = (k, s+1, t)$, this can be simply expressed by the following

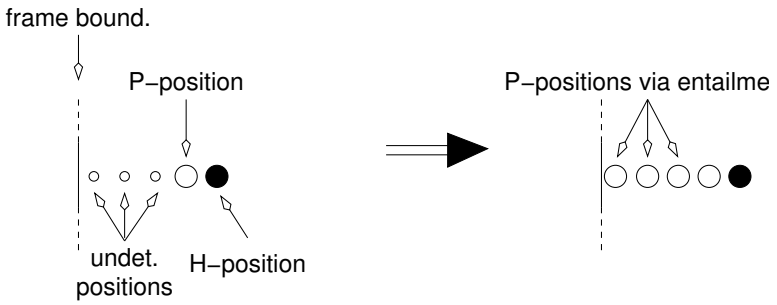


Figure 1.5: Example of the caveat-freeness constraint

entailment constraint:

$$c_{\vec{p}} = 0 \wedge c_{\vec{p}'} = 1 \implies \bigwedge_{\substack{\vec{p}'' = (k,s,t) \\ \text{in frame} \\ \text{with } r < s}} c_{\vec{p}''} = 0$$

Of course, we have to introduce such a constraint for every pair of left, right or vertical neighbors. For more details, the reader is referred to [1] and [108]. If caveats are allowed, then one can enumerate them explicitly and add the constraint for the remaining positions.

1.5.3 Protein docking and ligand binding

Protein docking attempts to find the most stable mode of association between two protein molecules, starting from the atomic coordinates of the two isolated components. It can be likened to a 'lock and key' mechanism , where both lock and key are plastic, and distort according to mutual interactions. The protein-protein interfaces are closely packed, similar to protein cores. The aim of any docking algorithm is to optimise the surface area and attractive forces and to minimise the loss of energy due to interaction with the solvent. This is a difficult area of research, but there are general rules. Optimisation must be performed on many degrees of freedom, since this is an example of 6-D problem of rigid body movement - 3 translations and 3 rotations, all of which must be searched. The approaches to rigid surfaces are broadly

1. Given the information of a pair of proteins crystallised together, to *reconstruct* the docking

2. Given the individual proteins separately crystallised, to *predict* their docking. requires trying all combinations of degrees of freedom note that *ligand binding* - small ligands tend to bind in big pockets; ligands are more flexible than proteins

1.5.4 Structure motif matching

Protein structures can be described at various levels of detail, ranging from atomic coordinates, through vector approximations to secondary structures elements (SSEs), to 'topological' models. These latter abstractions typically consider a sequence of SSEs, i.e. helices

or strands, together with relationships like spatial adjacency within the fold and approximate orientation, neglecting details like lengths and structures of loops, and the lengths of the secondary structure elements themselves. This level of abstraction can be useful to permit very fast algorithms for structure motif matching, discovery and structure comparison. Further, by neglecting many of the details which typically vary between related structures, like lengths and structures of loops, and exact lengths, spatial positions and orientations of SSEs, it has the potential to detect more distant structural relationships than could be found by methods based on more geometrical descriptions. On the other hand, its disadvantages are that there may be structures which, although related at the topological level, are very different from a geometric point of view, and have no meaningful biological relationship.

A TOPS *structure* is a triple (E, H, C) where $E = S_1, \ldots, S_k$ is a sequence of length k of secondary structure elements (SSEs) and H and C are relations over the SSEs, called respectively H-bonds and chiralities. In this description an H-bond constraint refers to a ladder of individual hydrogen bonds between adjacent strands in a sheet. An SSE S is a character from the alphabet $\{\alpha, \beta\}$ standing for helix and strand respectively. Since each SSE in a TOPS structure is associated with a direction *up* or *down* we associate a direction symbol, $+$ or $-$, with each letter of this alphabet. Both H-bonds and chiralities are symmetric relations (non-directed arcs in the graph). An H-bond constrains the types of the two SSE's involved to be strands, and each bond is associated with a relative direction $\delta \in \{P, A\}$, indicating whether the bond is between parallel or anti-parallel strands. Chiralities are associated with handedness $\chi \in \{L, R\}$ (left and right respectively), and only occur between pairs of SSEs of the same type. We denote the H-bond relationship between two SSEs S_i and S_j by (S_i, δ, S_j) and a chirality relationship by (S_i, χ, S_j).

Definition 8 (TOPS structure). *Given* $\Sigma = \{\alpha_+, \alpha_-, \beta_+, \beta_-\}$*, then a TOPS structure* D *is defined by the triple* (S, H_d, C_d)*, where*
$S = (S_1, \ldots, S_k)$, $S_i \in \Sigma$
$H_d = \{(S_i, \delta, S_j) | S_i, S_j \in \{\beta_+, \beta_-\},\ \delta = P \leftrightarrow S_i = S_j,\ \delta = A \leftrightarrow S_i \neq S_j\}$
$C_d = \{(S_i, \chi, S_j) | S_i, S_j \in \Sigma,\ \chi \in \{R, L, \}\}$

As an example, in Figure 1.6 we give a TOPS structure for the protein structure "2bop" (Protein Databank code) both in a form with '2-D' layout as well as in a linear form form. The textual form of the TOPS description for 2bop is:

