TEL AVIV UNIVERSITY
RAYMOND AND BEVERLY SACKLER
FACULTY OF EXACT SCIENCES
SCHOOL OF PHYSICS & ASTRONOMY

אוניברסיטת תל-אביב
הפקולטה למדעים מדוייקים
ע"ש ריימונד ובברלי סאקלר
בית הספר לפיסיקה ואסטרונומיה

Classification and Clustering of Protein Structures

by

Gaddy Getz

Thesis submitted in partial
fulfillment of the requirements
for the M.Sc. degree
at Tel-Aviv University
School of Physics and Astronomy

The research work for this thesis has been
carried out under the supervision of
Professor Eytan Domany (Weizmann Institute of Science) and
Professor Amnon Aharony (Tel-Aviv University)

September 1998

Acknowledgments

I wish to thank Professor Eytan Domany for his devoted guidance, his enlightening discussions and endless patience made this work possible. I also wish to thank Professor Amnon Aharony for his support throughout this work. I am grateful to Dr. Michele Vendruscolo, Dr. Shai Weismann and Noam Shental for fruitful discussions.
My deep gratitude goes to my family and friends for their encouragement.

Contents

Abstract

We predict the CATH architecture (Orengo *et al.* 1997) of 165 single–domain proteins that were not yet processed by CATH, using the FSSP Z-scores (Holm and Sander 1996). The architecture assignment in the CATH database is performed manually, while the FSSP Z-scores are computed automatically. We predict the architecture by applying standard and newly proposed automatic classification algorithms to the Z-scores; therefore, our procedure can save human effort in the enhancement of CATH.

In order to perform the classification we introduce improved similarity measures between the proteins; direct measures are derived using local normalization of the original similarities, and indirect measures are based on the outcome of hierarchical clustering methods. Finally, we suggest a new classification scheme which uses relationships between the unclassified proteins as part of the classification.

Chapter 1

Overview

Proteins are chain-like molecules that fold into different three dimensional structures. The biological function of a protein is determined mainly by its structure. Therefore, the structure is a very important characteristic of a protein and is widely studied. Many researchers classify proteins into groups of similar structures, using different techniques to organize the structures in groups. Some researchers use automatic structure comparison algorithms, while others view the structures and use human judgment based on experience and visual criteria to decide how to group the proteins.

The goal of this work is to use a pairwise structural similarity measure (FSSP score) calculated by one group of researchers to *automatically* predict an aspect of classification that is determined *manually* by another group (CATH). Such a procedure saves human intervention and helps to increase the number of classified proteins easily.

Usually, machine learning and pattern recognition techniques, such as nearest–neighbor classification, are used to give automatic predictions of classification. In this work we tested several classification algorithms to perform the automatic class prediction. At first, a nearest–neighbor classification algorithm was chosen. The performance of this algorithm depends on the similarity measure determined for the proteins. Hence, in order to increase the prediction ability of the algorithm, we propose and evaluate several similarity measures derived from the FSSP distances. A set of robust similarity measures are generated by first utilizing a hierarchical clustering algorithm that groups the

proteins into families of similar structures and, then, defining a new similarity measure using the produced hierarchy. We tested several clustering algorithms for this task. For comparison, the clustering outcome was used in a more direct manner to perform the classification. Finally, we suggest a new heuristic classification procedure combining clutering and classification features in one algorithm.

The success rate of the proposed algorithms is estimated by testing them on a set of classified proteins. Finally, we use our methods to classify a set of yet unclassified proteins.

1.1 Thesis Outline

The thesis consists of five chapters. The second Chapter, the introduction, includes a background on protein structure and protein classification and presents the different databases from which the data for this work were taken. It also describes the motivation for the work. The third Chapter gives a formal definition of the problem. It presents the input data and the desired output, together with the description of the evaluation procedure. Chapter 4 deals with the classification methods, presenting the algorithms used and similarity measures that were evaluated. We proposed several similarity measures; direct ones, that are locally derived form the "raw" similarity measures, as well as indirect ones that use clustering as a preprocessing stage to calculate a new set of similarities.

A summary of the prediction ability of all the methods, as tested on classified proteins, and our resulting predicted class for the yet unclassified proteins are given in Chapter 5.

The main new results of the thesis on protein structure prediction are summarized in tables 4.2 and 5.2 through 5.6. The first of these lists several proteins that we believe have been misclassified in the CATH database. The other tables contain our predicted classification for 165 proteins that have not yet been considered by CATH.

Chapter 2

Introduction

This Chapter contains a general background on protein structure. It includes, in the first section, a brief summary of protein structures and a description of protein structure studies that are relevant to this work. In the second section the motivation for this work is given. Terms and nomenclature used throughout the work are explained and appear in boldface.

2.1 Background on Proteins

2.1.1 Protein structure

Since the first three-dimensional structure of a protein was discovered four decades ago, a lot of research has been done on protein structures. Today it is well documented and agreed [7] that a protein's functional properties and evolutionary relations depend mainly on its three dimensional structure. In spite of many years of research, many questions about protein structures remain unsolved.

A protein consists of one or several **poly-peptide chains** of amino acids that act together to perform some biological task. There are 20 types of amino acids, all having a common central carbon atom (C_α) to which a hydrogen atom, an amino group (NH_2), and a carboxy group (COOH) are attached. The difference between amino acids is the side chain which is attached to the fourth C_α bond. When a protein is synthesized, the amino acids are joined by connecting the amino group of one acid to the carboxy group

of the next, while emitting a water molecule (creating a peptide bond). One can assign a direction to the amino acid chain since it has two distinct ends (amino and carboxy). By convention, the chain is said to run from the amino end to the carboxy end. The sequence of a protein, also called its *primary structure*, is encoded in the DNA. A human DNA, for example, encodes about 100,000 different proteins. Many of these sequences and also non-human ones are already known [16].

The different three-dimensional structures of proteins arise from the different interactions between parts of the protein with each other and with their surrounding. There are several kinds of interactions: between side chains and the water around them, interactions between two side chains (along the same poly-peptide chain or from other close chains) and interactions between atoms on the main chain. The protein uses the rotational degrees of freedom of the bonds along the main chain to fold into the lowest energy configuration. The energy of a configuration depends on all the interactions mentioned above.

Most of the three dimensional structure of a protein, especially its core, is stable and can be viewed as a function of the sequence alone. The key problem of molecular biology is to be able to predict the three dimensional structure and, therefore, functional properties of the protein from its amino acid sequence. This problem is usually called the *folding problem.*

A protein's structure can be described using a bottom-up approach; starting from the "primary structure" (the amino acid sequence) up to the entire three dimensional structure of the whole protein. Different parts of the protein form a local regular structure, such as α-helices or β-strands; these are called *secondary structures.* Most protein structures are a combination of **secondary structure elements** connected by loop regions of various lengths and irregular shape. Figure 2.1 shows a three–dimensional structure of a protein. Secondary structures combine to form higher level structures. β-strands usually combine to form a β-sheet, which is several β-strands aligned next to each other to form a sheet

in space. Note that similar three dimensional shapes can be formed by many different connectivity orders and directions of the secondary structure elements that form them. For example, a β-sheet can be formed by combining parallel or anti-parallel β-strands.

The secondary structures of a protein usually combine to form one or several globular structures which are called domains or *tertiary structures*. A **domain** is defined as a poly-peptide chain or part of a chain that can independently fold into a stable structure. Homologous amino acid sequences, *i.e.* similar sequences that are usually evolutionary related, form similar tertiary structures in different proteins. The domains also have a functional role. Often, each domain in a protein is associated with a different biological function. Due to the domain's structural independence, many research groups study and classify domain structures rather than whole proteins.

Most domain structures can be divided into three classes, according to the secondary structures found in their core: **mainly-α**, **mainly-β** and **α-β** domains. The remaining domains are built of few isolated α-helices and β-strands and can be considered as a small fourth class.

Many proteins are made of a single poly-peptide chain. There are, however, cases where several chains, often replicas of the same one, combine to form a single functional protein. The structure of these multi-chained proteins is described by the *quaternary structure*, which is the highest level model of protein structure.

2.1.2 Structure research

Three dimensional structures of proteins are collected and stored in a computerized archive called the Brookhaven Protein Data Bank (**PDB**) [3][1]. The number of known structures is constantly increasing and passed 9000 at the end of 1997 (the latest version from August 13, 1998 includes 11912 protein structures) [27]. The collection and study of protein structures can help the pursuit of a practical solution to the folding problem. Since evolution has created many proteins with practically identical structures and similar sequences, one can infer the structure and function of a protein of known sequence, provided a close

Figure 2.1: Three dimensional structure of the protein Deoxyribonuclease I (PDB code 3DNI). On the left, a spacefill view of the protein. On the right, a cartoon plot showing α-helices and β-strands as ribbons, and loops as strings. This shape is called $\alpha\beta$ 4-layer sandwich because it has 4 layers, the outer ones are made of α-helices and the inner ones are β-sheets.

sequence neighbor with known characteristics can be identified [17].

During evolution, a protein responsible for a specific biological task tends to keep its three dimensional structure over much longer periods than it maintains its sequence. Consequently, two proteins that have more than 30% sequence identity (*i.e.* close in evolutionary distance) have almost certainly the same fold [26]. However, there are many cases in which proteins with very low sequence identity (that share a distant common ancestor) do have the same fold [16]. Thus, studying the space of all protein folds (fold space) can teach more about evolutionary and functional relations than studying sequence resemblance alone.

Several research groups are studying the structural relationships between proteins, trying to find out whether certain structures are more frequent than others and what is the relationships and correlation to the protein function. Most of the groups divide the fold space into structural hierarchies (nested groups) describing different levels of similarity [15][26][22][30]. Many of these groups make their classification and identified

structural relationships publicly available in their web sites (see Appendix C for a list of all web sites referred to in this work).

Each research group has its own way to compare and group the proteins. Holm and Sander use a fully automatic structure comparison algorithm, DALI (Distances ALIgnment algorithm), to calculate a pairwise similarity measure (S-score) between protein *chains*; the more similar are two chains, the higher the similarity measure assigned to them. Holm and Sander maintain a database of fold classifications, named **FSSP** (**F**old classification based on **S**tructure-**S**tructure alignment[1] of **P**roteins), based on calculating a structural similarity measure for all pairs in a representative sub-set of the PDB [15][18]. The result of this all-against-all comparison is reported in the form of a fold tree generated by a hierarchical clustering algorithm (the algorithm is described in 4.5.5). Using the fold tree one can find proteins with similar structure at varying degrees of statistical significance.

The similarity measure, **S-score**, between protein chains i and j, denoted as S_{ij}, is calculated by comparing the structures of aligned sub-chains in the two proteins. The sub-chain alignment used in the calculation is selected so that the outcome S-score is maximal. The S-scores, $\{S_{ij}\}$, are then normalized, to give statistical meaning, by shifting and scaling them according to their average value and standard deviation. The scaled values are called **Z-scores**, $\{Z_{ij}\}$. A more detailed description of the FSSP database and similarity measure is given in Appendix A.

The group of Orengo *et al.* uses a combination of automatic and manual procedures to create a hierarchical classification of *domains* (**CATH**) [26]. They arrange the domains in a four level hierarchy of families according to the protein class (**C**), architecture (**A**), topology (**T**) and homologous superfamily (**H**). The hierarchy of CATH is demonstrated in figure 2.2. In the figure all the class types and most of the architecture types are listed,

[1] A structure-structure alignment is a correspondence between amino-acids of two proteins. Corresponding amino-acids are positioned in similar geometrical places relative to the whole protein three-dimensional structure.

but in the topology and homologous superfamily levels only representatives appears.

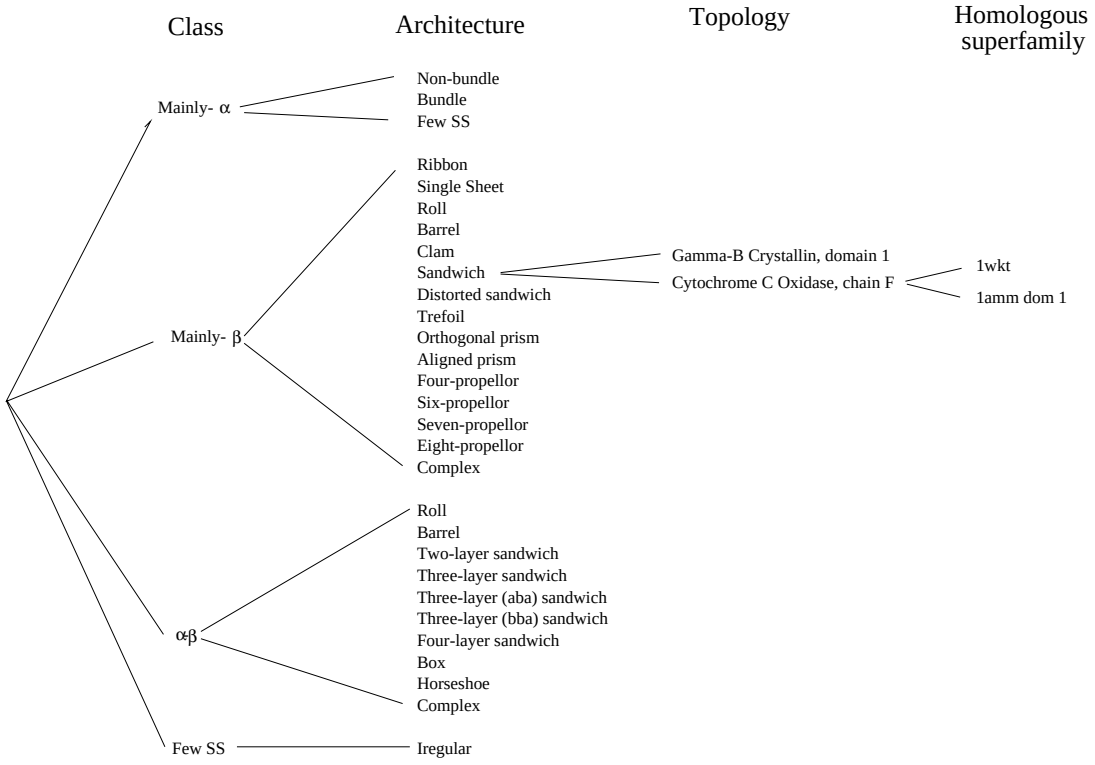


Figure 2.2: CATH hierarchical classification tree. Only representative types are listed.

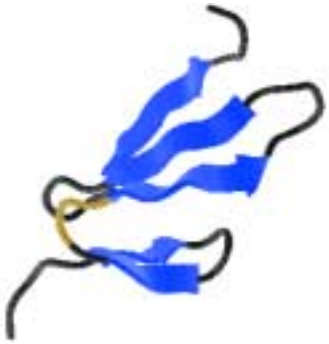
The class level describes the secondary structures found in the domain and is created *automatically*. The architecture level, on the other hand, is assigned *manually* (using human judgement) and describes the shape created by the relative orientation of the secondary structure units. The shape families are chosen according to a commonly used structure classification of Richardson [28] and other reported shapes, like barrel, sandwich, roll, etc. Figure 2.3 depicts the shapes of representatives from 6 frequent architecture families.

The topology level groups together all structures with similar sequential connectivity between their secondary structure elements. For example, two similar shapes that include a β-sheet with different relative direction of its β-strands will be assigned to different topology groups. Structures with high structural and functional similarity are put in

(a) Mainly-α, Non-bundle

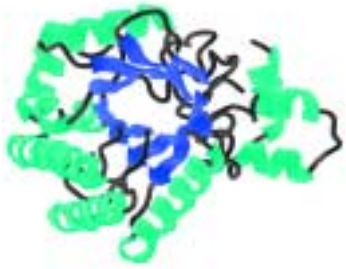
(b) Mainly-α, Bundle

(c) Mainly-β, Sandwich



(d) Mainly-β, Ribbon

(e) Mainly-β, Barrel



(f) α-β, Barrel

Figure 2.3: Cartoon plots of representatives from 6 frequent architecture types. The figures were taken from the CATH web site (see Appendix C).

the same fourth level family, called homologous superfamily. Such structures may have evolved from a common ancestor. Both the topology and homologous superfamily levels are assigned by thresholding a structural similarity measure (SSAP) at two different levels, respectively [32][24]. This measure is calculated *automatically* between the proteins. Assigning two domains to the same homologous superfamily requires a similar function in addition to high structural resemblance. An elaboration on the CATH database is presented in Appendix B.

The task of defining structural relationships between proteins is further complicated by the existence of multi-domain proteins. It is not obvious what similarity value should be assigned between a single-domain protein that is similar to one of the domains of a multi-domain protein. Several research groups, including Orengo *et al.* , tackle this problem by first separating the proteins into domains and then studying the structural resemblance between domains. There are several algorithms for domain identification. The domains identified in different proteins are also publicly available on the web in several databases (see Appendix C). In this work we used the 3Dee database of Siddiqui and Barton [36][30].

2.2 Motivation

The motivation for this work originated from a paper published by Holm and Sander [15] describing the FSSP database and the similarity measure used to create it. They showed that by using the similarity scores from an all-against-all comparison, one can separate the protein folds into groups that correspond to known structural families. On the other hand, the CATH database, which also classifies proteins into structural families, includes a *manual* procedure in order to identify the protein architecture. This gave rise to the question whether **the FSSP similarity measure can be used to give an automatic prediction of the CATH architecture.** Such a prediction can be used to enlarge the CATH database without the necessary manual intervention.

Furthermore, the FSSP database uses an average linkage clustering algorithm to ob-

tain its hierarchy of fold families. Recently, a robust clustering algorithm, based on the behavior of inhomogeneous magnets (SPC), was proposed by Blatt et at. [4]. Naturally, the next question is whether **the SPC algorithm can be applied to the FSSP data and how it compares to the average linkage algorithm** in identifying structurally related families.

Therefore, the goal of this work was set to address the above problems by proposing a robust automatic algorithm that can predict the CATH architecture for chains that are included in the FSSP database but have not yet been assigned a CATH classification.

Chapter 3

Problem definition

In this chapter we define the problem we want to solve and the criteria used to evaluate the alternative solutions. As mentioned previously (Section 2.2), we want to automatically predict the architecture classification of CATH, using the similarity measure of FSSP.

To deal with the architecture prediction using a statistical pattern recognition framework, we have to define the input data and desired output. The input data is a set of n protein entries, $\{E_i\}_{i=1}^{n}$, that appear in both FSSP and CATH. From the FSSP we obtain the pairwise similarity measures for all protein pairs, S_{ij} and Z_{ij}, and from CATH we take the classification data, $C_i = \{C_i^{\mathrm{C}}, C_i^{\mathrm{A}}, \ldots\}$. These data sets are used to build and test the classification algorithms. We seek an algorithm that predicts the correct architecture with the highest probability.

Before getting into details, it is important to state that the above data, together with the evaluation method, are all that is needed to properly define the problem of finding a good classification algorithm. In addition, we collected from other proteins databases a set of entries that were *not processed* by CATH, for which we can predict the yet undetermined architecture. Following is a more detailed description of how we assembled the protein data sets and the method used to evaluate the classification algorithms.

3.1 Getting the data

The input data used in this work are the FSSP S and Z scores. The S and Z matrices of the FSSP version from December 25, 1997 were obtained from Liisa Holm (one of the FSSP creators). A Z-score matrix that contains Z values with $Z > 2.0$, can be obtained directly from the FSSP web site [10]. The December 1997 version of the FSSP database included 1188 protein chains which represent 9153 PDB structures (see appendix A). The lengths of all FSSP chains were extracted from the FSSP files. The length, which is not part of the defined problem, is used for an initial analysis of the data. We denote the lengths vector as $L = \{l_i\}_{i=1}^{N}$.

The next step is to find which of the FSSP chains appears in CATH. Due to the fact that CATH handles domains whereas FSSP deals with chains, we decided to use only chains which have a single domain and, therefore, will appear as a single entry in both databases. The list of 1188 FSSP chain names was checked against the CATH database. Out of the 686 proteins that appear in CATH, 479 were single domains and had a single classification in CATH, whereas the remaining 207 were multi-domained and therefore had several classifications, one for each of their domains.

We retrieved and collected all the CATH classifications of the 479 single-domain chains. Not all of the CATH families had representatives in the 479 sample; especially small families at the topology and homologous superfamily levels were missing. Therefore, we renumbered the obtained class types of the chains in every CATH hierarchy level. In the first level, the class level, there were representatives from all four class types (numbered 1 to 4). In the second, architecture level, the 479 chains covered 29 different architecture types. Table 3.1 lists the 29 architecture families grouped according to their class.

The indices of the families at all CATH levels of the proteins are denoted by $C_i = \{C_i^{\mathrm{C}}, C_i^{\mathrm{A}}, \ldots\}$ where $C_i^{\mathrm{C}} = 1, \ldots, 4$ and $C_i^{\mathrm{A}} = 1, \ldots, 29$ and so on.

We define the dataset **PR479** as the list of these 479 single-domain chain entries, $\mathbf{E}$, with their corresponding FSSP similarity matrices $\mathbf{S}$ and $\mathbf{Z}$, their lengths vector L, and

#	C	Class	C.A	Architecture	Number
1	1	Mainly α	1.10	Non-bundle	72
2			1.20	Bundle	29
3			1.30	Few SS	24
4	2	Mainly β	2.10	Ribbon	23
5			2.20	Single Sheet	4
6			2.30	Roll	10
7			2.40	Barrel	23
8			2.50	Clam	1
9			2.60	Sandwich	52
10			2.70	Distorted sandwich	6
11			2.80	Trefoil	4
12			2.90	Orthogonal prism	1
13			2.100	Aligned prism	1
14			2.110	Four-propellor	2
15			2.120	Six-propellor	2
16			2.130	Seven-propellor	1
17			2.140	Eight-propellor	1
18			2.170	Complex	1
19	3	α-β	3.10	Roll	25
20			3.20	Barrel	26
21			3.30	Two-layer sandwich	64
22			3.40	Three-layer ($\alpha\beta\alpha$) sandwich	61
23			3.50	Three-layer ($\beta\beta\alpha$) sandwich	1
24			3.60	Four-layer sandwich	1
25			3.70	Box	1
26			3.80	Horseshoe	1
27			3.90	Complex	14
28			3.100	Few SS	13
29	4	Few SS	4.10	Irregular	15

Table 3.1: List of CATH architecture families that have representatives in PR479. The second and fourth columns are the original CATH classification of the architectures.

CATH classifications **C**.

$$\mathbf{PR479} = \{\mathbf{E}, \mathbf{S}, \mathbf{Z}, L, \mathbf{C}\} \ . \tag{3.1}$$

Of the 1188 FSSP entries, 503 chains were not processed by CATH. Some of these chains are single-domained and some are multi-domained. In order to identify the single-domain chains, these 503 chains were checked against a different database, the 3Dee database [36] (see Appendix C for the database web site), which includes an indication whether a chain is single or multi-domained. This way, 165 single-domain chains, which were not yet processed by CATH, were identified. These are candidates for new CATH entries, for which we predict the class and architecture levels. The dataset that contains these yet unclassified proteins is called **PR165**. The dataset of the combined list of protein entries is referred to as **PR644**.

To demonstrate the problem, consider figure 3.1, which shows the Z-score matrix between all proteins in PR644 (Z-scores larger than 2.0 are represented by black dots). The first 479 proteins constitute PR479, for which the CATH classification is known. Proteins numbered from 480 to 644 are PR165, for which we want to predict a classification. The order of the proteins within PR479 and PR165 is according to the original index in FSSP and has no particular meaning.

One can identify three parts in the Z-score matrix that are treated differently by classification algorithms; The Z-scores between PR479 proteins and themselves (upper left square sub-matrix); The Z-scores among the PR165 proteins (lower right square sub-matrix) and the Z-scores between PR479 and PR165 (the remaining rectangles). We will refer to these sub-matrices in the next chapter. The diagonal elements of the matrix, which represent the self-similarity of the proteins, are also important. Notice that, as expected, all the proteins are similar to themselves.

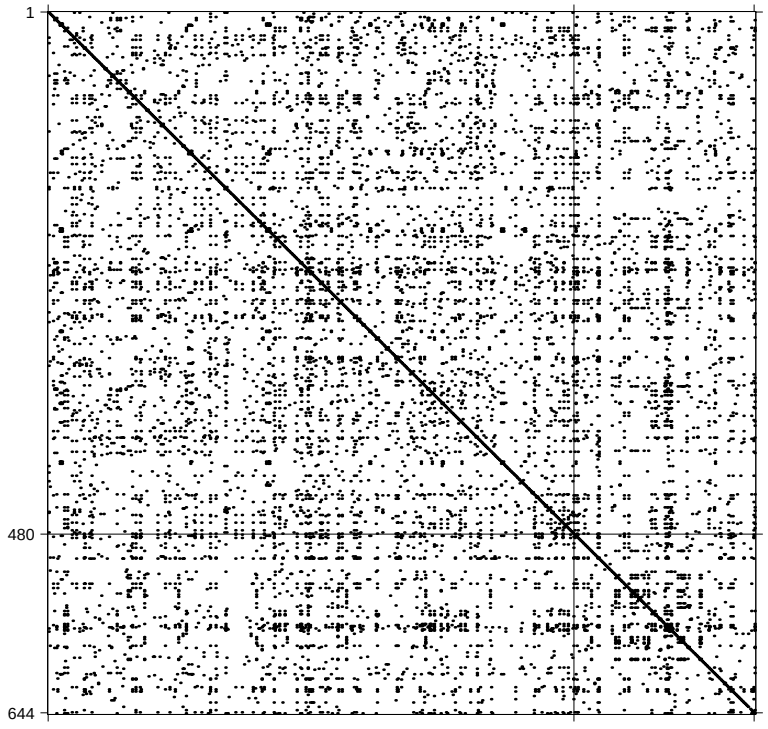


Figure 3.1: Z-score matrix between PR644 proteins. A black dot represents $Z > 2.0$. Larger Z corresponds to more similar proteins. For the first 479 proteins the CATH classification is known. For the last 165 proteins we predict the classification.

3.2 Problem Setting

This section describes the mathematical setting of the problem. We are seeking an algorithm which predicts the true classification of new proteins with high probability. Such classification problems are addressed by statistical pattern recognition, which gives the mathematical formulations and tools to deal with such problems [9][11].

Consider a set of n protein entries, $E = \{E_i\}_{i=1}^{n}$, and a set of c classes ω_i where $i = 1, \ldots, c$. For the first m proteins the classification is known and is denoted by $C = \{C_i\}_{i=1}^{m}$, where $C_i \in \{1, \ldots, c\}$ and indicates that $E_i \in \omega_{C_i}$. Note that in our problem each protein has several classifications, one for each level in the CATH hierarchy. Because we want to predict the architecture level of the protein, we define our classes according to architecture.

From the FSSP we obtain the pairwise similarity measures, $\mathbf{S} = \{S_{ij}\}_{i,j=1}^{n}$ and $\mathbf{Z} = \{Z_{ij}\}_{i,j=1}^{n}$. A classification algorithm, $\mathcal{A}$, can use all the known information to predict the class of the unclassified proteins entries $\{E_i\}_{i=m+1}^{n}$, i.e

$$\{\hat{C}_i\}_{i=m+1}^{n} = \mathcal{A}\left(\mathbf{Z}, \mathbf{S}, C\right) \tag{3.2}$$

where $\{\hat{C}_i\}_{i=m+1}^{n}$ represent predicted classification.

The statistical approach assumes that there is some probability function that governs the protein's classes and similarity measures and that the datasets (PR479, PR644) are samples drawn according to that probability [9][11]. It also states that in order to make the minimal number of misclassifications (on average), one should classify each protein of the set PR165 to the class to which it belongs with highest probability, given all the known information. This method is also known as maximum a-posteriori probability classification (MAP) or Bayes rule. For proteins $\{E_i\}_{i=m+1}^{n}$ we choose the predicted classes to satisfy the condition

$$\{\hat{C}_i\}_{i=m+1}^{n} = \arg\max_{C_i} \mathrm{P}\left(\{C_i\}_{i=m+1}^{n} \,|\mathbf{Z}, \mathbf{S}, C\right). \tag{3.3}$$

Note that, in general, the assigned classes are *not* independent and that the $\mathbf{S}$ and $\mathbf{Z}$

matrices include elements also between unclassified proteins.

In any classification problem, even the optimal classification algorithm has inherent errors since there are situations where the information given to the algorithm is not sufficient to make a decision. The minimum error rate is called the *Bayes error* and is obtained when using the Bayes rule (3.3) for classification.

3.3 Evaluating the prediction

In order to choose the best of all the suggested classification algorithms, we need to know how to evaluate an algorithm's quality. Assessment of the quality of the classification algorithm, like the classification itself, is studied in a statistical framework. One wishes to estimate P_{error}, the probability that the algorithm will misclassify new proteins, or

$$P_{success} = 1 - P_{error} = \frac{N_{success}}{N_{test}} \quad . \tag{3.4}$$

Usually, this is done by dividing the set of proteins with known classification into two subsets; one is used for *training* the algorithm, *i.e.* assign parameters in the algorithm according to the desired output, and the other set, of N_{test} proteins, is used to *test* the algorithm, by comparing its prediction to the true classification. Such a procedure imitates the real situation where one has a set of classified proteins (PR479) and tries to predict the class of a different set of proteins (PR165).

A more robust method to estimate the error is called *cross-validation* [31]. In this method one repeats the above procedure T times with different partitionings of the data. In each trial, the test set is randomly selected out of the data with known classification, using a certain train/test ratio. The error rate of the classification algorithm is then estimated by the average of the error rates of the individual trials. Formally,

$$\hat{P}_{error} = 1/T \sum_{t=1}^{T} P^t_{error} \ , \tag{3.5}$$

where $\hat{P}_{error}$ is the estimate for the error probability and P^t_{error} is the error for the individual trial t. Using cross-validation reduces the variance of the estimator and, there-

fore, gives a more robust estimate for the error rate of new predictions. Cross-validation schemes differ by the train/test ratio used and by the number of trials made.

An extreme case of cross-validation is the *leave-one-out* estimation method, in which for each data point a model is trained on all the data but itself and then the model's prediction for the left-out point is tested. The error rate is then estimated by the number of prediction errors divided by the number of data points,

$$\hat{P}_{error}^{\text{leave-one-out}} = \frac{N_{errors}}{N} \ . \tag{3.6}$$

The leave-one-out estimate gives, on average, an underestimation of the error rate [11]. Intuitively, this is so because the trained models are biased towards the left out point. If, for example, there is a distant set of proteins that for some reason are not present in the sample data, the trained models will perform much worse on them compared to a single point taken out from the sample. Such an example can occur if there is some technical difficulty, that is later overcome, to obtain the structures of proteins from a certain family.

Therefore we evaluate the error rate using cross-validation at various train/test ratios. For each of 11 different sizes we randomly selected 50 test sets out of **PR479**. The sizes taken were $n_i = 1$, 10, 20, 30, 40, 50, 60, 70, 80, 90, 95 percent out of 479. The prediction rate is plotted as a function of the test set percentage, which we call the *dilution* because the data set is diluted by removing the test set. Leave-one-out corresponds to a dilution of 1/479 averaged on the 479 different possibilities. For each prediction rate the standard deviation of the estimated error is plotted as an error bar, *i.e.* the standard deviation of the $T = 50$ cases divided by $\sqrt{50}$.

Figure 3.2 presents the success rate of a nearest-neighbor classification method, which classifies a protein according to its nearest classified one (the method is discussed in detail in the next Chapter). Note that the success rate decreases as the dilution is increased. This behavior is common because there is less data to train the model on and, therefore, less information to base the prediction on.

Classification algorithms do not always give a prediction; these cases are called *rejec-*

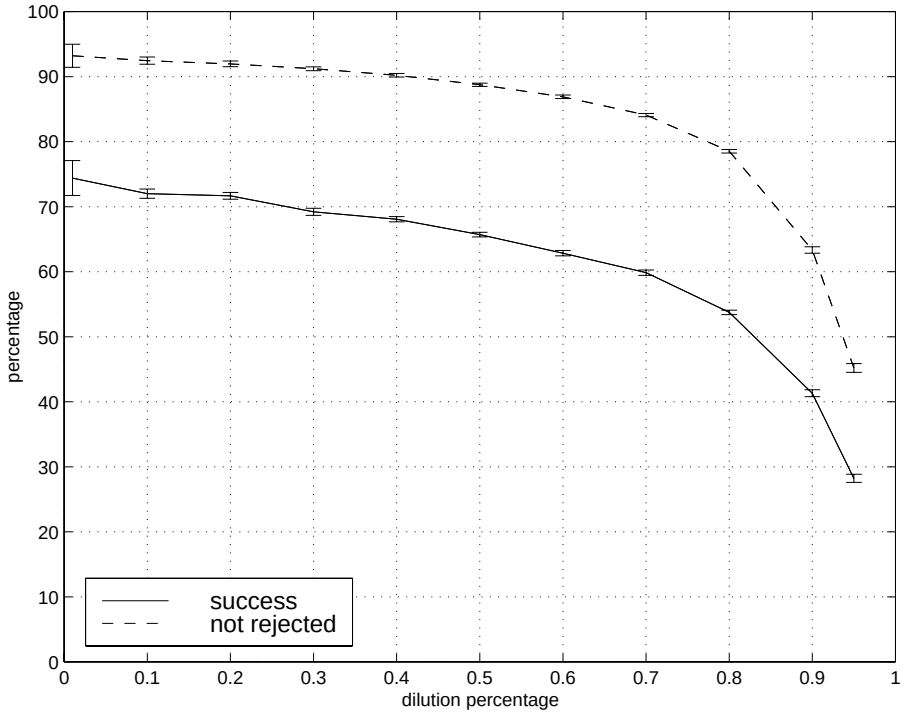


Figure 3.2: Success rate of a nearest-neighbor classification using Z-score as similarity measure. The rate is estimated using cross-validation of 50 random sets as a function of the dilution percentage. The non-rejection rate is plotted as a dashed line

tions. A common rejection case is when a nearest neighbor algorithm is asked to predict the class of a data point that has no close neighbors. Therefore, the rejection rate, P_{reject}, is also an important factor when evaluating a classification algorithm. The *purity* of an algorithm is defined as the probability of correct prediction out of the non-rejected ones,

$$P_{success} = \frac{N_{success}}{N_{test}} = \frac{N_{test} - N_{reject} - N_{error}}{N_{test}} \tag{3.7}$$

$$P_{non-reject} = 1 - P_{reject} = \frac{N_{test} - N_{reject}}{N_{test}} \tag{3.8}$$

$$P_{pure} = \frac{N_{success}}{N_{test} - N_{reject}} = \frac{P_{success}}{1 - P_{reject}} \; . \tag{3.9}$$

Figure 3.2 also shows the non-rejection rate as a function of the dilution. The purity is the ratio between the lower and upper lines. Naturally, the success rate is always less than the non-rejected rate because a rejection is not considered a success. In the case of nearest–neighbor classification, the rejection rate is increasing with the dilution because more data points are left without close classified neighbors.

Obviously, a classification algorithm with higher success rate is considered better. However, as will be seen in Chapter 4, some algorithms are better at low dilutions and others are better at high dilutions. If so, which algorithm should we use to predict the PR165 proteins? Our case of predicting 165 out of 644 corresponds to a dilution of 25%, but it is not guaranteed that if an algorithm predicts 25% out of the 479 proteins better it is also the best choice for predicting the additional 165 proteins. If, for example, the 165 proteins have few neighbors within the classified proteins, which corresponds to a high dilution, then it is preferable to use an algorithm which performes better at high dilutions. We will address these issues when our prediction of the 165 yet unclassified proteins is given in Chapter 5.

Chapter 4

Classification - methods and results

This Chapter describes the techniques tested in this work for the protein architecture classification. Each classification technique is evaluated using the methods described in the previous Chapter.

As a first step, we analyze the raw FSSP data on the proteins and check how they relate to the proteins' CATH classifications (Sec. 4.1). Then we describe our approach to slove the fully automatic classification problem (Sec. 4.2). We chose to use classification algorithms of the nearest–neighbor type using, first, the original similarity measures and then improve them by deriving new similarity measures based on the original ones. The classification algorithms we use are described in Section 4.3. In Section 4.4 we discuss and evaluate two types of similarity measures, direct (Sec. 4.4) and indirect (Sec. 4.5), which are used in combination with the classification algorithms. Finally, we describe, in Section 4.6, a newly suggested classification scheme that incorporates aspects of indirect similarities in the classification process.

The work described in this chapter is the basis for our decision with which classification methods to predict the architectures of the 165 yet unclassified proteins. The selection of classification method and it's prediction is reported in Chapter 5. As a byproduct of the FSSP and CATH analysis, we report (on page 34) a list of already classified proteins out of PR479 that may have been misclassified in the CATH database.

4.1 Initial analysis

First we performed an initial analysis of the data obtained from FSSP and CATH. The purpose of the analysis is to get a deeper understanding of the data and to look for correlations between the raw FSSP data and the CATH classifications. This analysis enables us to appreciate the complexity and solubility of the problem and help us choose the right approach to the solution.

4.1.1 Z and S properties

The S-score is defined as a positive score measuring the structural similarity between proteins. The S-score is symmetric with respect to the two proteins compared, therefore **S** is a symmetric matrix. It is useful to study separately the diagonal elements, which represent self-similarity of the proteins, and the off-diagonal ones which represent similarities between different proteins. Deeper analysis of the S-score is described in appendix A.

The S-matrix is very sparse, *i.e.* most if its elements are zero. The S matrix for PR479, for example, is a 479-by-479 matrix with 114481 ($479 * 478/2$) off-diagonal upper triangle elements ($S_{i>j}$) out of which only 3636 elements are non-zero (3.1%). A zero S-score means that the DALI algorithm, which is used to create the FSSP, could not find any part of the two compared protein that have a sufficiently similar structure.

The Z-score, which is the scaled version of the S-score, can in principle be positive or negative as it measures standard deviation from the mean. However, the Z-matrix, obtained from Holm, includes only positive elements located where the S-matrix elements are non-zero. Note that the Z-scores that can be obtained from the FSSP web site are all above 2.0, which is regarded as significant similarity.

One can view the **Z** or **S** matrices as representing a *weighted graph*, where each vertex in the graph is a protein and the weights on the edges are the Z (or S) scores between the proteins. Edges with zero (or less) weight are absent. Following this representation, we call proteins that are connected via an edge - *neighbors*. In other words, the neighbors of

protein entry E_i are all the proteins $\{E_j\}$ for which $Z_{ij} > 0$.