2bop = (E, H, C), where
$E = (\beta_{+_1}, \alpha_{-_2}, \alpha_{-_3}, \beta_{+_4}, \beta_{+_5}, \beta_{-_6}, \alpha_{+_7}, \beta_{-_8})$
$H = \{(\beta_{+_1}, A, \beta_{-_6}), (\beta_{+_1}, A, \beta_{-_8}), (\beta_{+_4}, A, \beta_{-_6}), (\beta_{+_5}, A, \beta_{-_6})\}$
$C = \{(\beta_{+_1}, R, \beta_{+_4}), (\beta_{-_6}, R, \beta_{-_8})\}$

A TOPS *pattern*, or *motif*, is similar to a TOPS structure, but is a generalisation which can describe several structures conforming to some common topological characteristics. This generalisation is achieved by permitting 'gaps', standing for the insertion of SSEs (and any associated H-bond and chiralities), in the sequence of secondary structure elements; indeed a structure is just a pattern where no inserts are permitted. A gap is described by a pair (n, m) where n stands for the minimum and m for the maximum number of SSEs which can be inserted at that position. The range of n and m is from zero to the largest number of SSE's in any TOPS structures (approximately 60).

In principle, just as for TOPS structures, each SSE in a TOPS pattern is associated with a direction *up* or *down* ($+$ or $-$ respectively) relative to the X-axis, and is a character from the alphabet $\{\alpha, \beta\}$.

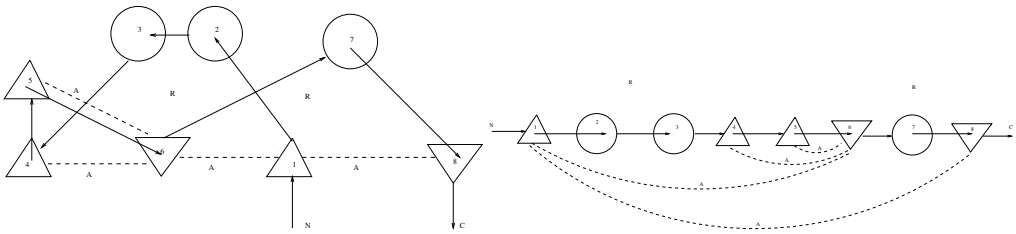


Figure 1.6: TOPS structure for 2bop. Circles represent α-helical secondary structure elements, triangles represent β-strand secondary structure elements, arrows represent loop regions, heavy dotted lines represent hydrogen-bond relationships ('A' – anti-parallel), light dotted lines represent chiralities ('R' — right-handed)

However, since any TOPS description of pattern (or a structure) can be flipped about the X-axis without loss of meaning, in order to facilitate pattern matching we associate a *direction variable*, $\oplus$ or $\ominus$ with each SSE in a pattern P s.t. they satisfy the constraint

$$\forall \oplus, \ominus \in P : opp(\oplus, \ominus) \leftrightarrow (\oplus = + \wedge \ominus = -) \vee (\oplus = - \wedge \ominus = +)$$

Note that it is possible, but redundant if we are to perform pattern matching, to associate a similar constraint with each SSE in a structure description.

Definition 9 (TOPS pattern). *Given* $\Sigma = \{\alpha_\oplus, \alpha_\ominus, \beta_\oplus, \beta_\ominus\}$ *then a TOPS pattern* $P = (T, H_p, C_p)$, $\forall \oplus, \ominus \in P : opp(\oplus, \ominus)$, *where* $T = (n_0, m_0) - V_1 - (n_1, m_1) - V_2 - \ldots - (n_{k-1}, m_{k-1}) - V_k - (n_k, m_k)$, $V_j \in \Sigma$, $n_j \leq m_j$
$H_p = \{(S_i, \delta, S_j) | S_i, S_j \in \{\beta_\oplus, \beta_\ominus\},\ \delta = P \leftrightarrow S_i = S_j,\ \delta = A \leftrightarrow S_i \neq S_j\}$
$C_p = \{(S_i, \chi, S_j) | \chi \in \{R, L, \},\ S_i, S_j \in \Sigma\}$

For example a TOPS pattern which describes plaits (2bop is an instance of a plait) is illustrated in Figure 1.7; arrows between SSEs in the sequence have been annotated with pairs of integers standing for (n_i, m_j), in this case $(0, N)$.

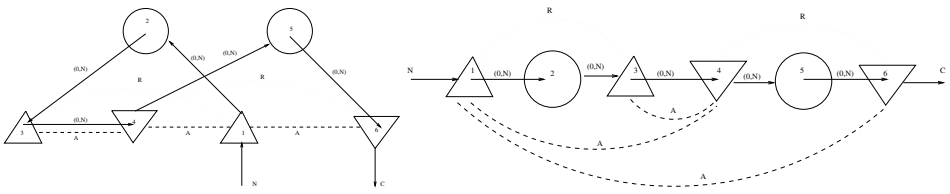


Figure 1.7: TOPS plait motif

Definition 10 (Size of a TOPS structure (resp pattern)). *The size of a TOPS structure* $D = (S, H, C)$ *(resp. pattern) is* $|S|$, *the number of SSEs in the structure (pattern).*

Gilbert et al [47] have defined a simple backtracking algorithm which is guaranteed to find all the ways in which a TOPS pattern matches a TOPS structure; for each match it returns the set of pairs of corresponding SSEs between the pattern and the structure, and the set of corresponding insert sizes in the pattern.