The sparsity of the similarity matrices makes the classification task more difficult since many proteins have few neighbors to base the prediction on. The average number of neighbors in PR479 is 15.2 and there are 70 proteins (14.6%) with 3 or less neighbors.

4.1.2 Class and Architecture distribution

Before trying to classify new proteins, we study the distribution of classes within the already classified ones in PR479. In a statistical sense, we want to learn the prior probability that a protein belongs to a certain class, $\mathrm{P}(E \in \omega_i)$.

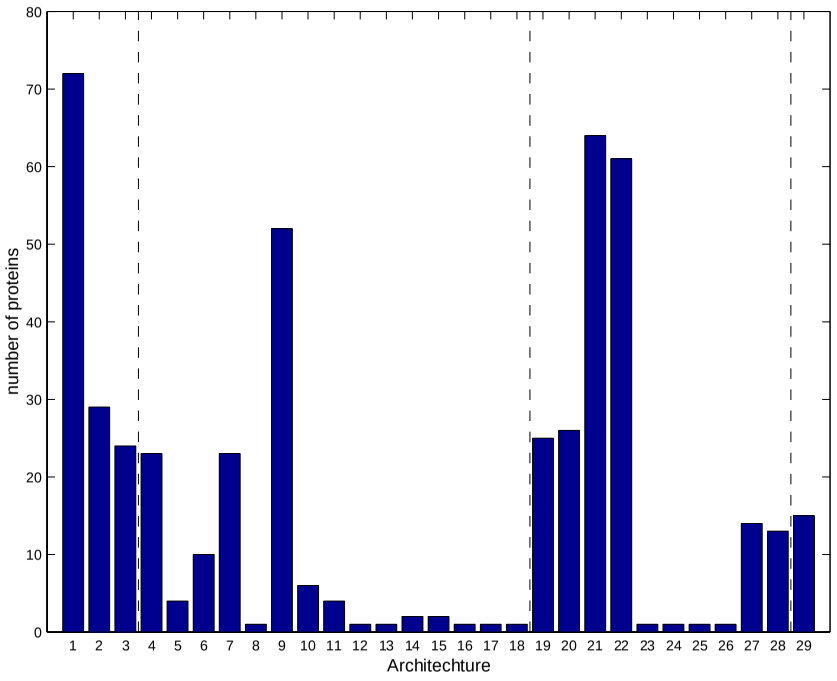


Figure 4.1: Histogram of the 29 CATH architectures that appear in PR479. Dashed lines separate for four class families.

In Figure 4.1 we plotted a histogram of the PR479 proteins according to their CATH

architecture. Dashed lines separate the four class families. It is evident from the figure that the architecture distribution is far from uniform and that there are only 14 architectures with ten or more proteins which, together, amount to 451 proteins (94%). Of the remaining architectures, there are 12 with one or two proteins only. This non-uniform distribution clearly makes it difficult to predict the less-frequent architectures. Therefore, when we evaluate the performance of a classification algorithm, we compare it to an upper bound (see 4.4.1) which takes into account the non-uniformity of the architectures.

4.1.3 Z ordered by CATH

As explained earlier, for each of the PR479 we obtain CATH classifications for all levels. Using these we can change the order of the PR479 set, which is originally set by FSSP according to the PDB entry. We can first order the proteins by their class; within the class, by the architecture and, within it, by the topology, and so on. As an example, in the new order all mainly-α proteins are numbered from 1 to 125. Using this order, one can reorder the columns and rows of the **Z** matrix. The reordered Z-matrix incorporates all the information we have to perform the classification; the matrix itself holds the similarities between the proteins and the order is set according to the CATH classification. By viewing the reordered Z matrix, one can get a clear idea of the extent to which the FSSP Z-scores reflect the CATH classification and, hence, how difficult is the task at hand.

Figure 4.2 depicts the ordered **Z**-matrix with a grid separating the proteins according to their CATH class. Figure 4.3 is the same matrix with additional grid placed at boundaries between the architectures. These matrices are the same as the upper–left sub-matrix of figure 3.1, that corresponds to PR479, only reordered according to CATH classification.

One can see from in figure 4.3 that there are no edges with $Z > 2.0$ in region A, that connect the mainly-α class and the mainly-β class. On the other hand, both of these classes are connected with proteins from the α-β class (region B), which is perfectly reasonable because a mainly-α protein can be similar to the α part of an α-β protein. Moreover, one can see that there are architecture families which are highly connected

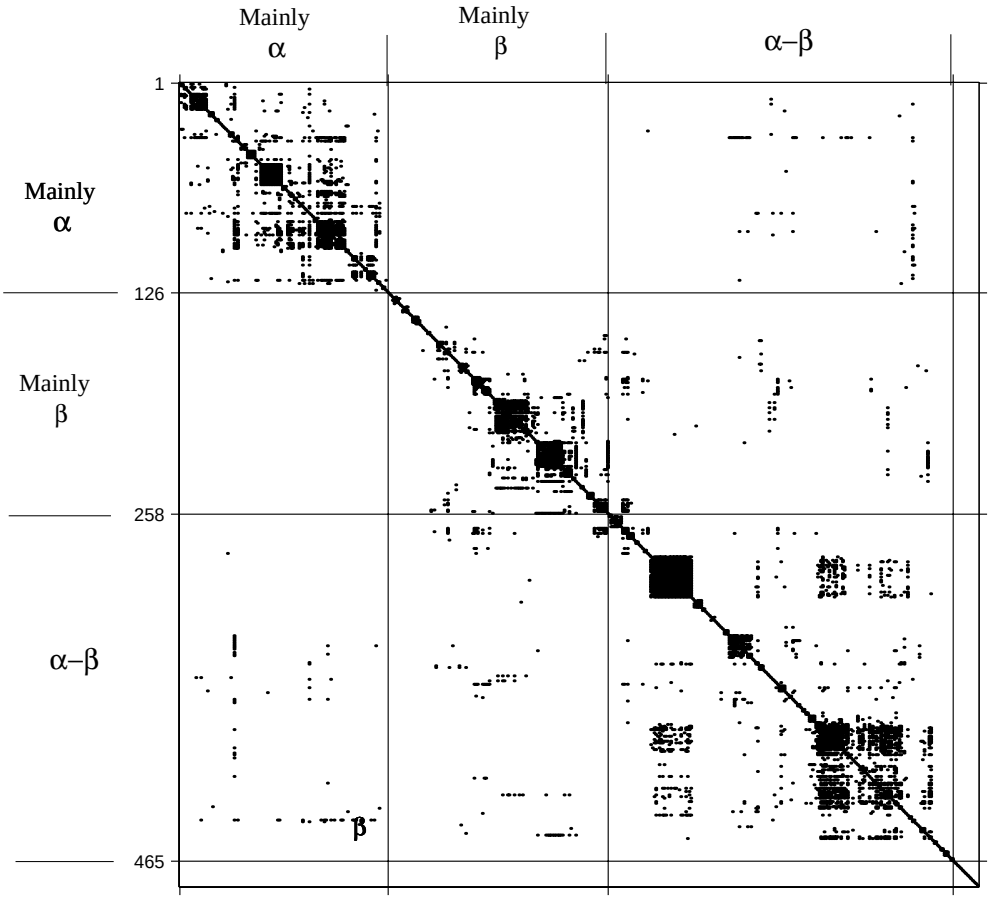


Figure 4.2: The PR479 Z-matrix ordered according to CATH classification. Each black dot represents $Z > 2.0$. The full–line grid separates the class level families.

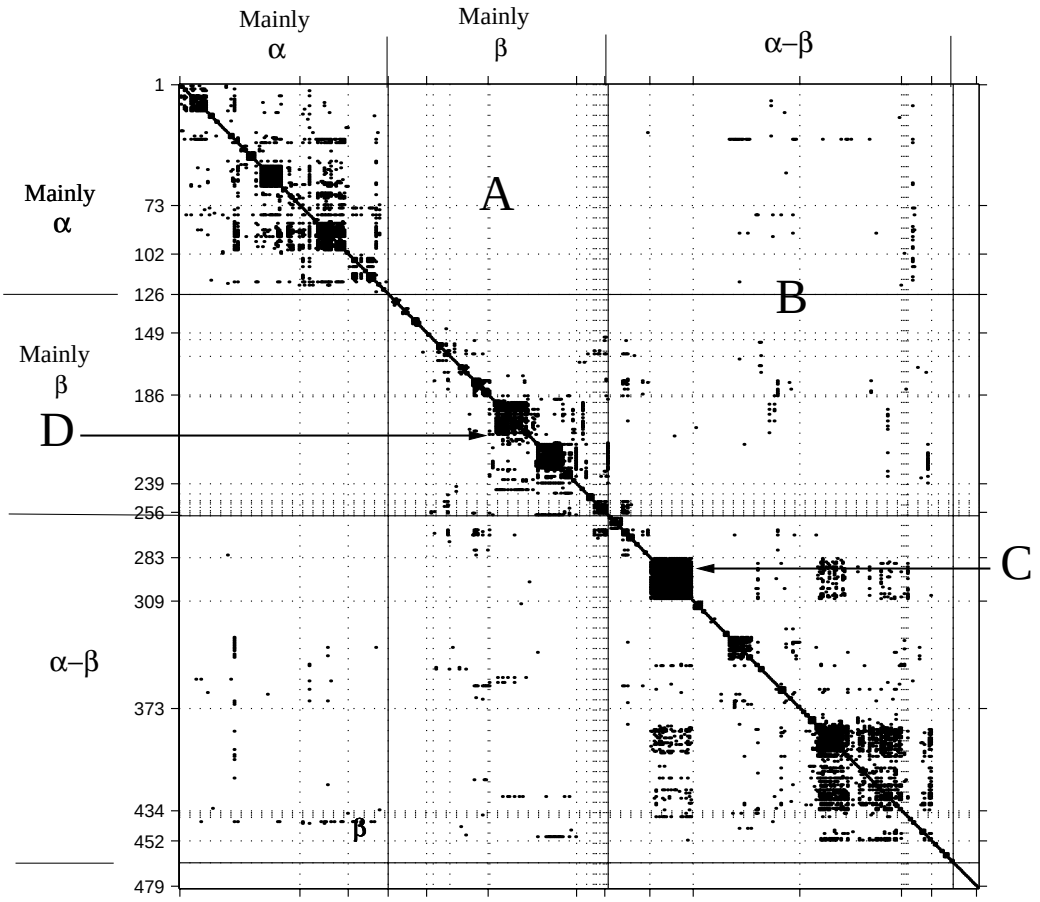


Figure 4.3: Same figure as previous one with additional dotted–line grid between the architectures. Marked regions: A - no $Z > 2.0$ connections between mainly-α and mainly-β; B - some connections between (mainly-α, mainly-β) and α-β; C - α-β barrels; D - mainly-β sandwiches

within themselves , e.g. α-β barrels (283-308: region C), and others which are very sparse. The connections within the mainly-β sandwich family (186-238: region D) have two relatively distinct subgroups which suggest an inner structure corresponding to the lower levels in the CATH hierarchy. Checking the topology level (the third CATH level), one indeed finds two large topology sub-families, the Immunoglobolin-like proteins (189-209: upper left part of region D) and the Jelly-Rolls (214-235: lower right part of region D), which correspond to these sub-groups. Noticing topology level structure in the Z-matrix is expected because the Z-score measures structural similarity of two aligned proteins, while keeping their connectivity order. Such similarity corresponds to the topology level in CATH. Overall, there are still many connections between proteins of different topology but of the same architecture and class that can be used for architecture classification.

4.2 Solution approach

As discussed in the previous Chapter, we want to find an algorithm that correctly classifies yet unclassified proteins. In particular, we chose to use nearest–neighbor type algorithms for the automatic architecture prediction, and as will be explained later it was the obvious choice in our case.

We tackle the classification problem using a series of techniques, trying to improve the classification ability at each step. We start from commonly used methods, like simple nearest–neighbor classification [9][11] using the similarity measures taken from the FSSP. We continue to improve the outcome by deriving new similarity measures based on the original ones, and use them with standard classification techniques to perform the classification. We identify two types of methods to derive new similarity measures; *direct* methods use only local considerations to alter the similarity measure, whereas *indirect* methods use global aspects of the original similarities to calculate new ones. Finally, we introduce a classification method that uses indirect considerations as part of the classification procedure.

The following sections describe and evaluate the algorithms and similarity measures used. Section 4.3 describes the nearest–neighbor algorithms we chose to use for the classification task. Section 4.4 discusses and evaluates the direct similarity measures, which include the original similarities and our proposed ones. The indirect similarities are dealt with in Section 4.5.

4.3 Nearest–neighbor algorithms

4.3.1 Background

Nearest–neighbor (NN) algorithms are widely known and studied [9][11]. A NNalgorithm classifies a protein according to the class of its nearest neighbors, where nearest is defined using some similarity measure or distance. The simplest NN algorithm is the single nearest–neighbor, $1NN$, in which the predicted class of a data point is the class of its closest or most similar classified neighbor, *i.e.* the one with highest similarity measure value with it.

Other NN methods use k neighbors in order to predict the class. One of them is the kNN method [11] which predicts the unknown class by a majority vote between k neighbors. *Weighted* kNN methods assign a weight to each of the neighbors' vote according to their distance and class. It is clear from the NN algorithm's definition that only similarities (or distances) between the data points are needed in order to predict the class of an unclassified point.

The $1NN$ algorithm is widely used because of its simplicity and relative success. It can be shown that the $1NN$ error rate, in the limit of large sample size, is bound by only twice the Bayesian error (the minimal error), which is quite remarkable for such a simple algorithm [9]. The disadvantage of NN–type algorithms is that the whole database of classified points has to be kept in memory and for each classification of a new point one has to find its closest points within the database. For very large databases this search could cause practical problems.

In our case *NN* algorithms are the obvious choice, since we already have pairwise similarities between the proteins. Other classification methods require that the items for classification be represented as vectors in some metric feature space. There are methods that first embed the points in some metric space while trying to maintain their given distances and then use other metric-space techniques for classification [12]. We decided to use *NN* algorithms using similarity measures. The size of our database is not large and therefore it causes no problems.

We tested two types of *NN* classification methods; The $1NN$ method and a weighted kNN method which we call *maxfield*.

4.3.2 Formulation

In the statistical sense, *NN* classification methods assume that the probability for the assignment of the unclassified points to classes can be separated into a product of single point probabilities, which means that these probabilities are assumed to be independent. Moreover, *NN* methods assume that a certain point belongs to a class with a probability that depends only on the classes of its close classified neighbors. Both assumptions are generally not fulfilled in real life situations, and therefore *NN*classification may not be optimal. These issues are further discussed in Sec. 4.6.

In our case, these assumptions mean that a protein's class depends only on those classified proteins which have non-zero similarity with it. Formally, Equation 3.3 in the previous Chapter can be written under these assumptions as

$$\{\hat{C}_i\}_{i=m+1}^{n} = \arg\max_{C_i} \mathrm{P}\left(\{C_i\}_{i=m+1}^{n} | \mathbf{Z}, \mathbf{S}, C\right) = \tag{4.1}$$
$$\arg\max_{C_i} \prod_{i=m+1}^{n} \mathrm{P}(C_i | \{Z_{ij}\}, \{S_{ij}\}, \{C_j\}) \; \forall j = 1, \ldots, m \text{ such that } S_{ij}, Z_{ij} > 0 \; .$$

The maximum of this product of independent probabilities is obtained by maximizing each of the terms. Therefore the choice of class for each unclassified protein can be performed separately,

$$\hat{C}_i = \arg\max_{C_i} \mathrm{P}(C_i | \{Z_{ij}\}, \{S_{ij}\}, \{C_j\} \; \forall j = 1, \ldots, m \text{ such that } S_{ij}, Z_{ij} > 0) \tag{4.2}$$

where $i = m+1, \ldots, n$.

In the $1NN$ algorithm the meaning of selecting the class of the nearest neighbor corresponds to formally assuming that this class is maximizing the above probability.

We tested a weighted kNN algorithm, which we call *maxfield*[1]. The maxfield method takes into account all the classified neighbors by assigning the similarity as the weight of each such neighbor. Hence the nearest neighbor (the one with the highest similarity) has the largest contribution to the class decision, but in cases where the nearest neighbor's class does not agree with several next nearest neighbors, the class prediction can change.

The maxfield algorithm assigns a protein to the class which has the maximal overall similarity with it. In order to calculate the overall similarity with a certain class, one adds the similarity of all the neighbors from that given class. Formally,

$$\hat{C}_i = \arg\max_{C_i} \sum_{j=1}^{m} S_{ij}\delta(C_i, C_j) \quad \forall\, i = m+1, \ldots, n\ , \tag{4.3}$$

where S_{ij} is any similarity measure and the δ is Kronecker's δ-function. Note that the sum is over all classified proteins. Proteins that have zero similarity with the classified protein do not affect the classification.

The maxfield algorithm can be introduced using a more statistical approach, giving a statistical meaning to the similarity score. Since this algorithm is related to the SPC clustering algorithm, the statistical representation is explained in Sec. 4.5.4.

NN algorithms have two types of rejections. The first type occurs when there are no close neighbors with known classification to base the prediction on, *i.e.* $S_{ij} = 0$ for all $j = 1, \ldots, n$ where S is any similarity measure and i is the index of the unclassified protein. The second type is when there is a tie between several classes and therefore no prediction can be made. A $1NN$ algorithm returns a type-two rejection when the point to be classified has several neighbors with identical similarity that belong to different classes. In this case, the algorithm can not decide which class to assign to the unclassified point.

[1]The name *maxfield* is used by an analogy to physics of inhomogenous magnets. This analogy is discussed in Sec. 4.6.

When we apply a *NN* algorithm for the classification of an unclassified protein we search for neighbors only among the set of classified proteins. To view this graphically, return to the Z-matrix in figure 3.1 (page 16). *NN* classification methods utilize only the rectangles in the matrix that represent similarities between the classified and unclassified proteins. The upper–left and lower–right square sub-matrices, which correspond to connections within the classified proteins (PR479) and *within* the unclassified ones (PR166) are disregarded.

4.4 Direct Similarities

We tested the classification methods using four direct similarity measures, two of which are the original S and Z scores obtained from the FSSP and the other two are derivatives of them, CS and CZ (defined in 4.4.2), which are normalized versions of the S and Z, that take into account the self-similarity of the pair of compared proteins. The results of these methods are described below.

Before testing the direct similarities, we analyze the distribution of the data in order to reach an upper bound for the success rate of the algorithms.

4.4.1 Upper bound analysis

In order to obtain an upper bound on nearest-neighbor success, we examined the neighborhoods of the proteins in PR479. We characterize the neighborhood of a protein according to the kinds of its neighbors. There are four possible cases:

A. "Islands" - The protein has no neighbors.

B. "Colonies" - It has no neighbors of its own kind.

C. "Border" - It has neighbors of its own kind as well as of other kinds.

D. "Interior" - The protein has only neighbors of its own kind.

Using these definitions we can arrange the proteins in PR479 in 7 groups according to their neighborhood type at the class and architecture levels. The 7 possible groups are 1(AA), 2(BB), 3(CB), 4(CC), 5(DB), 6(DC), 7(DD), where the first letter defines the neighborhood at the class level and the second letter specifies the architecture surrounding. Group 5, for example, includes those proteins whose neighborhood consists of proteins of the same CATH class as their own (Type D) but different architecture from its own (Type B).