Finite domain constraints over integers are used in the algorithm in order to prune the search space. A correspondence is established between the SSE numbers in a structure size j and the SSE numbers in a pattern size k $Corr := (d_1, d_2, \ldots, d_k)$, where d_i ($i \in 1 \ldots k$) is a constraint variable representing the number of the SSE in the structure matching SSE i in the pattern. In addition $Ins := (I_1, I_2, \ldots, I_{k-1})$ is the sequence of insert sizes, where I_i ($i \in 1 \ldots k-1$) is a constraint variable representing the number of inserts between SSEs i and $i+1$ in the pattern, The matching algorithm *imposes constraints* on the SSEs in the pattern by setting up constraints for $i \in 1..k$, C1: $1 \leq d_i \leq j$
C2: $n_i \leq I_i \leq m_i$,
C3: $d_i + I_i + 1 = d_{i+1}$
Constraint C1 gives the range of d_i (a pattern cannot have more SSEs than a matching structure); C2 sets up a constraint variable for each insert in the pattern, and C3 ensures that the insert sizes are respected in the matching.

The simple algorithm then proceeds by matching the H-bonds (respecting the parallel/antiparallel labels), the chiralities (respecting the right/left-hand labels) and the SSEs (respecting the type and orientation) between the pattern and the structure.

In fact, matching of TOPS motifs to TOPS structures is an instance of the subgraph isomorphism problem which remains NP-complete for such vertex ordered graphs. There are several non-polynomial algorithms for subgraph isomorphism problem, the most popular being by Ullmann [102] and McGregor [76]. Although these are not straightforwardly adaptable to vertex ordered graphs, the vertex ordering seems to be the property that could considerably improve the algorithm efficiency.

Viksna et al. [106] give a fast matching algorithm for TOPS structures, which is a variant of a method based on constraint satisfaction [76]. The algorithm tries to match edges in the increasing order of edge positions and backtracks if for some edge match can not be found. Since the graphs are ordered, the positions in the target graph to which a given edge may be mapped and which have to be checked can only increase. Two additional ideas are used to make this process more efficient. Firstly, a number of additional labels are assigned to vertices and edges; they comprise the numbers of incoming and outgoing edges of all possible types for a given vertex, whilst for an edge they describes how many "shorter" or "longer" other edges are connected to the endpoints of a given edge. describes how many shorter or longer other edges are connected to the endpoints of a given edge Secondly, if an edge *e* can not be mapped according to the existing mapping for previous edges, then the next place where this edge can be mapped according to the labels is found, and the minimal match positions of previous edges are advanced in order to be compatible with the minimal position of *e*. The full algorithm is given in [106].

1.5.5 Structure motif discovery

Pattern discovery for sequences is a well-established technique [15] which could be applied to TOPS structures as follows. The first, "pattern driven" (PD) is based on enumerating candidate patterns in a given solution space and picking out the ones with high fitness; the second, "structure driven" (SD) comprises algorithms that try to find patterns by comparing

given diagrams and looking for local similarities between them. In SD an algorithm may be based on constructing a local multiple alignment of given sequences and then extracting the patterns from the alignment by combining the segments common to most of the sequences.

Essentially the difference between pattern discovery for sequences and TOPS structures is that techniques for the former assume that the grammar of the former is regular whilst that of the latter is context–sensitive. Thus in a naive version of a PD approach for TOPS diagrams not only would we have to enumerate an exponentially large number of patterns comprising not only all the possible combinations of the SSEs (and their orientations) in a pattern of length k, but also all the possible H-bond and chirality connections over them.

Viksna et al [106] find maximal common subgraphs for a set of TOPS graphs by an exhaustive search comprising repeated extension of an initial subgraph and checking for subgraph isomorphisms in the target set of graphs. In doing so, they exploit the speed of their specialised subgraph isomorphism algorithm for TOPS graphs. Starting with a simple (one vertex) pattern graph, subgraph isomorphism is used to check against all graphs in a given set and in the case of success attempt to extend the already matched pattern graph in all possible ways. Some restrictions on the number of different types of edges and vertices can be deduced from the given set of target graphs and are used by the algorithm. Apart from that, the previous successful match may be used to deduce information about extensions which are more likely to be successful in the next match. In general this does not prune the search space but may help to discover large common subgraphs earlier. The advantage of this approach is that the algorithm has time complexity that is linear with respect to the number of graphs in the given input set.

Gilbert et al. [46] report an heuristic algorithm which discovers patterns of H-bonds (and chiralities) based on the properties of sheets for TOPS diagrams; they also derive patterns on the associated sequences of SSEs and insert sizes. Briefly, the algorithm attempts to discover a new sheet by finding, common to all the target set of diagrams, a (fresh) pair of strands, sharing an H-bond with a particular direction. Then it attempts to extend the sheet by repeatedly inserting a fresh strand which is H-bonded to one of the existing strands in the (current) sheet. The algorithm then finds all further H-bonds between all the members of the current sheet. The entire process is repeated until no more sheets can be discovered; any chirality arcs between the H-bonds in the pattern are then discovered by a similar process. The numbers of inserts between each strand in the pattern are then computed for all the patterns in the learning set, and the minimum and maximum size of the gaps in the corresponding insert positions in the pattern are thus found, and combined with the SSE sequence to give the T-pattern. The result is the least general common TOPS pattern characterising the target set of protein descriptions.

Other methods that are known mostly correspond to the SD approach outlined above, for example as described by Koch et al. [66]. These may be more efficient for sets containing a small number (basically just two) of graphs, but in general cannot be used to find the exact answer to the problem for larger sets.

The goodness of a pattern can be stated in several ways, including the *size* of the pattern, its discriminative performance against a set of positive and negative examples, and its *compression* value. In [45] Gilbert et al describe how to compute the compression of a TOPS pattern with respect to a set of graphs of structures using a general data compression measure applied to the size of the pattern graph and the total size of the components of the structures which are not included in the pattern. This value can be normalised to the range 1 (best) to 0 (worst).

Definition 11 (Raw compression). *The raw compression of a pattern length k w.r.t. a set of n structures of lengths $l_1, \ldots, l_n$ is*
$\Sigma_{i=1}^{n} l_i - (n-1) * k$

Definition 12 (Normalised compression). *The normalised compression of a pattern length k w.r.t. a set of n structures of lengths $l_1, \ldots, l_n$ is*
$\frac{(n-1)*k}{\Sigma_{i=1}^{n} l_i - min_{i=1}^{n}(l_i)}$

These definitions can be extended in a natural way to include complete structural definitions (H-bonds and chiralities). When there are only two structures in the set, the compression measure can be used as a measure of structure comparison, as ultilised in the online TOPS system reported by Torrance et al [101] which operates over the TOPS database [77].

1.6 Function related problems

1.6.1 Metabolic pathways

Living organisms function by a complex set of interactions at the molecular level which occur in a highly organised manner. They involve metabolic reactions which transform some compounds (*substrates*) into others (*products*). In general a reaction $S \rightarrow P$ can be described by a transition $S \rightarrow S' \rightarrow P$, transforming the set of substrates S into the set of products P via a transition state S' in which the substrate molecules are distorted into some electronic conformation which more readily converts to the products. In order to occur, $S \rightarrow P$ has a negative free energy, i.e. the free energy of S is greater than that of P; however $S \rightarrow S'$ has a *positive* free energy change, termed the energy of activation. This energy is a barrier preventing $S \rightarrow P$ occurring spontaneously, without which all reactions would occur in an uncontrolled way. Most reactions are catalysed by special proteins called *enzymes* which control the reaction by lowering the energy barrier (i.e. increasing the rate of flow). They do this by binding substrates at combining sites within active sites, positioning substrate molecules in the most favourable orientations for reactions to occur, as well as distorting them in order to favour transition state formation. During this process the enzyme may change shape in order to induce a fit with the substrate, rather than just rely on a rigid 'lock and key' mechanism. In general, reactions can be chained together into paths so that the products of one reaction become the substrates of another [36].

1.6.2 Regulatory networks

Metabolic reactions can be regulated in two ways. The first is by the direct activation or inhibition of activity of enzymes by small molecules. This method is relatively fast in action, since it directly affects the chain of reactions. Another method of regulation is that of transcriptional regulation, in which the production of the enzyme itself is controlled by a transcription factor (a protein which activates the capacity of a gene to produce another protein). This method is relatively slow, since it indirectly affects the reaction path.

Reactions can be self-regulated using either the direct or transcriptional method, since it is common that products of an immediate or eventual reaction act have a direct or transcriptional effect on enzymes involved earlier in the chain of reactions. These regulatory

relationships can be quite complex in that products from one path can regulate enzymes involved in another path.

1.6.3 Querying and analysing networks of cellular function

In [105] van Helden et al. give a data model for representing and analysing networks of cellular function (metabolic and regulatory pathways). This has been extended by Deville et al. [26] to the general case including signalling pathways. Often the information is stored in a database, with the associated the database model permitting simple analysis to be directly be performed on through a database query language which are often unsuitable for algorithmic use. Specific algorithms with their own data structures are required for more sophisticated analyses. Often graphs are used as representational data structures – these can be compound, reaction, bipartite and hyper-graphs. Object-oriented models can be seen as a generalization of bipartite graphs, where the nodes are typed, permiting detailed descriptions, and the use of inheritance to structure data.

Current computational systems often path navigation routines in addition to simple data retrieval. A simple query is to get all the reactions catalysed by a gene product More complex queries require the application of specialised algorithms, often involving the use of graph analysis. These are for example (adapted from [105])

- find all metabolic pathways that convert compound A into compound B in less than X steps
- find all genes whose expression is directly or indirectly affected by a given compound.
- find all compounds that can be synthesised from a given precursor in less than X steps
- in the complete set of metabolic reactions, find all feedback loops including a given compound, or, in a defined biochemical pathway, find all feedback loops.

Another type of complex queries involve sub-graph extraction. Here the user specifies a set of seed nodes in the network the system is required to extract the portions of the network or sub-graphs that interconnect each pair of seed nodes via the smallest number of individual links. The user can specify the maximum number of individual links, or graph arcs, that can be inserted between any two seed nodes. The resulting sub-graph can then be displayed and analysed. Algorithms for sub graph extraction and maximal path enumeration used in this context have been described in van Helden et al [104]. Examples of major databases and computational systems for storing and analysing biochemical pathway and network data include KEGG (Kyoto Encyclopedia of Genes and Genomes) [61], BioCyc [68], and Amaze [72].