Using this partitioning, one can calculate an upper bound for a leave–one–out success rate estimate of a nearest–neighbor algorithm. When calculating a leave–one–out estimate, one tries to predict the class of a protein using all others. Therefore, proteins which do not have neighbors of their own type could never be classified correctly by a nearest–neighbor algorithm. This includes proteins with neighborhoods from types A and B; thus only proteins that belong to groups 4,6 and 7 can attain their correct class and architecture assignment.

Group	number of points	Class level $1NN$ (Bound)	Architecture level $1NN$ (Bound)
1 (AA)	24	0 (0)	0 (0)
2 (BB)	17	0 (0)	0 (0)
3 (CB)	32	23 (32)	0 (0)
4 (CC)	250	236 (250)	216 (250)
5 (DB)	9	9 (9)	0 (0)
6 (DC)	102	102 (102)	95 (102)
7 (DD)	45	45 (45)	45 (45)
Total	479	415 (438)	356 (397)

Table 4.1: Number of correct predictions using the $1NN$ algorithm with Z-score similarity measure. The bound for the classification appears in parenthesis.

Using the above 7 groups, one can calculate the bound for predicting the class and the architecture. Table 4.1 lists the number of proteins in each group and the number of correct predictions made by $1NN$ algorithm using the Z-score as similarity measure. In the table rejections are counted as not correct. The bounds for the classification appear

in parenthesis.

The class prediction success rate is bound by **91.4%** (438/479) which is the percentage of proteins in groups 3,4,5,6 and 7 together. These groups include the proteins with at least one neighbor from their own class. The architecture prediction rate is bound by **82.9%** (397/479) which is the percentage of proteins in groups 4,6, and 7.

The leave–one–out success rate estimate of $1NN$ using Z-score as a similarity measure, can be read off table 4.1. For class prediction it is **86.7%** (415/479) and for architecture prediction it is **74.3%** (356/479). One can see that the margin between the estimates for the $1NN$ architecture prediction rate and the upper bound is only **8.6%**.

It is important to keep in mind that there are possible mistakes in the CATH classification. For example, proteins in group 2 do not have any neighbor of their own kind, not even at the class level. This means that the FSSP similarity measure finds these proteins similar only to proteins of different classes and obviously different architectures. This makes their CATH classification questionable.

PDB entry	CATH arch.	No. of FSSP neighbors	Most frequent neighbor arch.	No. of neighbors from that arch.
1rboC	21	7	9	7
2cas	18	22	9	20
1hgeA	27	21	9	18
1regX	27	9	21	7
1celA	10	18	9	13

Table 4.2: List of our suspected misclassifications in the CATH database.

One of the 17 proteins in group 2 is PDB entry **1rboC**, which is assigned to architecture 21 (α-β : Two-layer sandwich), but all of its 7 neighbors are assigned to architecture 9 (mainly-β : Sandwich). One could argue that architecture 21 is a very small family and that the protein happens to be more similar to proteins of a different family. But this is not the case since both of these architecture families have more than 50 proteins in PR479. This protein may be a misclassified in the CATH database. We suspect as

misclassifications proteins from group 2 and 3 which have more than 5 neighbors out of which at least 70% are from the same kind. A list of these proteins is given in 4.2.

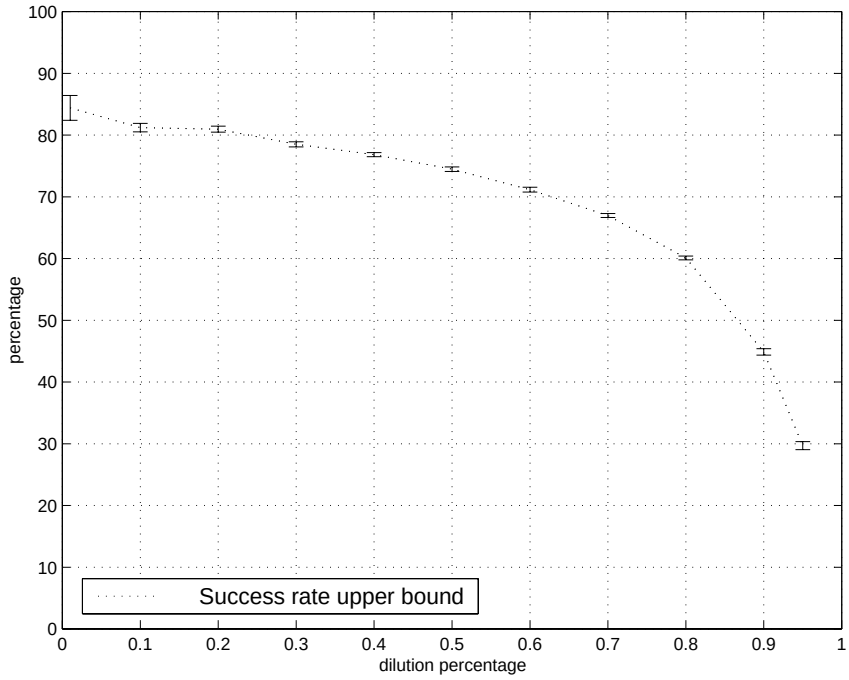


Figure 4.4: The upper bound for classification using any similarity measure that corresponds to a sub-graph of the original one.

We repeated the above upper bound calculation for all the realizations at the 11 dilution values used to evaluate the classification performance (see Sec. 3.3). For a given realization at a certain dilution value, one can calculate what is the percentage of points in the test set that have at least one neighbor from the training set of their own kind. This percentage is an upper bound for the performance of a nearest–neighbor classification method on this specific realization. In figure 4.4 we plot the average of this percentage over our realizations. Notice that the average is very stable, *i.e.* the error-bars are small and therefore this average is a good estimation for the true bound at these dilution values. We shall plot this bound using dotted lines on all the evaluation figures from now on. This bound takes into account the non-uniformity of the architectures in the data (see 4.1.2) since non-frequent architectures are likely to be diluted and leave their members without any neighbors of their kind.

The above bounds are applicable to any similarity matrix that corresponds to a subgraph of the original one, *i.e.* it has non-zero elements positioned where **S** (or **Z**) is non-zero. Any such similarity measure can only reorder or break edges in the graph and, therefore, all proteins with no neighbors of their own kind will stay in that situation.

4.4.2 Z and S scores

The straightforward direct similarity measures are the original FSSP S and Z scores obtained from Liisa Holm (one of the creators of the FSSP database). We tested the $1NN$ and maxfield classification algorithms using each of these similarities. We will refer to a test by specifying the classification algorithm and in parenthesis the similarity measure used, e.g. $1NN$(Z) means classifying using $1NN$ and Z as similarity measure.

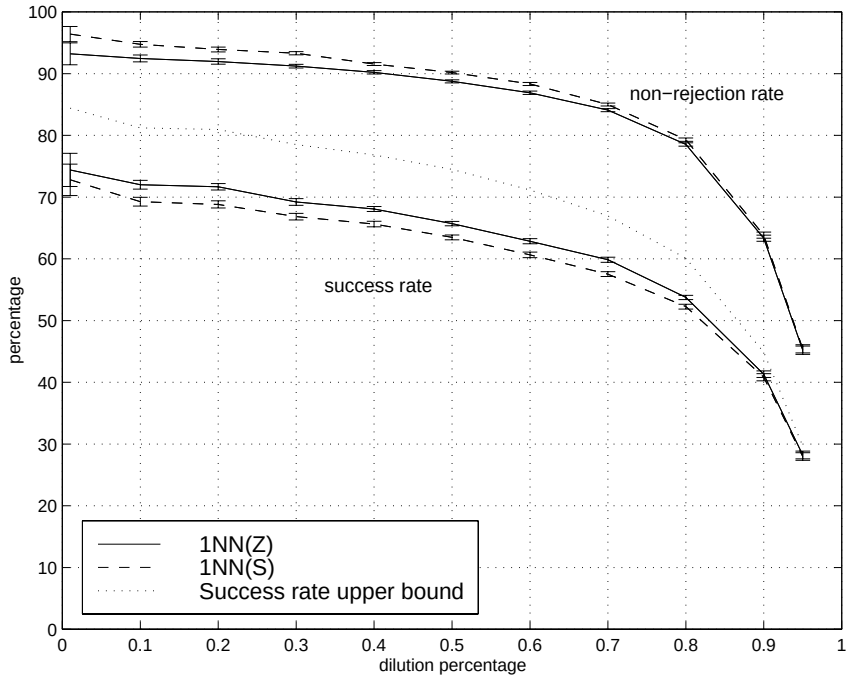


Figure 4.5: The success (lower lines) and non-rejection (upper lines) rates for the $1NN$ algorithm using S and Z similarity measures.

Figure 4.5, shows the cross-validation results for a $1NN$ classification algorithm using these scores at different dilution percentages. One can see that the Z score performs *better* than the S score at all dilution rates. At 30% dilution, for example, the $1NN$

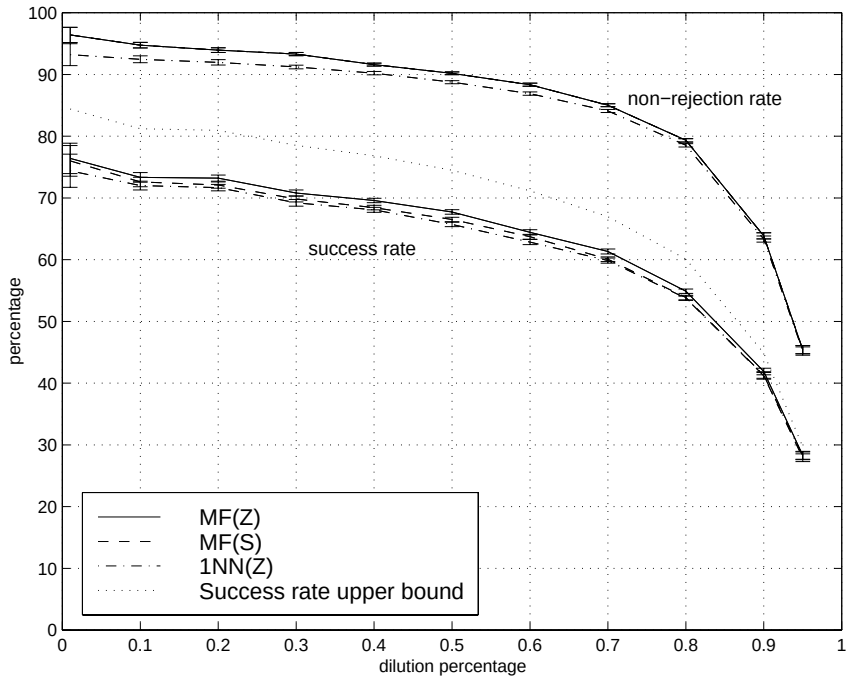


Figure 4.6: The success (lower lines) and non-rejection (upper lines) rates for the maxfield algorithm using S and Z similarity measures compared to the $1NN$ algorithm using Z.

using the Z score has a success rate of $69.2 \pm 0.5\%$ whereas the performance using the S score is $66.8 \pm 0.5\%$, which is a difference of above 4 standard deviations. In addition , the rejection rate of $1NN$(S) is lower than the one of $1NN$(Z), which makes the purity (success out of the non-rejected proteins) difference even bigger. Figure 4.6 depicts the maxfield cross-validation results using S and Z scores compared to the $1NN$(Z). The plot shows that the MF(Z) performs slightly better than the others. When using the original scores, the best classification can be achieved using MF(Z).

4.4.3 CS and CZ scores

This section describes two direct similarity measures, CS and CZ, derived from the original S and Z scores, respectively. These measure are based on our understanding that the S and Z scores behave like a dot product between two unnormalized vectors (see Appendix A). For simplicity, we define the similarity measure CZ in terms of Z, but the same definition is applied for CS.

Each protein is treated as a vector in some inner–product–space, where the Z score is considered the outcome of a dot product between two proteins. The self-similarity of the proteins is, therefore, the squared norm of the vector. Following this picture, we define a new similarity measure, CZ, as the cosine of the angle between the vectors, *i.e.*

$$CZ_{ij} = \frac{Z_{ij}}{\sqrt{Z_{ii}Z_{jj}}} \ . \tag{4.4}$$

Surprisingly, the S and Z scores agree with this measure and all the $\{CZ_{i\neq j}\}$ are less than 1, as expected from a cosine measure. Note, that all the CZ_{ij} are positive since all the S and Z scores are positive.

The advantage of using the cosine score is that it takes into account the self-similarity of the proteins. As explained in Appendix A, the S and Z scores are higher when the aligned segments of the compared proteins are longer. Consider two short mainly-β proteins from the same architecture family, and suppose 70% of their residues are aligned. In addition, consider a long α-β protein that in its β part there is a segment that is aligned to 80% of one of the mainly-β proteins. In this case, it is possible that the Z (and S) score for the pair of short proteins will be smaller than the score of the short-long pair, and could cause a misclassification of the short protein.

We tested the CS and CZ similarities using both $1NN$ and maxfield classification algorithms. The results compared to MF(Z) appear in figures 4.7 and 4.8. From figure 4.8 we learn that the success rate of MF(Z), MF(CS) and MF(CZ) are practically the same. In figure 4.7 we see that the $1NN$ classification algorithm using CZ is slightly better than MF(Z) at low dilutions. Note that the non-rejection rates of all methods are identical, as can be seen in both figures. Therefore, the best combination of algorithm and direct similarity measure is $1NN$(CZ).

Note that in order to change the $1NN$ algorithm's performance by using a new similarity measure derived from the original ones, one has to apply a non-monotonous transformation on the scores. The transformation has to reorder the similarities between neighbors of a protein to change the $1NN$ classification. The CZ transformation is non-monotonous

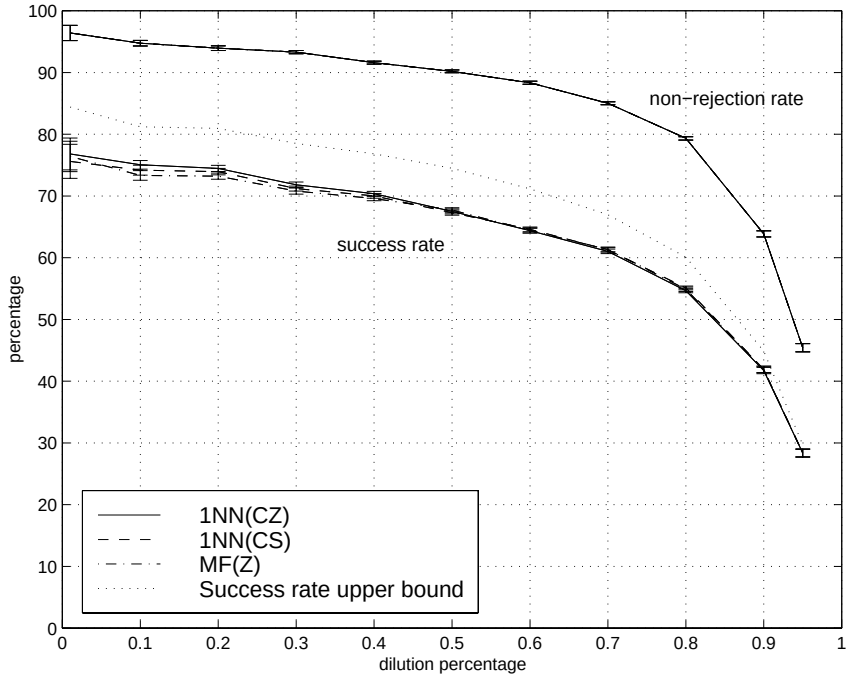


Figure 4.7: The success (lower lines) and non-rejection (upper lines) rates for the $1NN$ algorithm using CZ, CS similarity measures compared to the maxfield algorithm using Z. The non-rejection rates of all methods coincide.

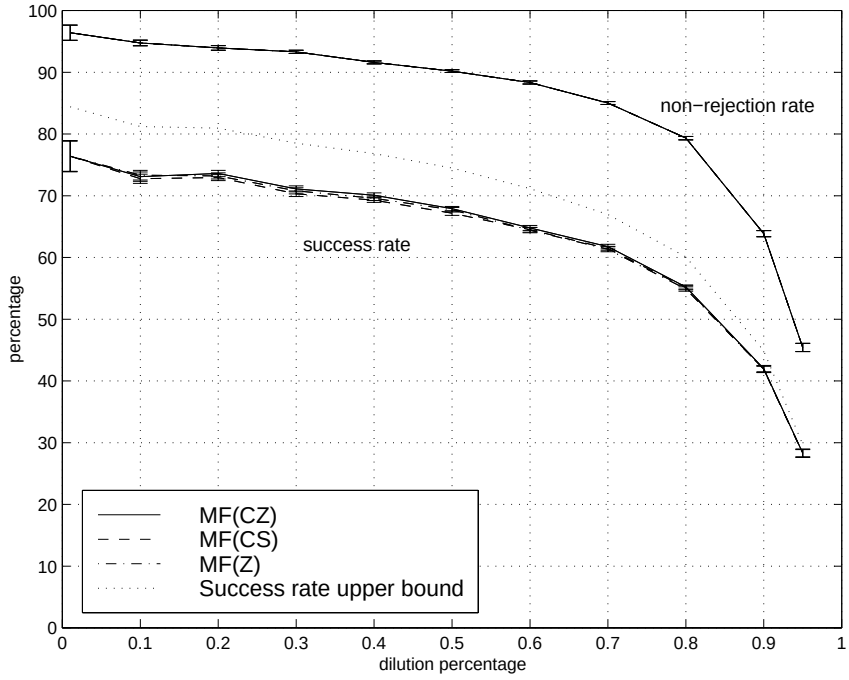


Figure 4.8: The success (lower lines) and non-rejection (upper lines) rates for the maxfield algorithm using CZ, CS and Z direct similarity measures. The non-rejection rates of all methods coincide.

with respect to Z_{ij} and indeed improves the $1NN$ classification. In the maxfield case, there is no need for a non-monotonous transformation, even a slight shift in the scores can change the algorithm's outcome.

4.5 Indirect similarities

In this section we describe our proposed *indirect* similarity measures. These measures use non-local properties of the original similarities to calculate new ones. The classification, using these measures, is a two staged process. First, we calculate the new similarities based on the direct ones and then we apply a $1NN$ algorithm for the classification.

Indirect similarities do not assume that the class probabilities of the unclassified proteins are independent. They try to incorporate the correlations between the proteins into the new similarities, *i.e.* move correlated points closer together. The indirect similarities take into account the direct similarities between all the proteins, classified as well as unclassified, when calculating the new similarities.

We propose and evaluate several indirect similarity measures which are all based on pre-clustering the data. Data clustering (or "unsupervised learning") is an important aspect of data analysis, where one learns about the data by investigating its distribution and identifying in it distinct groups or hierarchies of groups [19][9][6]. Clustering techniques use only measurements regarding the objects to be clustered and do not use any external category labels. This is the distinction between clustering and classification techniques. In other words, clustering algorithms make use of the similarity matrix, $\mathbf{S}$, between all the proteins, classified and unclassified, but they do not use the known labels of the classified proteins, $\{C_i\}_{i=1}^{m}$.

We test a number of clustering methods for the preprocessing stage: The first method, the K-mutual-neighborhood-value (KMNV) [4] (see 4.5.1), removes connections from the similarity matrix using correlations between close neighborhoods of the compared proteins; therefore, this method is still rather local. The next two methods are a modified version of the super para-magnetic clustering algorithm (SPC) [4] (see 4.5.4) and the average–linkage algorithm (AVL) [19] (see 4.5.5). Both these algorithms are hierarchical clustering methods (see 4.5.2) that produce a tree of nested partitions of the data called a *dendrogram*. We use the dendrogram to create new similarities.