In recent work Dooms et al have described an approach using constraint programming to solve constrained path finding problems in metabolic networks [29] [30], and have applied it to discover pathways from a set of their reactions. [78]. This approach builds on earlier work by the same authors [31] in which they defined a graph computation domain for constraint programming in order to provide a high level modeling language with the data and results are graphs.

1.7 Microarrays

DNA microarrays ("DNA chips") are made by the deposition of DNA spots on a solid support, often a coated glass surface. For an in-depth review, see e.g. [19]. Two main procedures have been used to produce these: photolithography (e.g. by as developed and marketed by Affymetrix Inc. [74], and mechanical gridding [16]. Photolithography is a technique used in the computer microchip industry. There is, however, an inherent length restriction with this *in situ* synthesis technology limiting the probes to about 25 nucleotides in length. This is offset by the use of high-density arrays which allow the use of multiple probes per gene. The arrayed probes can be oligonucleotides (photolithography and gridding) or cDNAs (gridding).

Arrays of thousands of DNA sequences representing part of all of the genome of an organism can be constructed. Such arrays can then be used to compare the relative abundance of the transcriptional products of each of these gene sequences in two DNA or RNA samples, for example from two different cell populations, or from one population exposed to two different stimuli. In the spotting techniques the two samples are first labelled using different fluorescent dyes and are then mixed and hybridized with the arrayed DNA spots. After hybridization, fluorescence measurements are made for each DNA spot, and recording the fluorescence for each dye separately. These measurements are used to determine the ratio, and in turn the relative abundance, of the sequence of each specific gene in the two mRNA or DNA samples. (Adapted from [16]).

The computational challenges can be broadly divided into two major categories:
(1) Normalisation and background correction of microarray data,
(2) Modelling and analysis of the networks that are represented by the sets of genes in the samples.

We briefly overview the second challenge. Network or pathway reconstruction from microarray data is based on observations of the expression of a set of genes under varying conditions such as time-series, targeted mutation or exposure to different evnironmental conditions (stress, starvation etc) [81]. These data are usually taken as steady-state. The goal is to identify which genes control (the expression of) other genes, and the results if these controls. The analysis often involves the clustering of genes by expression data and the analysis of promoter elements within the same clusters. Machine learning techniques are commonly used for reconstgruction of gene networks, for example Soinov et al [95]

A potential application of constraint programming in this area is proposed by Dooms et al [78] is the explanation of DNA microarray experiments using a CSP able to solve pathway discovery problems. However, this area is ripe for the application of constraint computation techniques, both in the processing of low-level (primary) data, as well as in the analysis and interpretation of results, for example cross-referencing into biochemical pathway data.

Bibliography

[1] R. 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.

[2] R. 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.

[3] R. Backofen, N. Narayanaswamy, and F. Swidan. On the complexity of protein similarity search under mrna structure constraints. In H. Alt and A. Ferreira, editors, *Proc. of 19th International Symposium on Theoretical Aspects of Computer Science (STACS2002),*, volume 2285 of *Lecture Notes in Computer Science*, pages 274–286, Berlin, 2002. Springer Verlag.

[4] R. Backofen and S. 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.

[5] R. Backofen and S. 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.

[6] R. Backofen and S. Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2(4):681–698, 2004.

[7] R. Backofen, S. Will, and E. Bornberg-Bauer. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics*, 15(3):234–242, 1999.

[8] R. Backofen, S. Will, and P. 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.

[9] A. Bairoch, P. Bucher, and K. Hofman. The PROSITE database, its status in 1995. *Nucleic Acids Research*, 24(1):189–196, 1996.

[10] G. Benson. Sequence alignment with tandem repeats. In *Proc. of the First Annual International Conferences on Compututational Molecular Biology (RECOMB97)*, pages 27–36, 1997.

[11] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) modell is NP-complete. In *Proc. of the Second Annual International Conferences on Compututational Molecular Biology (RECOMB98)*, pages 30–39, New York, 1998.

[12] B. Billoud, M. Kontic, and A. Viari. Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence databases. *Nucleic Acids Research*, 24(8):1395–1403, 1996.

[13] E. 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.

[14] 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.

[15] A. Brazma, I. Jonassen, I. Eidhammer, and D. R. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5 (2):277–303, 1998.

[16] P. O. Brown and D. Botstein. Exploring the new world of the genome with dna

microarrays. *Nature Genetics*, 21:33 – 37, 1999.

[17] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics*, 48:1073–1082, 1988.

[18] CASP3. http://predictioncenter.llnl.gov/casp3/casp3.html. Third Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction,, Dec 1998.

[19] J. E. Celis, M. Kruhoffer, I. Gromova, C. Frederiksen, M. Ostergaard, T. Thykjaer, P. Gromov, J. Yu, H. Palsdottir, N. Magnusson, and T. F. Orntoft. Gene expression profiling: monitoring transcription and translation products using dna microarrays and proteomics. *FEBS Letters: Functional Genomics*, 480(1):2–16, 2005.

[20] T. Christof, M. Jünger, J. Kececioglu, P. Mutzel, and G. Reinelt. A branch-and-cut approach to physical mapping with end-probes. In *Proc. of the First Annual International Conferences on Compututational Molecular Biology (RECOMB97)*, pages 84–92. ACM Press, 1997.

[21] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction*. Mathematical and Computational Biology. Jon Wiley & Sons, Chichester, August 2000. series editor S. Levin. 290 pages.

[22] J. Cohen. Bioinformatics—an introduction for computer scientists. *ACM Computing Surveys*, 36(2):122–158, June 2004.

[23] F. Corpet and B. Michot. RNAlign program: alignment of RNA sequences using both primary and secondary structures. *Comput Appl Biosci*, 10(4):389–99, 1994.

[24] P. Crescenzi, D. Goldman, C. Papadimitrou, A. Piccolboni, , and M. Yannakakis. On the complexity of protein folding. In *Proceedings of STOC 1998*, pages 597–603, 1998.

[25] F. H. C. Crick. On protein synthesis. *Symposium of the Society of Experimental Biology*, 12:138–167, 1958.

[26] Y. Deville, D. Gilbert, J. van Helden, and S. Wodak. An overview of data models for the analysis of biochemical pathways. *Briefings in Bioinformatics*, 4(3):246–259, 2003.

[27] D. Diaz and P. Codognet. A Minimal Extension of the WAM for clp(FD). In D. S. Warren, editor, *Proceedings of the Tenth International Conference on Logic Programming*, pages 774–790, Budapest, Hungary, 1993. The MIT Press.

[28] K. A. Dill, K. M. Fiebig, and H. S. Chan. Cooperativity in protein-folding kinetics. *Proc. Natl. Acad. Sci. USA*, 90:1942 – 1946, 1993.

[29] G. Dooms, Y. Deville, and P. Dupont. Constrained path finding in biochemical networks. In *Proceedings of JOBIM 2004*, 2004.

[30] G. Dooms, Y. Deville, and P. Dupont. A mozart implementation of CP(bionet). In *Proceedings of the second International Mozart/Oz Conference*, pages 237–250. Springer-Verlag LNAI 3389, 2004.

[31] G. Dooms, Y. Deville, and P. Dupont. CP(graph): Introducing a graph computation domain in constraint programming. In *Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming*, pages 211–225, 2005.

[32] M. Dsouza, N. Larsen, and R. Overbeek. Searching for patterns in genomic data. *Trends in Genetics*, 13(12):497–498, 1997.

[33] R. Durbin, S. Eddy, A. Krough, and G. Mitchison. *Biological Sequence and Analysis*. CUP, 1998.

[34] I. Eidhammer, D. Gilbert, I. Jonassen, M. Ratnayake, and S. H. Grindhaug. A constraint based structure description language for biosequences. *Constraints*, 6 (2–3):141–156, 2001.

[35] I. Eidhammer, I. Jonassen, and W. R. Taylor. Structure Comparison and Structure Patterns. Technical Report 174, Department of Informatics, University of Bergen, Bergen, Norway, Jul 1999.

[36] W. H. Elliott and D. C. Elliott. *Biochemistry and Molecular Biology*. OUP, 1997.

[37] J. Feldenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39:783–791, 1985.

[38] W. M. Fitch. Toward defining the course of evolution: minimum change for a specified tree topology. *Systematic Zoology*, 20:406–416, 1971.

[39] S. Foissac and T. Schiex. Integrating alternative splicing detection into gene prediction. *BMC Bioinformatics*, 6(1):25, 2005.

[40] D. Gautheret, F. Major, and R. Cedergren. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *Computer Applications in the Biosciences*, 6:325–331, 1990.

[41] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. *Proc. Natl. Acad. Sci. USA*, 93(17):9061–6, 1996.

[42] I. P. Gent, P. Prosser, B. M. Smith, and W. Wei. Supertree construction with constraint programming. In *ICCP: International Conference on Constraint Programming (CP), LNCS*, pages 837–841, 2003.

[43] Z. Ghahramani. Learning dynamic Bayesian networks. In C. Lee Giles and Marco Gori, editors, *Adaptive Processing of Sequences and Data Structures*, number 1387 in Lecture Notes in Artificial Intelligence, LNAI, pages 168–197. Springer-Verlag, 1998.

[44] D. Gilbert, D. Westhead, J. Thornton, and J. Viksna. Tops cartoons: formalisation, searching and comparison. *RECOMB99 (poster)*, 1999.

[45] D. Gilbert, D. Westhead, and J. Viksna. Techniques for comparison, pattern matching and pattern discovery: From sequences to protein topology. In *Artificial Intelligence and Heuristic Methods in Bioinformatics*, pages 128–147. IOS Press, 2003.

[46] D. Gilbert, D. Westhead, J. Viksna, and J. Thornton. Topology-based protein structure comparison using a pattern discovery technique. *Journal of Computers and Chemistry*, 26(1):23–30, 2001.

[47] D. R. Gilbert, D. R. Westhead, N. Nagano, and J. M. Thornton. Motif–based searching in tops protein topology databases. *Bioinformatics*, 15(4):317–326, 1999.

[48] D. S. Greenberg and S. Istrail. Physical mapping by sts-hybradisation: Algorithmic strategies and the challenge of software evaluation. *Journal of Computational Biology*, 2(2):219–273, 1995.