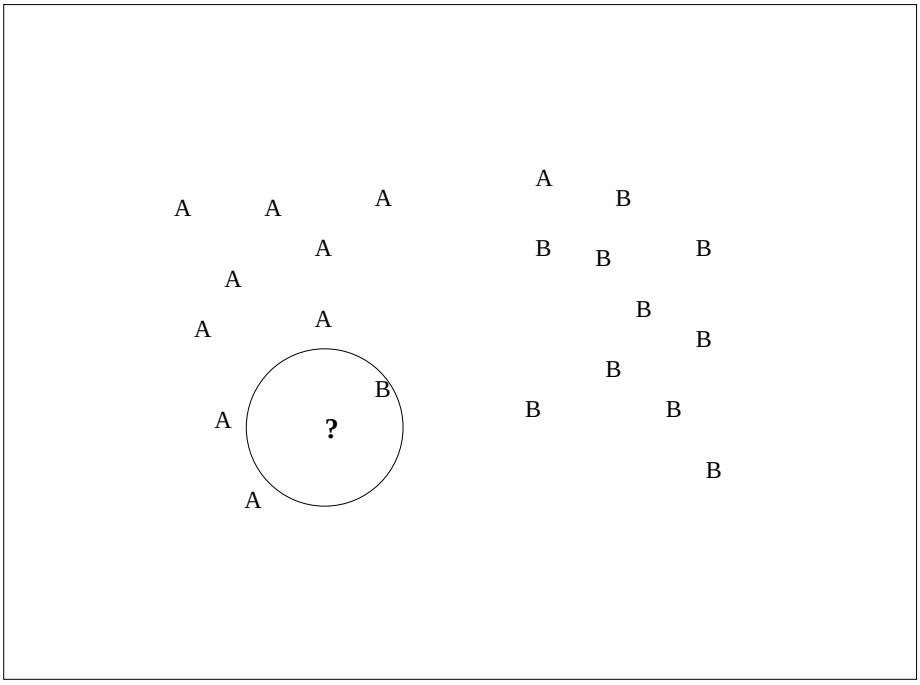


Figure 4.9: "stray" B-type point is the nearest neighbor of the unclassified point.

Using clustering in order to increase classification performance is based on the following notion: consider a classification problem with two different classes, A and B, that have overlapping distributions, meaning that there are A points within the B regions and *vise versa* (see figure 4.9). In such a case there are many A-type points along the borders , such that within their close neighborhood there are "stray" B-type points and are therefore likely to be classified wrongly by nearest–neighbor techniques. If, however, we assume that the classes form separate clusters, we can first use a clustering algorithm to bind together points of the same cluster (and class). Then, we can perform the classification based on the points of known classes from the same cluster and may improve the performance.

The effect of using clustering can be even more noticeable as the data get more diluted. This is due to the fact that at high dilutions more points are at the border between classified points of different classes, meaning that the closest classified point (to the one of unknown class) may often belong to a different class. Moreover, at high dilutions, the fraction of unclassified points is higher and the information contained in their distribution, which is disregarded by direct similarities, can help the clustering methods to separate

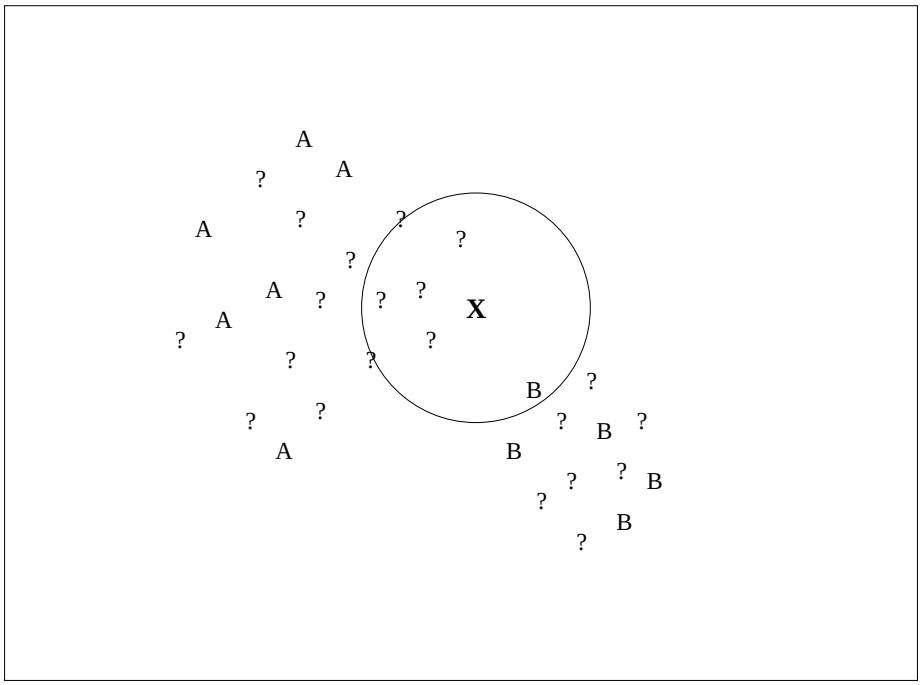


Figure 4.10: Highly diluted distribution; the nearest labeled neighbor of the unclassified point x is a B-type point. On the other hand, a clustering algorithm will assign point x to a cluster that contains classified points of type A.

the classes. Such a case is presented in figure 4.10.

Notice that all these methods are "additive", in the sense that we can apply them consecutively. For example, we can use the KMNV method to cluster the data on the basis of the CZ direct similarities; use the resulting clustering to create new similarities, and then apply the SPC clustering method using these new similarities as input in order to generate the final similarities that then are used for classification.

We now describe each of the clustering methods and evaluate their performance using a $1NN$ classification. We continue with the same notation as for the direct similarities, but now we specify also the clustering techniques used as part of the similarity measure; For example, $1NN$(SPC-K10-CZ) refers to the $1NN$ algorithm applied to the combined similarities SPC-K10-CZ. These are produced by first applying the KMNV method (with $K = 10$) to the CZ direct similarities and then using the SPC algorithm to generate a dendrogram from which new similarities are produced (that are used as inputs of the

$1NN$algorithm).

4.5.1 K mutual neighborhood value method (KMNV)

The task of the K-mutual-neighbors-value method (KMNV) is to remove (prune) edges from a similarity graph or, in other words, setting some similarity values to zero. One can summarize the algorithm as follows:

Step 1. For each point i, order all of its neighbors j by their similarities to i and set k_{ij} to the ordinal number of neighbor j. Note that in general $k_{ij} \neq k_{ji}$.

Step 2. Keep only connections for which $k_{ij} \leq K \; and \; k_{ji} \leq K$, meaning that point i and point j are at most the K-th neighbors of each other. All other connections are set to zero.

The number of connections that are removed by KMNV depend on the integer parameter K. For large K values, ($K > n$ number of points), no connections are removed. Small K values leave only few connections. The maximum number of connections that are kept is, of course, $K * n$.

This algorithm was used by Blatt *et al.* [4] in order to save computation time of the SPC algorithm by reducing the number of connections in the graph. In the problems they dealt with the SPC's performance was not sensitive to the value of K. When testing the SPC clustering algorithm on the protein data, we found out that the results do depend on K and that this preprocessing step plays an important part in the clustering.

We analyzed the effect of using this algorithm and found that it separates (*i.e.* removes connections between) regions that have different densities. In order to explain this behavior, picture the points in some metric space and use distances instead of similarities. Using this picture it is convenient to state the algorithm as follows: Consider a sphere around each data point that includes its K nearest neighbors. The connection between two points is kept only if both are within the spheres of each other.

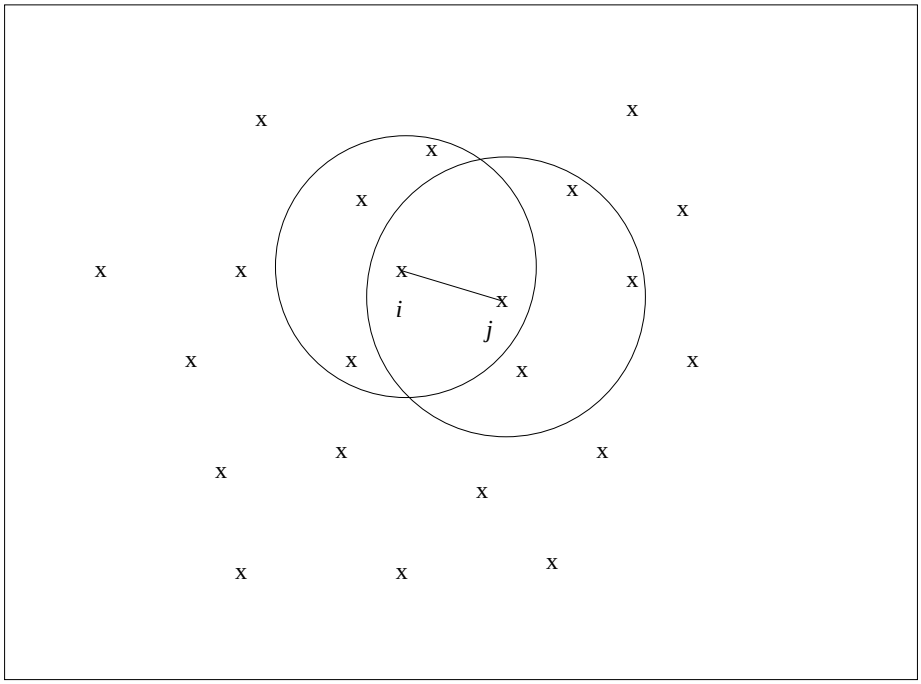


Figure 4.11: Uniform distribution; The connection between i and j is kept.

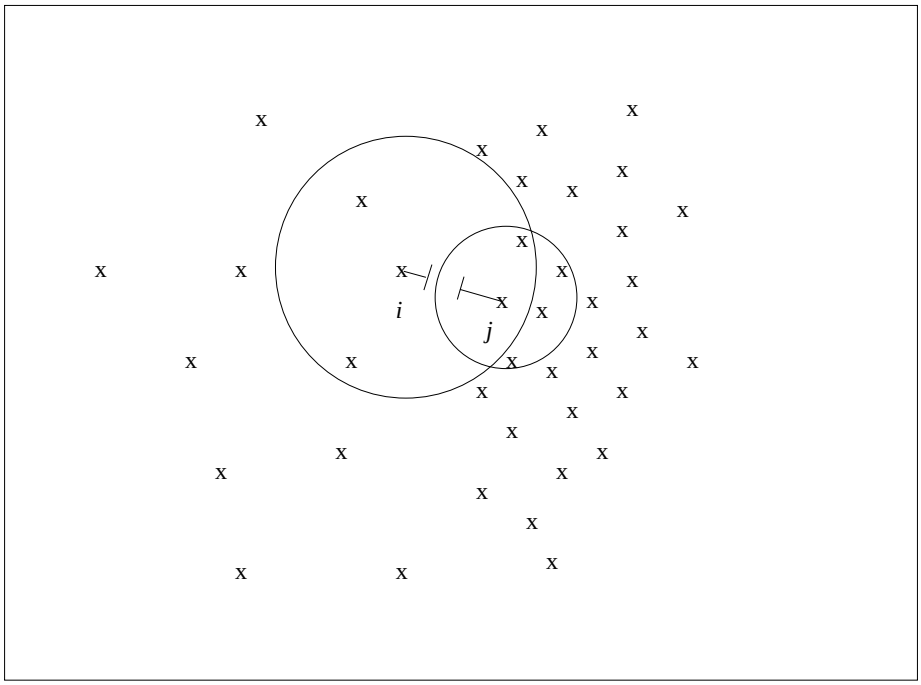


Figure 4.12: Gradient in the density; The connection between i and j is removed.

In a *uniform* density, see figure 4.11, it is likely that if point i is inside the sphere of point j then also point j is inside the sphere of point i. Therefore, in the uniform density case, as a first approximation, points that are closer than the average distance to the K-th neighbor will be connected. On the other hand, if there is a *gradient* in the density (see figure 4.12) on the scale of the average K-th neighbor distance, then spheres around points in the dilute region will include points from the dense region, whereas points in the dense region will have small spheres around them which will not include the points in the dilute region; hence two such points will not be connected. Therefore, the overall effect of the KMNV algorithm is to remove the connections between regions of different densities.

Note that because the KMNV algorithm can only remove connections, the upper bounds on the classification performance obtained in Sec. 4.4.1 are still valid when classifying using the new similarities.

The KMNV algorithm may help the classification by removing connections between the α and the α-β proteins and β and α-β proteins. The α-β proteins have large similarities between them which corresponds to a dense region in space, whereas, the similarities between the α proteins and themselves (and the β proteins and themselves) are relatively lower which corresponds to low density regions. Since the KMNV algorithm separates regions of different density it might help to classify the proteins. On the other hand, removing connections also reduces information in the data that is used by classification methods. Hence, we expect that there will be an optimal K value for each classification method.

The KMNV's performance depends on its parameter K. In order to select a range for the values for K, we examined the distribution of number of neighbors in the PR479 direct similarities. We chose to test the following K values: 5, 10, 15, 20, 25, 30. It is obvious that large K values will not have a large effect on the similarity and the classification performance since most of the direct similarities are kept.

We created a set of 12 similarity matrices based on 2 direct similarities Z and CZ, which gave the best performance when used alone, using the 6 different K values. Overall, the KMNV algorithm does not improve the classification beyond the best algorithm so far.

The performance of the classification using K=25 and 30 was practically the same and very similar to the performance of the direct similarities with out the KMNV. The KMNV algorithm slightly improved the classification performance of the Z direct similarity, but did not improve the CZ direct similarity. This can be the case if the KMNV has a similar effect on the similarity matrix as the transformation from Z to CZ. The CZ transformation reduces the relative similarity between a pair of proteins that differ in their self-similarity. For example, suppose an α protein, that has small self-similarity, has higher Z value with an α-β neighbor compared to other α neighbors. The CZ value with the α-β neighbor can be lower than CZ value with the α neighbors since the self-similarity of the α-β protein is large. Thus, the CZ value behaves like the KMNV and separates the α proteins from the α-β proteins.

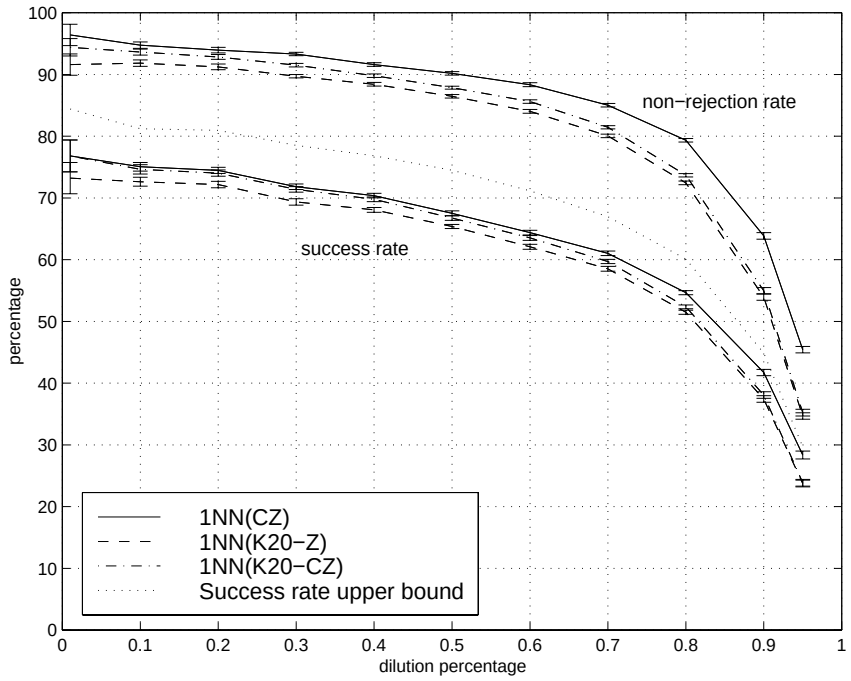


Figure 4.13: The best KMNV similarities ($K = 20$) and the best direct similarity $1NN$(CZ) success (lower) and non-rejection (upper) rates.

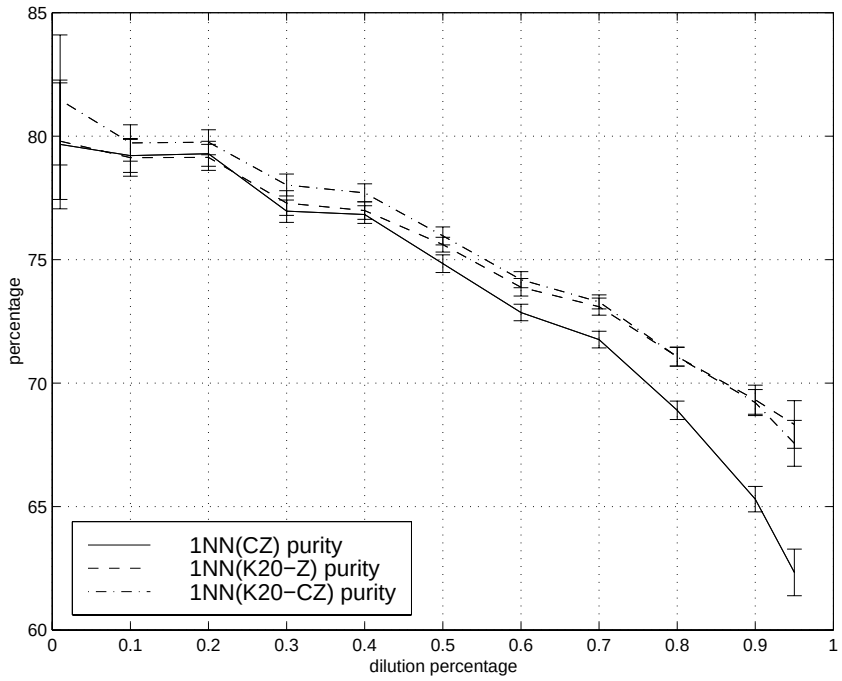


Figure 4.14: The purity obtained using the best KMNV similarities ($K = 20$) and the best direct similarity $1NN$(CZ).

For comparison we plot the best direct similarity classification which was $1NN$(CZ) together with the best of the KMNV ($K = 20$) similarities in figure 4.13. Notice that all the KMNV performance does not improve the overall success rate of the classification. However, if one considers the purity, *i.e.* the number of correct classification out of the non-rejected ones, the $1NN$(K20-CZ) has slightly higher purity for at all dilutions (see figure 4.14).

4.5.2 Similarities based on hierarchical clustering methods

This section describes how we use hierarchical clustering methods to produce new similarity measures.

Hierarchical clustering methods produce a sequence of nested[2] partitions to describe the data [19]. Such a sequence can be described by a tree which is called a *dendrogram* (see figure 4.15).

[2]Partition A is nested in partition B, if each set in A is a proper subset of a set in B.

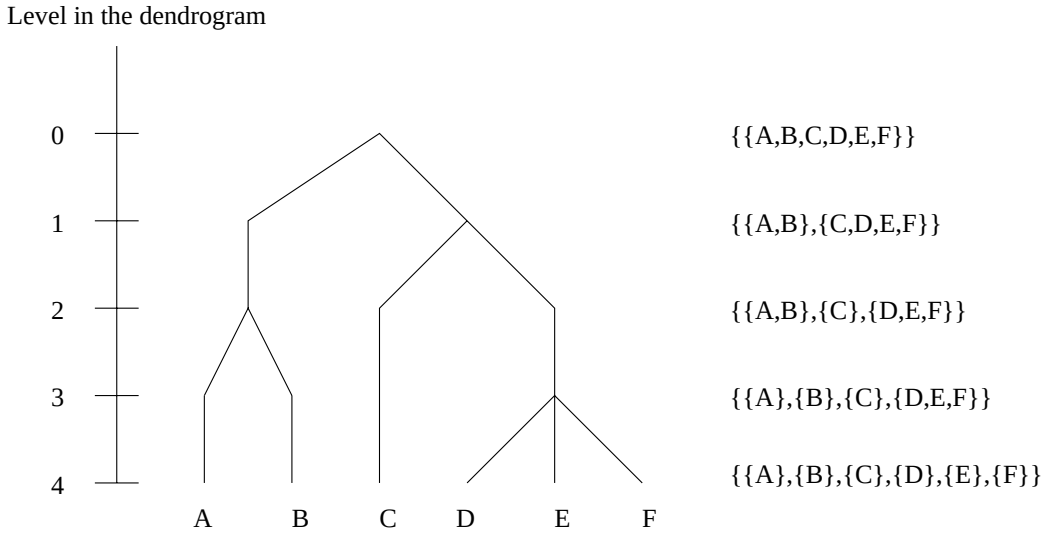


Figure 4.15: A dendrogram which describes the sequence of partitions listed on the right.

The dendrogram provides a picture of the clustering that can be easily interpreted. Cutting the dendrogram at any level defines a particular clustering, or partition, in the sequence. The level itself which is the index in the sequence of partitions, in general, has no direct correspondence to the original similarities that are used to create the clustering. The leaves of the dengrogram are the individual data points; at the lowest level each points constitutes its own cluster.

Using the dendrogram one can define a new similarity measure between the data points, called the *cophenetic similarity* [19]. The cophenetic similarity, S_{ij}^{C}, between two given points is defined as the lowest level in the dendrogram at which the two points are still in one cluster. The levels are numbered in increasing order from zero at the root towards the leaves. The cophenetic proximity is a similarity measure since two points which stay together at the same "branch" in the dendrogram, *i.e.* remain members of the same cluster, get a higher cophenetic proximity than points that split to different clusters close to the root.

Note that the cophenetic similarity is an integer valued variable that could be at most $n-1$ for a dendrogram that splits, at each level, one point out of the main cluster. Since there are at most n different values for $n*(n-1)/2$ relationships, there are many pairs

which have the same cophenetic similarity.

We suggest to use the cophenetic similarity as our new similarity measure. Finally, the classification is performed by applying a $1NN$ algorithm to the new similarities.

One can express graphically the classification procedure as follows: in order to classify a yet unclassified point, one starts going up towards the root from the leaf that corresponds to that point in the dengrogram. At each step, which corresponds to a lower cophenetic similarity, more points join the cluster of the unclassified point. The first time one (or more) classified points join the cluster, one performs a majority vote between their classes to obtain the predicted class of the unclassified point. In the case where a single classified point joins the cluster, the prediction is based only on a single point's class. This fits the $1NN$ algorithm. However, when several classified points join at the same level, meaning that there are several points with the same cophenetic similarity, one has to perform a kNN type decision. We chose to use a majority vote between the classes and demand that at least 70% of the votes are for a certain class, otherwise a rejection of the second type (see Sec. 4.3) is returned.

4.5.3 Direct classification using the dendrogram

We compare the results of the $1NN$ algorithm applied to the cophenetic similarity measure to a more direct classification method (denoted by D) that uses the resulting dendrogram with a top-down approach. The algorithm is as follows: first, we identify large and pure clusters by going down the dendrogram and stopping whenever a cluster is larger than 5 proteins and its purity exceeds 80%, meaning that more than 80% of its classified members are of the same architecture. Then, all unclassified members of these clusters are assigned to the class of the majority. The remaining unclassified proteins that belong to small or non-pure clusters are rejected. Figure 4.20 (on page 58) shows the dendrogram created using the SPC algorithm. The clusters that are plotted in the figure coincide with those used for classification in the direct algorithm.

4.5.4 Super Paramagnetic Clustering algorithm (SPC)

The first type of hierarchical clustering method we used is the Super Paramagnetic Clustering algorithm (SPC) which was introduced by Blatt *et al.* [4][5][6]. We modified the algorithm in order to use it as a hierarchical clustering method. The algorithm, which was first motivated by properties of inhomogeneous magnets, will be described here in terms closer to those used in the machine learning literature.

Each protein E_i is assigned an integer random variable, c_i, describing its "color" out of q possible colors, *i.e.* $c_i = 1, \ldots, q$. For each coloring configuration $\mathcal{C} = \{c_i\}$ a cost is assigned that penalizes assigning different colors to any pair of similar proteins. We choose the similarity measure of proteins i and j, S_{ij}, which can be any of the similarity measures we used, as the value of this penalty. The cost function is defined as the sum of penalties for all protein pairs $\langle i, j \rangle$,

$$E(\mathcal{C}) = \sum_{\langle i,j \rangle} S_{ij} \left[1 - \delta(c_i, c_j)\right] \ . \tag{4.5}$$

Note that if a pair of proteins $\langle i, j \rangle$ has $S_{ij} = 0$ it does not contribute to the cost function. Hence, the cost function depends only on colors of neighboring proteins in the corresponding weighted graph.

The general idea of the algorithm is to measure correlations between colors assigned to proteins at different values of average cost. Then, for each average cost value, one creates clusters by joining together all protein pairs whose correlation exceeds some threshold. The average cost at which the correlations are measured controls the size of the clusters and, hence, the resolution at which we view the data. By increasing the average cost, one gets a hierarchy of clusters, starting from the root where all the proteins are in one cluster and going towards the leaves where each protein forms its own cluster.

Setting the average cost to a specific value and using the maximum entropy principle [20] implies a Gibbs distribution over all possible configurations. The maximum entropy principle states that in order to infer an unknown distribution given some prior knowledge,

one should select the distribution with maximum entropy that satisfies the prior knowledge constraints. In our case, the constraint is the average cost value. The distribution that maximize the entropy, given this constraint, is the Gibbs distribution,

$$\mathrm{P}(\mathcal{C}) = \frac{\exp\left[-E\left(\mathcal{C}\right)/T\right]}{Z} \quad , \tag{4.6}$$

where Z is a normalization factor and $1/T$ is the Lagrange coefficient coupled to the average cost constraint. T has a one-to-one monotonic relation with the average cost, hence can be used to control it. In statistical physics terminology, Z is called the partition function and T is the temperature.

For each value of T one can form clusters using the two-point correlations of the colors. The two-point correlation function $G_{ij}(T)$ is defined as,

$$G_{ij}(T) = \sum_{\mathcal{C}} \mathrm{P}(\mathcal{C})\delta(c_i, c_j) \ , \tag{4.7}$$

where $\delta(c_i, c_j) = 1$ if i and j have the same color in the assignment $\mathcal{C}$, otherwise $\delta(c_i, c_j) = 0$. $G_{ij}(T)$ is the expectation of proteins i and j having the same color at a given average cost. The clusters are built by connecting all neighboring pairs that are highly correlated, *i.e.* $G_{ij} > \theta$; The threshold θ is usually taken to be 0.5. The clusters represent a typical color assignment of the proteins for a given average cost.

In order to form a hierarchy, one should make sure that if two proteins are in different clusters at $T = T_1$ then they stay in different clusters for any $T > T_1$. Two points are assigned to the same cluster if they are connected by a "path" of neighboring pairs, whose correlations exceeds the threshold. It can be shown that $G_{ij}(T)$ is a monotonically decreasing function of T [13]. Therefore, if a pair's correlation decreases below the threshold at T_1, it will remain below the threshold for all $T > T_1$. Consequently, if at $T = T_1$ there is no path connecting two points then there should not be one at $T > T_1$. This makes sure that the clusters created at different values of T form a hierarchy; every cluster found at $T = T_2$ is fully contained within some cluster created at $T = T_1 < T_2$.

The SPC algorithm uses a Potts model [35] of an inhomogeneous ferromagnet, which

can have three phases: at low values of T (low average costs), the two-point correlation is high for all pairs and all the points belong to one cluster. For high values of T the correlation between any pair is low, and each point is considered its own cluster.

If there are dense regions of points separated by low density ones, namely, there are clusters in the data, an intermediate phase appears. In this phase points that are within the same dense region are highly correlated, whereas points in the low density regions and points from different dense regions are not correlated. This phase reveals the structure in the data by grouping the points in dense regions into ordered clusters. An important issue is that the transitions that occur between the phases are due to collective behavior and are not influenced by small changes. This property ensures the robustness of the clustering solution.

$G_{ij}(T)$ is estimated using a Monte-Carlo simulation at a series of temperatures. Further details regarding the exact algorithm appear in papers by Blatt *et al.* [4][5]. Note that the estimate of $G_{ij}(T)$ (obtained by simulations) has noise added to the true value and is not guaranteed to decrease monotonically as a function of temperature and, thus, it can violate the hierarchy condition. The noise level can, however, be lowered by increasing the number of Monte-Carlo iterations.

When the clusters in the data have a distribution with a density that gradually decreases as one moves away from their cores, the peripheral points decorrelate from the core and are split off the cluster one by one, as the temperature is increased. Thus the cluster, rather than disintegrating in a single sharp transition, decreases gradually in size until only a small dense core remains. In order to overcome this problem, Blatt *et al.* suggested to add another step when creating the clusters; generating a *directed graph*. After joining all pairs with high correlation, every point is connected to that of its neighbors with which it has the highest correlation. Since points are more correlated with neighbors in the direction of the core, peripheral points become reconnected to the cluster's core.

This step does not necessarily obey the hierarchy condition, even without the noise

added by the Monte-Carlo sampling. Although the correlations between a point and its neighbors decrease monotonically with temperature, they are not guaranteed to conserve their relative order. The most correlated neighbor can switch its identity, which may cause a violation of the hierarchy condition by connecting points which were not connected at lower temperatures.

To overcome this problem we modified the original algorithm and explicitly added the hierarchy condition when creating the clusters. At each temperature we forbid connecting neighbors that were disconnected at the previous temperature.

The SPC algorithm suggests to identify the temperature range of each of the super-paramagnetic phases and take the clustering solutions as the ones generated at temperatures in the middle of these ranges. These phases are identified using the *susceptibility* graph $\chi(T)$. The susceptibility, in general, measures the fluctuations in the size of the clusters at any given temperature. In order to formally define the susceptibility, one first has to define *total magnetization* $m(\mathcal{C})$. The total magnetization is a linear function of the number of data points with the most populated color;

$$m(\mathcal{C}) = \frac{q n_{\max}(\mathcal{C}) - n}{(q-1)n} \tag{4.8}$$

where $n_{\max}(\mathcal{C}) = \max\{n_1(\mathcal{C}), n_2(\mathcal{C}), \ldots, n_q(\mathcal{C})\}$ and $n_i(\mathcal{C})$ is the number of points with color i. This definition of the magnetization is convenient since it assumes the value 1 when all proteins are of the same color (at low temperatures); and a value of 0 when all colors are evenly distributed (at high temperatures) having $n_{\max} \approx n/q$. The susceptibility is related to the variance of the magnetization at a given temperature;

$$\chi(T) = \frac{n}{T}\left(\left\langle m^2 \right\rangle - \langle m \rangle^2\right) \tag{4.9}$$

It is easy to identify the phase transition using the susceptibility since it has a peak whenever clusters break. A typical susceptibility graph goes through the following stages as the temperature is raised; at low temperatures, it starts close to zero, which represent a single correlated cluster. Then, it goes through an intermediate phase which is governed

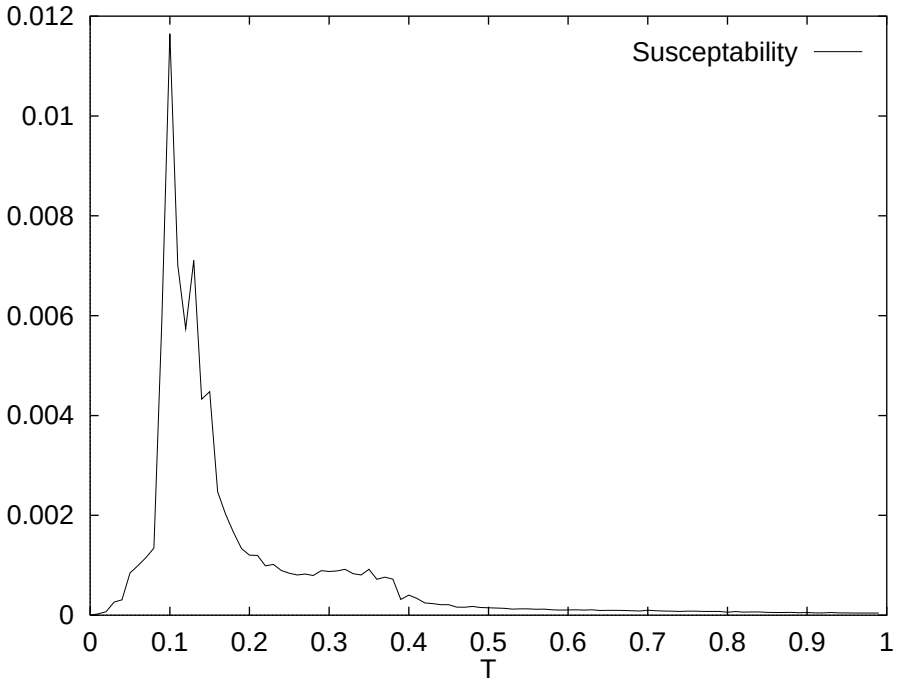


Figure 4.16: The susceptibility as a function of temperature for the SPC-K15-CZ test.

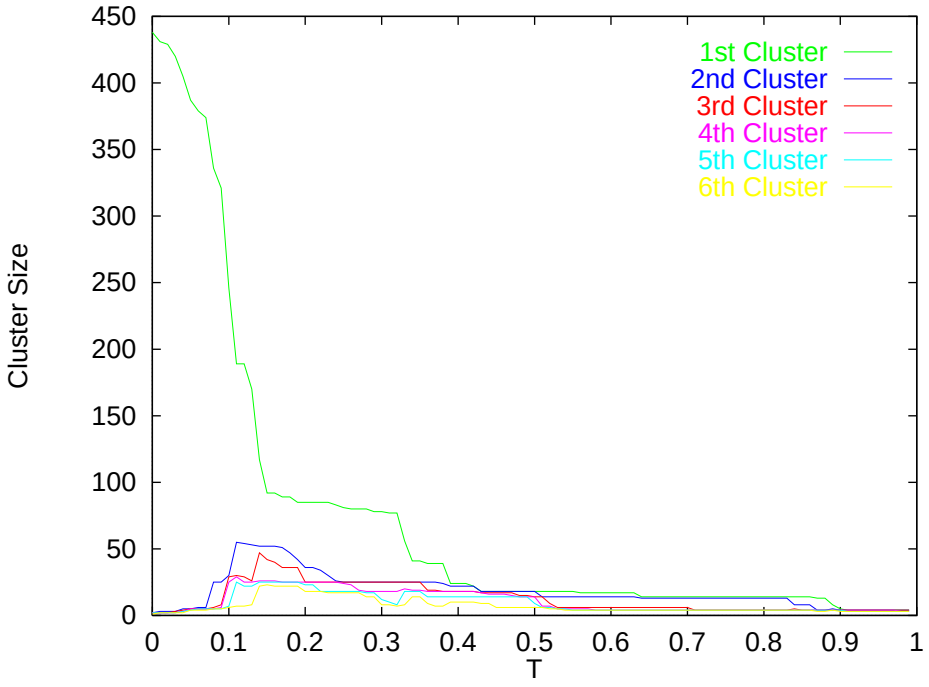


Figure 4.17: The sizes of the 7 largest clusters as a function of temperature for the SPC-K15-CZ test.

by series of peaks separated by plateaus, which represent the transitions and the steady clusterings between them, and finally drop back to zero, which represents the uncorrelated configurations of high temperatures. In figure 4.16 the susceptibility obtained in the SPC-K15-CZ test is plotted. The general behavior is as expected but the peaks are not well separated, which is an indication that the data do not form clusters that are sharply defined over a wide range of temperatures. The sizes of the 7 largest clusters as a function of temperature (for the SPC-K15-CZ test) are plotted in figure 4.17. One can see that the peak in the susceptibility corresponds to the breaking of the largest cluster. It is evident from the graph that most of the clusters are not stable for a wide range of temperatures, except a few small clusters. This rather low stability agrees with the lack distinct peaks in the susceptibility.

In order to create the hierarchy we use the calculated correlations at all temperatures, where each hierarchy level corresponds to a temperature value. Since many clusters can split between two consecutive temperature steps the resulting tree can have many offsprings at each node.

The number of colors used by the SPC, q, is not related to the number of class types. The parameter q controls the sharpness of the transitions between the different phases and is set as a compromise between sharpness and number of Monte-Carlo steps needed to reach equilibrium. We used $q = 20$.

We tested the algorithm using several similarity measures. We used the Z and CZ direct similarities and similarities created by applying the KMNV algorithm with several values of K ($K = 5, 10, 15, 20, 25, 30$). The best results of the SPC algorithm were obtained using KMNV with $K = 10$ applied on the CZ direct similarity measure. They are plotted together with the best results so far ($1NN$(CZ)) in figure 4.18. One can see that the SPC success rate is significantly lower than the $1NN$(CZ). This is due to a large number of rejections that are mostly from the second type, *i.e.* the algorithm can not decide to which class the unknown protein should be assigned, because there are many cases in which the

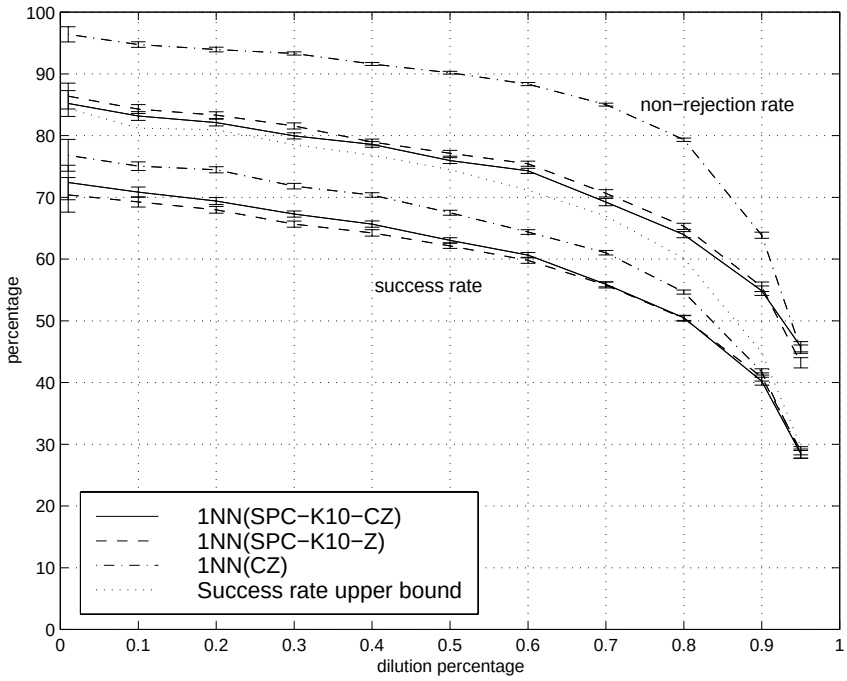


Figure 4.18: The success (lower lines) and non-rejections (upper lines) rates obtained using the best SPC similarities (using CZ and KMNV $K = 10$) and the best direct similarity $1NN(\mathrm{CZ})$.