[49] B. J. Haas, A. L. Delcher, S. M. Mount, J. R. Wortman, R. K. J. Smith, L. I. Hannick, R. Maiti, C. M. Ronning, D. B. Rusch, C. D. Town, S. L. Salzberg, and O. White. Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies. *Nucleic Acids Research*, 31(19):5654–66, 2003.

[50] R. M. Hall and C. M. Collis. Mobile gene cassettes and integrons: capture and spread of genes by site-specific recombination. *Mol Microbiol*, 15(4):593–600, 1995.

[51] W. E. Hart. On the computational complexity of sequence design problems. In *Proc. of the First Annual International Conferences on Compututational Molecular*

Biology (RECOMB97), pages 128–136, Santa Fe, New Mexico, 1997.

[52] F. U. Hartl and J. Martin. Molecular chaperones in cellular protein folding. *Current Opinion in Structural Biology*, 5(92):92–102, 1995.

[53] C. Helgesen and P. Sibbald. PALM - a pattern language for molecular biology. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proceedings First International Conference on Intelligent Systems for Molecular Biology*, pages 172–180. AAAI Press, 1993.

[54] M. Hiller, K. Huse, M. Platzer, and R. Backofen. Creation and disruption of protein features by alternative splicing – a novel mechanism to modulate function. *Genome Biol*, 6(7):R58, 2005.

[55] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 2004.

[56] A. Horwich and J. Weissman. Deadly conformations: Protein misfolding in prion disease. *Cell*, 89:499–510, 1997.

[57] T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–88, 2002.

[58] I. Jonassen, I. Eidhammer, and W. R. Taylor. Discovery of local packing motifs in protein structures. *Proteins*, 34(2):206–219, 1999.

[59] N. C. Jones and P. A. Pevzner. *An Introduction to Bioinformatics Algorithms (Computational Molecular Biology)*. The MIT Press, 2004.

[60] M. Kanehisa. Grand challenges in bioinformatics. *Bioinformatics*, 14(4):309, 1998.

[61] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, 28:27–30, 2000.

[62] J. Kececioglu, H.-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, and M. Vingron. A polyhedral approach to sequence alignment problems. *Discrete Applied Mathematics*, 104(1-3):143–186, 2000.

[63] J. D. Kececioglu. *Exact and Approximation Algorithms for DNA Sequence Reconstruction*. PhD thesis, University of Arizona, 1991.

[64] J. D. Kececioglu. The maximum weight trace problem in multiple sequence alignment. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Proc. 4th Symp. Combinatorical Pattern Matching*, pages 106–119, 1993.

[65] H. Kitano. Looking beyond the details: a rise in system-oriented approaches in genetics and molecular biology. *Current Genetics*, 41(1):1–10, 2002.

[66] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *Journal of Computational Biology*, 3(2): 289–306, 1996.

[67] L. Krippahl and P. Barahona. Psico: Solving protein structures with constraint programming and optimization. *Constraints*, 7(4-3):317–331, 2002.

[68] M. Krummenacker, S. Paley, L. Mueller, T. Yan, and P. D. Karp. Querying and computing with BioCyc databases. *Bioinformatics*, 21(16):3454–3455, 2005.

[69] R. H. Lathrop and T. F. Smith. Global optimum protein threading with gapped alignment and empirical pair score functions. *Journal of Molecular Biology*, 255: 641–665, 1996.

[70] K. F. Lau and K. A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22:3986 – 3997, 1989.

[71] F. Lefebvre. A grammar-based unification of several alignment and folding algorithms. In David J. States, Pamkaj Agarwal, Terry Gaasterland, Lawrence Hunter,

and Randall Smith, editors, *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 143–154, Menlo Park, June 12–15 1996. AAAI Press.

[72] C. Lemer, E. Antezana, F. Couche, F. Fays, X. Santolaria, R. Janky, Y. Deville, J. Richelle, and S. Wodak. The aMAZE LightBench: a web interface to a relational database of cellular processes. *Nucleic Acids Res.*, 32:D443–D448, 2004.

[73] H. P. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. *Journal of Computational Biology*, 5(3):517–30, 1998.

[74] R. J. Lipshutz, S. P. Fodor, T. R. Gingeras, and D. J. Lockhart. High density synthetic oligonucleotide arrays. *Nature Genetics*, 21:20 – 24, 1999.

[75] T. Macke, D. Ecker, R. Gutell, D. Gautheret, D. Case, and R. Sampath. RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Research*, 29(22):4724–4735, 2001.

[76] J. J. McGregor. Relational consistency algorithms and their application in finding subgraph and graph isomorphisms. *Information Sciences*, 19:229–250, 1979.

[77] I. Michalopoulos, G. M. Torrance, D. R. Gilbert, and D. R. Westhead. Tops: an enhanced database of protein structural topology. *Nucleic Acids Research, Database issue*, 32:D251–D254, 2003.

[78] Y. Deville P. D. G. Dooms. Constrained metabolic network analysis: discovering pathways using CP(Graph). In *Workshop on Constraint Based Methods for Bioinformatics*, pages 29–35, 2005.

[79] A. D. Palù, A. Dovier, and E. Pontelli. A new constraint solver for 3d lattices and its application to the protein folding problem. In Geoff Sutcliffe and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 12th International Conference, LPAR 2005, Montego Bay, Jamaica, December 2-6, 2005, Proceedings*, volume 3835 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 2005.