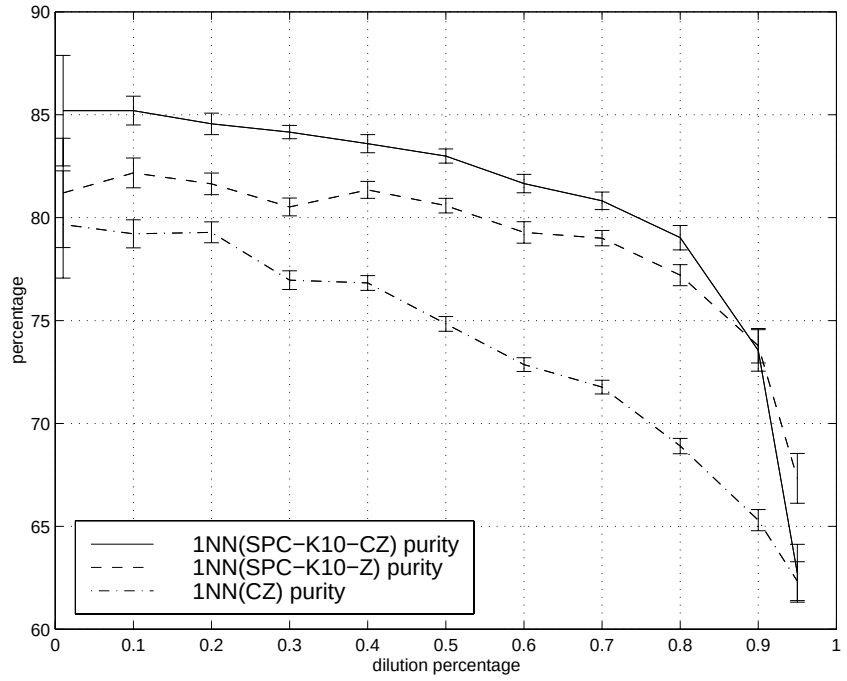


Figure 4.19: The purity obtained using the best SPC similarities (using CZ and KMNV $K = 10$) and the best direct similarity $1NN(\mathrm{CZ})$.

closest classified points belong to several different classes. The purity of the classification obtained by using the SPC results is, however, higher than that of the $1NN$(CZ), see figure 4.19.

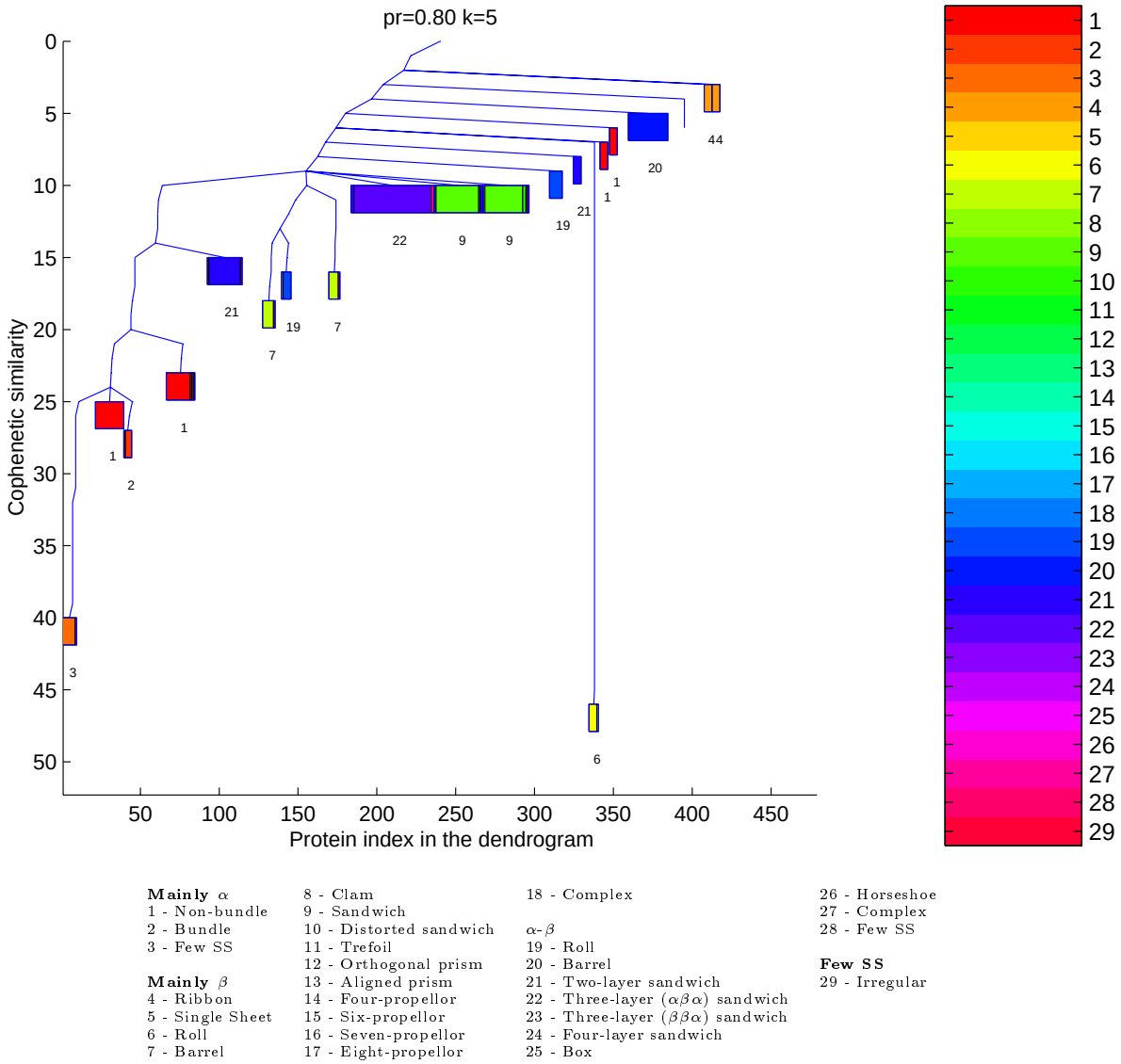


Figure 4.20: The dendrogram created by the SPC algorithm using K10-CZ similarity measure.

The resulting dendrogram for PR479 is depicted in figure 4.20. In order to produce the figure, we start from the top of the dendrogram and follow the splits. Each cluster is represented by a vertical bar colored according to the classes of the proteins in it. We do not draw clusters smaller than 5 proteins. We stop splitting the cluster when it first

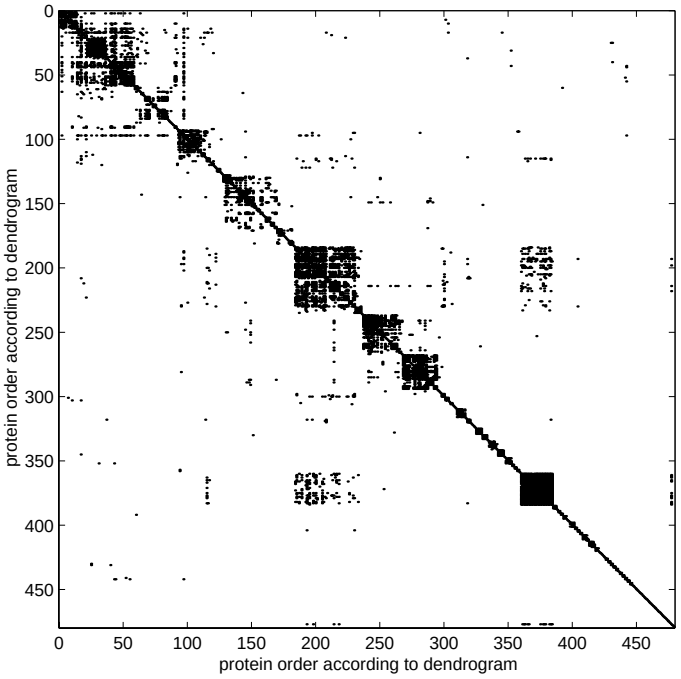


Figure 4.21: The Z-matrix ordered according to the SPC dendrogram of the previous figure.

passes a purity of 80%, meaning that more than 80% of the proteins in the cluster belong to the same class. The class index of the majority of the proteins in the cluster is written below the cluster's bar. The clusters (bars) that appear in the dendrogram represent 58.3% of the proteins, out of which 89.6% belong to the architecture of the majority in their cluster; the remaining proteins belong to small or non-pure clusters.

At each node of the tree we order the sub-clusters according to size from left to right; with leftmost offspring the largest. This order is arbitrary and is not an outcome of the clustering algorithm.

The dendrogram induces a new order on the PR479 proteins; the order can be extracted by listing the proteins in the leaves of the dendrogram from left to right. One can reorder the rows and columns of the Z-matrix according to the induced order, as was done using the original CATH order in figure 4.3. The reordered Z-matrix, which appears in figure 4.21, shows how well the clustering algorithm succeeded in grouping together closely related proteins.

As discussed previously (Subsec. 4.5.3), we compare the $1NN$(SPC-K10-CZ) results to a classification method that uses the dendrogram directly, D(SPC-K10-CZ). The direct method has a much lower success rate but a higher purity compared to the $1NN$method. At dilution of 20% the success rate of D(SPC-K10-CZ) is $49.9 \pm 0.5\%$ and the purity is $90.5 \pm 0.5\%$, whereas the performance of $1NN$(SPC-K10-CZ) is: success $68.0 \pm 0.5\%$,purity $81.6 \pm 0.5\%$.

4.5.5 Average Linkage (AVL)

The second type of hierarchical clustering algorithm we used is the average–linkage method (AVL) [19]. This method is an agglomerative clustering algorithm, meaning that it starts from n distinct clusters, one for each point, and forms the hierarchy by successively merging clusters. The agglomerative clustering methods are widely used due to their conceptual and implementational simplicity. The various agglomerative clustering methods can be described using the following scheme, known as Johnson's algorithm for hierarchical clustering [21];

Step 1. Assign each point to a different cluster $\omega_i = \{E_i\}$ and use the similarity between the points as the initial similarity matrix between the clusters $S(\omega_i, \omega_j) = S_{ij}$.

Step 2. Find the most similar pair of distinct clusters, say ω_μ and ω_ν.

Step 3. Merge clusters ω_μ and ω_ν, and call it ω_γ, where $\gamma = \min(\mu, \nu)$.

Step 4. Update the similarity matrix by deleting the rows and columns corresponding to cluster ω_ν and calculate the similarities (see below) between the new cluster ω_μ and the other clusters.

Step 5. Stop if only one cluster, that contains all the points, is left; otherwise go to **Step 2**.

Various agglomerative clustering methods differ by the way they calculate the new similarities between the joined cluster and the other clusters in **Step 4**. The AVL algorithm defines the similarity between two clusters as the average of the pairwise similarity between all pairs of points in the two clusters,

$$S(\omega_\mu, \omega_\nu) = \frac{1}{n_\mu n_\nu} \sum_{\substack{E_i \in \omega_\mu \\ E_j \in \omega_\nu}} S_{ij} \quad . \tag{4.10}$$

The AVL algorithm belongs to a sub-family of agglomerative clustering algorithms called *Sequential Agglomerative Hierarchical Non-overlapping* clustering methods (SAHN). In

these methods, the similarity between clusters can be computed using the similarities of the previous iteration of the algorithm. For example, the AVL similarity measure (Equation 4.10) can be calculated by

$$S(\omega_\mu \cup \omega_\nu, \omega_\gamma) = \frac{n_\mu}{n_\mu + n_\nu} S(\omega_\mu, \omega_\gamma) + \frac{n_\nu}{n_\mu + n_\nu} S(\omega_\nu, \omega_\gamma) \quad , \tag{4.11}$$

Calculating the new similarities using the previous ones make the SAHN algorithms relatively efficient. Since the agglomerative algorithms merge, at each iteration, two clusters, the resulting dendrogram is a binary tree. The number of iterations is always $n - 1$.

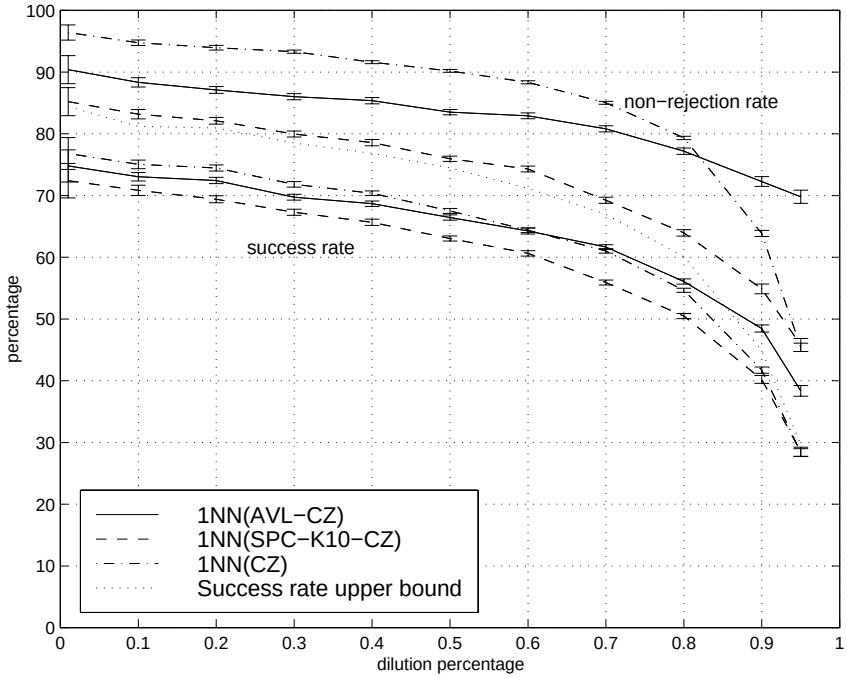


Figure 4.22: The success (lower lines) and non-rejection (upper lines) rates of the best AVL similarities (without using KMNV) compared to $1NN$(SPC-K10-CZ) and $1NN$(CZ).

The AVL algorithm applied to the Z score was used by Holm and Sander to create the dengrogram reported in the FSSP database [16]. We tested the AVL algorithm using the direct similarities Z and CZ. In addition we tested it on the outcome of the KMNV algorithm using $K = 5, 10, 15, 20, 25, 30$. The best classification results are obtained using the CZ direct similarity measure without using KMNV. ($1NN$(AVL-CZ)). The performance of using KMNV with $K > 20$ does not make much difference. In figure 4.22 we

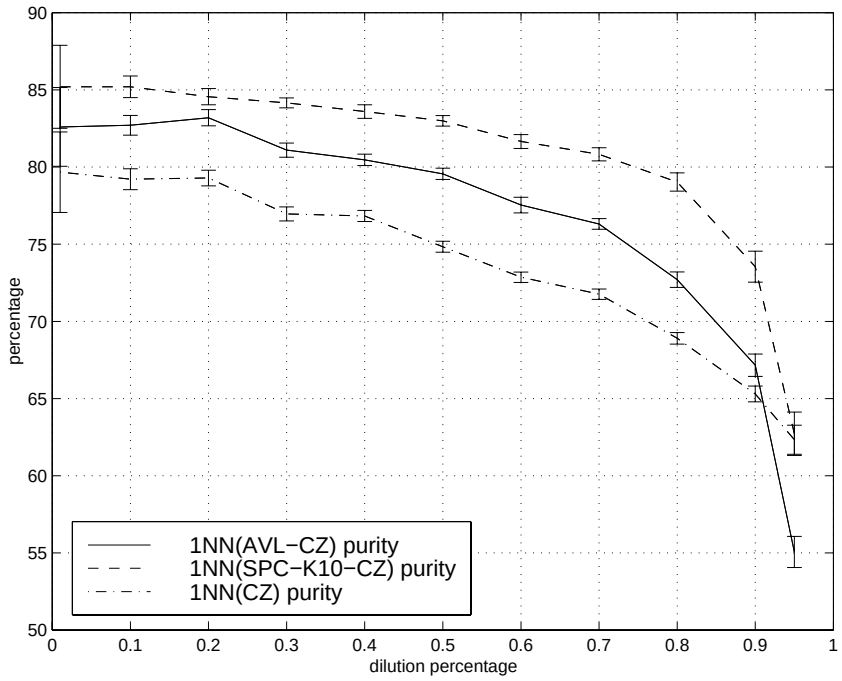


Figure 4.23: The purity obtained using the best AVL similarities (without using KMNV) compared to $1NN$(SPC-K10-CZ) and $1NN$(CZ).

plotted the evaluation of the $1NN$(AVL-CZ) compared to $1NN$(SPC-K10-CZ) and the best classification method so far $1NN$(CZ). Notice, that when using the AVL algorithm and high dilutions (dilution ≥ 90) the success rate exceeds the upper bound obtained for direct similarity measures. This is possible since the indirect similarities introduce connections between proteins that were not connected in the original direct similarities. In terms of success rate the AVL method performes better than the SPC algorithm. Testing the purity, however, shows that the SPC algorithm has a higher purity (see figure 4.23.