[80] A. Dal Palu, A. Dovier, and F. Fogolari. Constraint Logic Programming approach to protein structure prediction. *BMC Bioinformatics*, 5(1):186, 2004.

[81] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(Suppl 1):S215–224, 2001.

[82] A. Policriti, N. Vitacolonna, M. Morgante, and A. Zuccolo. Structured motifs search. *Journal of Computational Biology*, 12(8):1065–1082, 2005.

[83] S. D. Prestwich, D. G. Higgins, and O. O'Sullivan. A sat-based approach to multiple sequence alignment. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 940–944. Springer, 2003.

[84] K. Reinert, H.-P. Lenhof, P.Mutzel, K. Melhorn, and J. Kececioglu. A branch-and-cut algorithm for multiple sequence alignment. In *Proc. of the First Annual International Conferences on Compututational Molecular Biology (RECOMB97)*, pages 241–249, Santa Fe, New Mexico, 1997.

[85] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences. *Bioinformatics*, 14(1):55–67, 1998.

[86] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45(5):810–825, 1985.

[87] D. Searls. The computational linguistics of biological sequences. In Lawrence

Hunter, editor, *Artificial Intelligence and Molecular Biology*, chapter 2, pages 47–120. AAAI/MIT Press, 1993.

[88] D. Searls. String variable grammar: A logic grammar formalism for the bioligical language of DNA. *Journal of Logic Programming*, 24(1–2):73–102, July/August 1995.

[89] D. Searls and S. Dong. A syntactic pattern recognition system for DNA sequences. In C. R. Cantor H. A. Lim, J. Fickett and R. J. Robbins, editors, *Proceedings Second International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, pages 89–101. World Scientific, 1993.

[90] P. R. Sibbald and P. Argos. Scrutineer: a computer program that flexibly seeks and describes motifs and profiles in protein sequences databases. *Computer Applications in the Biosciences*, 6(3):279–288, 1990.

[91] P. R. Sibbald, H. Sommerfeldt, and P. Argos. Overseer: a nucleotide sequence searching tool. *Computer Applications in the Biosciences*, 8(1):45–48, 1992.

[92] S. Siebert and R. Backofen. MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, 21 (16):3352–9, 2005.

[93] T. Smith and M. Waterman. Comparison of biosequences. *Adv. appl. Math.*, 2: 482–489, 1981.

[94] G. 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.

[95] L. Soinov, M. Krestyaninova, and A. Brazma. Towards reconstruction of gene networks from expression data by supervised learning. *Genome Biology*, 4(1):R6, 2003.

[96] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438, 1958.

[97] R. Staden. Searching for Patterns in Protein and Nucleic Acid Sequencies. In R. F. Doolittle, editor, *Methods in Enzymology, Vol. 183*, pages 193–211. Academic Press, 1990.

[98] M. Steel2 and D. Penny. Parsimony, likelihood, and the role of models in molecular phylogenetics. *Molecular Biology and Evolution*, 17:839–850, 2000.

[99] M. J. Sternberg, P. A. Bates, L. A. Kelley, and R. M. MacCallum. Progress in protein structure prediction: assessment of CASP3. *Curr Opin Struct Biol*, 9(3):368–373, 1999.

[100] P. Thebault, S. de Givry, T. Schiex, and C. Gaspin. Combining constraint processing and pattern matching to describe and locate structured motifs in genomic sequences. In Christian Bessiere, Brahim Hnich, Toby Walsh, and Zeynep Kiziltan, editors, *The Fifth Workshop on Modelling and Solving Problems with Constraints*, pages 53–60, 2005.

[101] G. M. Torrance, D. R. Gilbert, I. Michalopoulos, and D. R. Westhead. Protein structure topological comparison, discovery and matching service. *Bioinformatics*, 21(10):2537–2538, 2005.

[102] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23 (1):31–42, January 1976.

[103] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231:75–81, 1993.

[104] J. van Helden, D. Gilbert, L. Wernisch, M. Schroeder, and S. Wodak. Application of regulatory sequence analysis and metabolic network analysis to the interpretation of gene expression data. In *Computational Biology, LNCS 2006*, pages 147 – 163. LNCS, 2000.

[105] J. van Helden, A. Naim, R. Mancuso, M. Eldridge, L. Wernisch, D. Gilbert, , and S. J. Wodak. Representing and analysing molecular and cellular function in the computer. *Journal of Biological Chemistry*, 381(9–10):921–935, 2000.

[106] J. Viksna and D. Gilbert. Pattern matching and pattern discovery algorithms for protein topologies. In *WABI: International Workshop on Algorithms in Bioinformatics, WABI, LNCS 2149*, pages 98–111, 2001.

[107] M. Waterman. *Introduction to Computational Biology*. Chapman & Hall, London, 1995.

[108] S. 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.

[109] R. H. C. Yap. Parametric sequence alignment with constraints. *Constraints*, 6(2/3): 157–172, 2001.

[110] K. Yue and K. A. Dill. Sequence-structure relationships in proteins and copolymers. *Physical Review E*, 48(3):2267–2278, September 1993.

[111] K. Yue and K. A. Dill. Forces of tertiary structural organization in globular proteins. *Proc. Natl. Acad. Sci. USA*, 92:146 – 150, 1995.

[112] M. Zuker. On Finding All Foldings of an RNA Molecule. *Science*, 244:48–52, 1989.