Figure 4.24 shows the dendrogram created by the AVL algorithm. The dendrogram is plotted using the same scheme used for the SPC dendrogram (figure 4.20), *i.e.* starting at the root and splitting until a pure enough cluster remains (80% purity), while omitting small clusters (5 points). Figure 4.25 depicts the Z-matrix reordered according to the dendrogram. In the dendrogram 64.9% of the proteins belong to large and pure enough clusters, out of them 90.0% belong to the architechture of majority in their cluster. Comparing $1NN$(AVL-CZ) to D(AVL-CZ) reveals that, as was the case for SPC, the direct

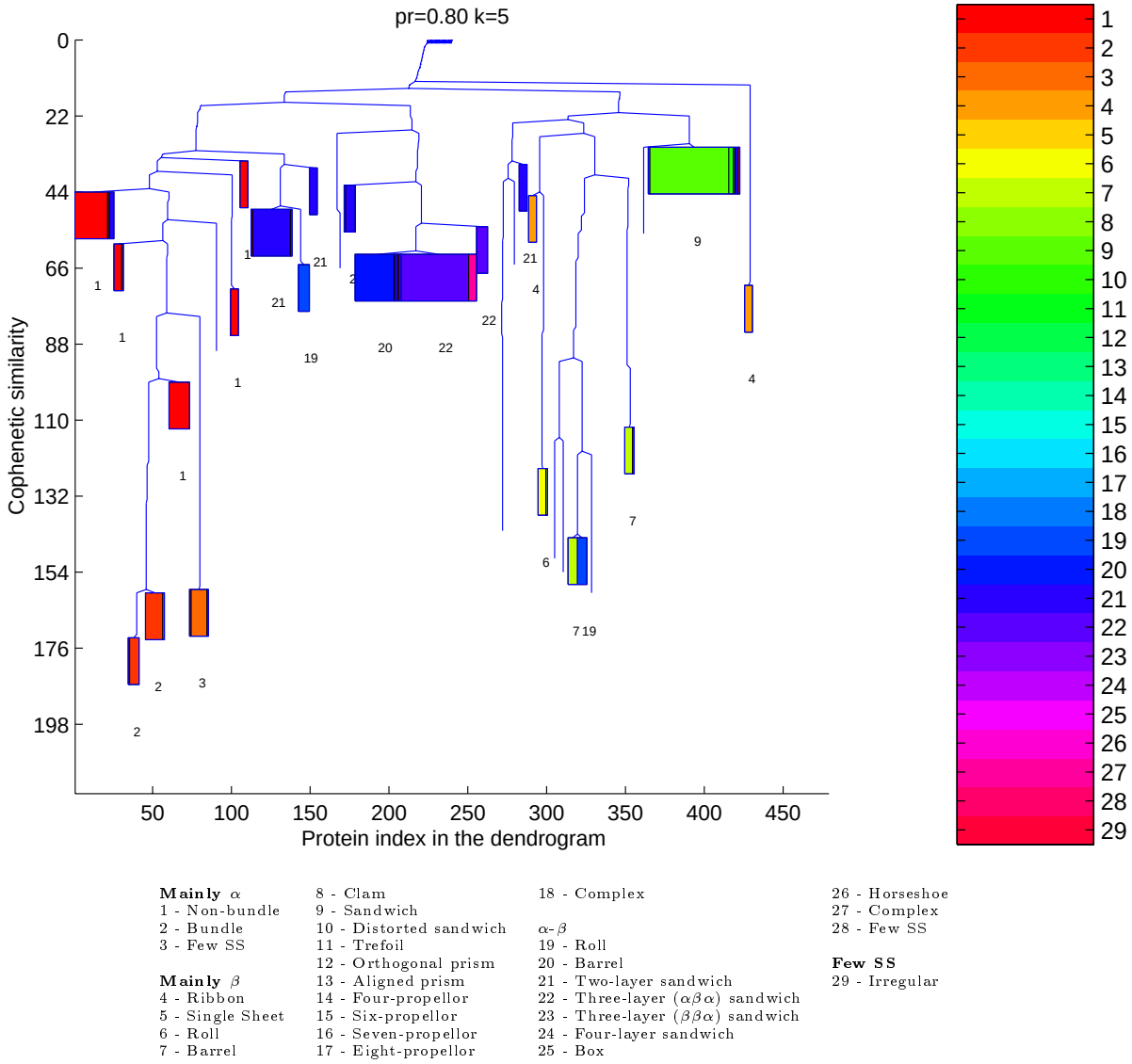


Figure 4.24: The dendrogram created by the AVL algorithm using CZ similarity measure.

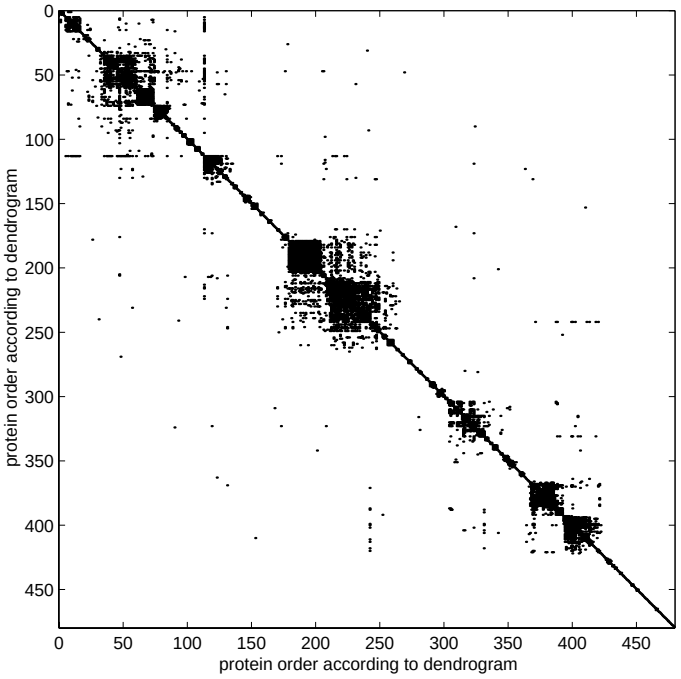


Figure 4.25: The Z-matrix ordered according to the dendrogram in previous figure.

method has lower success rate and higher purity.

One conclusion from the average–linkage analysis is that the CZ similarity measure produces a better dendrogram (for architecture classification) than the Z similarity measure. This means that using CZ instead of Z can improve the dendrogram reported in FSSP.

4.6 New heuristic approach to classification

We suggest a new classification scheme that is based on the cost function defined by the SPC algorithm (Eq. 4.5). As opposed to clustering methods, a classification algorithm can use the information from the already classified proteins and, in particular, the number of possible classes is known. Therefore, one can choose q (the number of colors in the SPC cost function) to be equal to c (the number of different classes), and set the colors of already classified proteins, which now represent the classes, according to their classification. The classification problem can now be stated as finding the minimal cost configuration of the uncolored proteins, while keeping the colors of the classified proteins fixed.

Generally, there is no efficient algorithm to find the minimal energy configuration of the unclassified proteins since it was shown that it is an NP-complete problem [6] We suggest a heuristic approach to find a low energy configuration which in some cases finds the global minimum. The heuristic is an iterative greedy algorithm. The algorithm can identify in which iteration, if at any, it performed a heuristic decision.

In order to explain the algorithm we first define *field* variables. For a given color configuration $\mathcal{C}$, one can associate with each protein, say j, a set of variables $\left\{h_j^c\right\}_{c=1}^{q}$, called *fields*. The field h_j^c measures the cost gained by assigning protein j to color c while keeping all other colors fixed;

$$h_j^c = \sum_i S_{ij}\delta(c_i, c) \quad . \tag{4.12}$$

Note that only proteins, i, that are neighbors of j contribute to the sum. In case all proteins but one have known classification, the minimal cost is obviously reached by assigning the unclassified protein to the class of its maximal field. Notice that the field variables attached to protein j are independent of its color c_j.

In case several proteins have no known classification, the field variables can be uncertain, since the field contribution from unclassified neighbors is unknown. Therefore we define an additional variable for each protein, which we call *unknown field* h_j^{unknown}; that

is, the total field from unclassified neighbors;

$$h_j^{\text{unknown}} = \sum_{\substack{i \\ i \text{ is unclassified}}} S_{ij} \quad . \tag{4.13}$$

The *maxfield* (MF) algorithm, discussed in Section 4.3.2, assigns unclassified proteins to the class of their maximal field regardless of the unknown field (this clarifies the origin of the algorithm). Only classified neighbors of the unclassified protein take part in the classification. This method obviously suffers at high dilution, where the unknown field can be very large compared to all other fields.

The algorithm we suggest here takes the unknown field into account. For each protein we identify the maximal field (induced by the neighbors of known color) $h_i^{\max} = \max_c h_i^c$ and maximal field color $c_i^{\max} = \arg\max_c h_i^c$. In addition, we identify the second highest field $h_i^{\text{next}} = \max_{c \neq c_i^{\max}} h_i^c$. We define a variable Δ_i that measures the difference between the maximal field and the next to maximal field,

$$\Delta_i = h_i^{\max} - h_i^{\text{next}} \quad . \tag{4.14}$$

Finally, we measure the ratio between Δ_i and the unknown field, $r_i = \Delta_i / h_i^{\text{unknown}}$. It is clear that if $r_i > 1$ the minimal cost configuration has protein i assigned to the class of its current max field, independent of the classes of its unclassified neighbors. Even if all unclassified neighbors will belong to the class of the next highest field, their contribution will increase the next highest field by the unknown field and it will still not exceed the current maximal field. Therefore, assigning a protein with $r_i > 1$ to the class $c_i^{\max}$ is a "safe" step. On the basis of this idea we suggest the following algorithm:

Step 1. Calculate the ratios for the unclassified proteins.

Step 2. Find the protein with maximal ratio.

Step 3. Assign the protein to the class of its maximal field.

Step 4. Stop when all unclassified proteins have an assignment, otherwise go to **Step 1**.

Notice that in each iteration one should update the ratios of the remaining unclassified proteins since new classifications alter the fields. One can view this algorithm as gradually classifying points along the border between the classified and unclassified regions, this corresponds to classifying proteins with ratio > 1. At a certain iteration, the highest ratio drops below 1, and only the core (or cores) of the unclassified proteins remain. The classification of these proteins depend on each other and at this stage a *heuristic* step is performed by taking the "safest" decision. Later, after updating the fields, it is possible that some ratios will exceed 1. This process continues until there are no unclassified points left.

The classification of each protein, performed using this algorithm, depends on all the heuristic steps that were performed prior to its classification. At low dilutions it is common that the ratios of all unclassified proteins are greater than 1, meaning that the classification performed by the algorithm is the global minimum of the cost function.

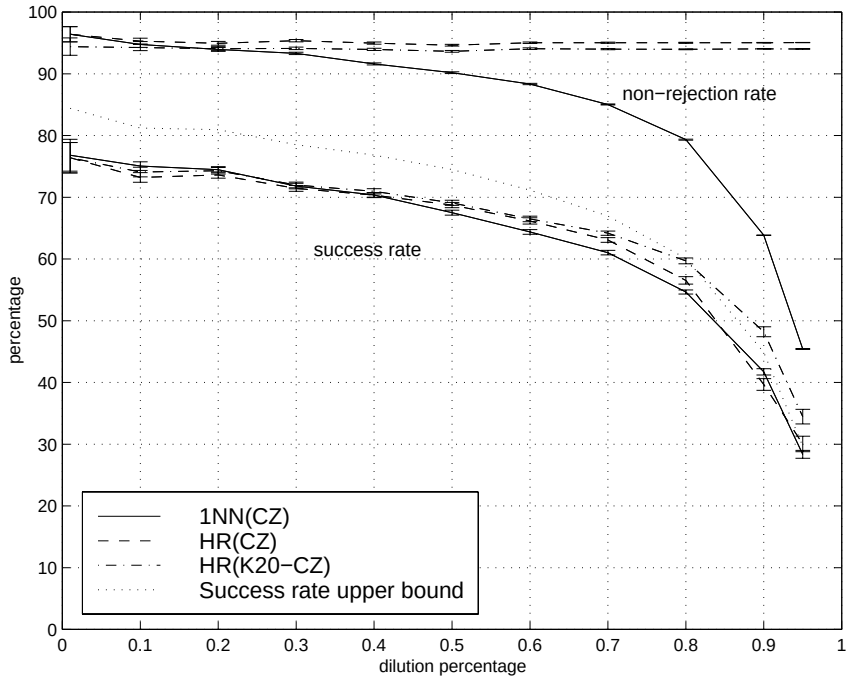


Figure 4.26: The HR success (lower lines) and non-rejection (upper lines) rates using CZ and K20-CZ similarity measures compared with $1NN$(CZ).

We tested the heuristic classification algorithm (HR) using the direct similarities Z and

CZ and also using the outcome of the KMNV algorithm using $K = 5, 10, 15, 20, 25, 30$. The best performance was obtained using $K = 20$. Figure 4.26 plots the performance of the heuristic algorithm using CZ and K15-CZ similarity measures, HR(CZ) and HR(K20-CZ), compared with the best classification so far $1NN$(CZ). One can see that at high dilutions the classification using HR is better than $1NN$(CZ) and that HR(K20-CZ) even exceeds the upper bound for direct similarities. This is possible since the heuristic algorithm uses indirect considerations in the classification. Another important property of the HR algorithm is that it yields a rejection only for proteins that have no neighbors (classified or not). This causes the purity of the algorithm to decrease as the dilution rate is increased.

The classifications of the HR algorithm coincide with those given by the MF algorithm at low dilutions. This is clear since at low dilutions unclassified proteins are rarely neighbors of each other. At high dilutions, on the other hand, the HR algorithm can use the relationships between unclassified proteins to reach a better classification.

Chapter 5

Prediction

In this Chapter we summarize the results of the classification methods presented in the previous Chapter and select the methods we use to classify the set PR165 (Sec. 5.1). We report our predicted classifications in Sec. 5.2. Since we do not know the true classification of the PR165 proteins, we can not check the algorithm's performance. We intend to publish the results and let researchers, from CATH or other groups, check our prediction.

5.1 Selecting the Classification Method

We summarize the performance of the classification methods presented in the previous Chapter. We tested several similarity measures; direct similarity measures that included the original S,Z and our normalized ones CS,CZ. In addition, we tested indirect measures that were created using KMNV, SPC and AVL algorithms. We used combinations of these measures using $1NN$, maxfield (MF) and the heuristic (HR) classification methods. The clustering methods were also tested using the direct (D) classification method. Each combination was evaluated using a cross-validation scheme at different dilutions.

Since we want to select a classification method for predicting the PR165 proteins out of the PR644 proteins (PR479 together with PR165), we are generally interested in methods that perform well at dilution of around 25% (165/644). Since, however, the dilution is a global parameter, that is, the average of local dilutions surrounding the yet unclassified

proteins, we seek an algorithm that performs well in a range of dilutions. A further complication in selecting the classification method is that for each method we estimated the success rate and purity, and we do not have a single parameter to maximize. We avoid making the tradeoff between success rate and purity since it depends on the specific use of the prediction. If one wants to use the prediction without any human intervention it is preferable to have higher success rate, since the success rate estimates the probability of correctly predicting the architecture. In case one has a semi-automatic process where only the rejected proteins are examined by humans and the non-rejected ones are classified as predicted, a higher purity is preferable since it estimates the probability of correct predictions within the non-rejected proteins. Therefore, we study the properties of the tested methods and give the prediction at several selected working conditions.

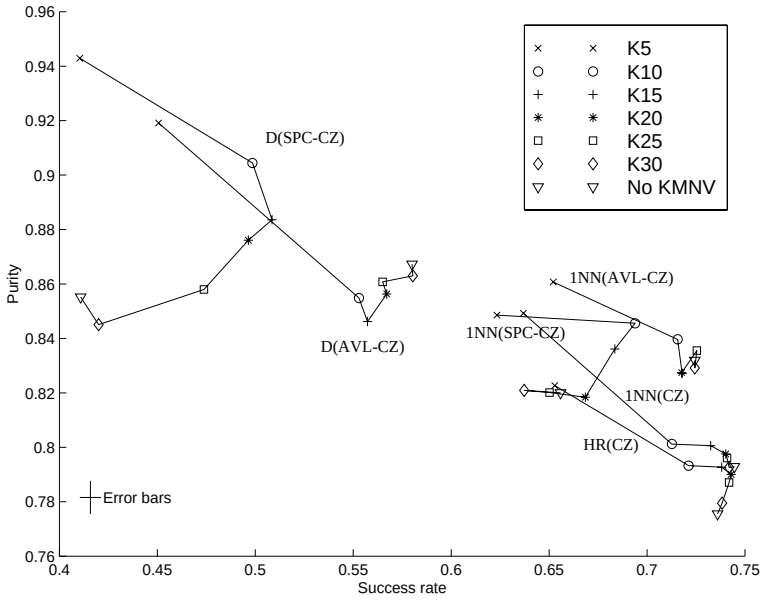


Figure 5.1: Different classification methods plotted on the success rate/purity plane at dilution 20%.

On the basis of the results presented in Chapter 4 (see figure 4.7), we chose the CZ direct similarity measure, since it improves the performance of all indirect measures and classification methods. The last that is left is to compare the performance of the different algorithms and indirect similarities based on the CZ measure. After removing

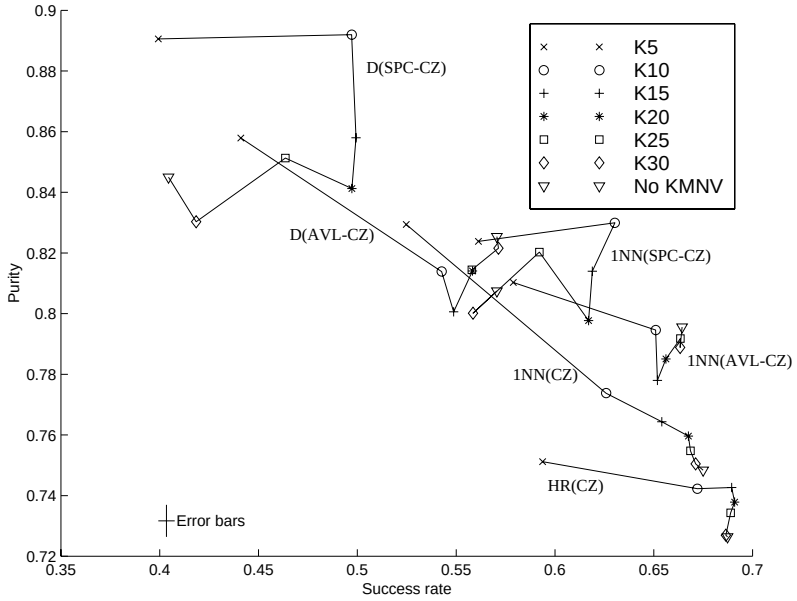


Figure 5.2: Different classification methods plotted on the success rate/purity plane at dilution 50%.

all methods that were inferior in both success rate and purity, 6 algorithms remain; $1NN$(CZ), HR(CZ), $1NN$(AVL-CZ), $1NN$(SPC-CZ), D(SPC-CZ) and D(AVL-CZ). Each of the similarities can be preprocessed by the KMNV algorithm using different K values.

Figures 5.1 and 5.2 plot the success rate and purity of the tested methods at 20% and 50% dilution rates respectively. Each algorithm is represented by a line which is parameterized using the K parameter of the KMNV algorithm. Each point is marked by a symbol that corresponds to a specific K value (and one corresponds to not using KMNV). In order to make the plot clearer we plotted the error bars, which are roughly the same for all points, in the lower-left corner.

Using these graphs we select the K values and algorithms that we use for predicting the PR165 architectures. It is clear that a point which has higher success rate and purity is preferable. Graphically, it means that a mark in the graph, which represents an algorithm and K value, performs better than those that are marked lower and to the left of it. From figure 5.1 we can see that at low dilutions the highest success rate is obtained by the $1NN$(CZ) and HR(K20-CZ) algorithms. Lower success rates with higher

purity is obtained by the $1NN$(AVL) using $K \leq 25$. Much higher purities are obtained using the D(SPC) but at the cost of much lower success rates. Moving to higher dilutions (figure 5.2) shows that the HR(K20-CZ) reaches the highest success rate and does no longer behave like the $1NN$(CZ). At this dilution the $1NN$(SPC-K10-CZ) has high purity with reasonable success rate. D(SPC) maintains its high purity and achieves its highest success rate at $K = 10$.

We selected to use for the prediction the following 5 algorithms; $1NN$(CZ), HR(K20-CZ), $1NN$(AVL-CZ), $1NN$(SPC-K10-CZ) and D(SPC-K10-CZ). All of these are the most upper-right points of each algorithm which define the best performance envelope. Table 5.1 lists the success rate and purity of each of the selected algorithms.

method	dilution	success rate	non-rejection rate	purity
$1NN$(CZ)	20%	$74.5 \pm 0.5\%$	$93.9 \pm 0.4\%$	$79.3 \pm 0.5\%$
	50%	$67.5 \pm 0.4\%$	$90.2 \pm 0.2\%$	$74.8 \pm 0.4\%$
HR(K20-CZ)	20%	$74.3 \pm 0.5\%$	$94.1 \pm 0.3\%$	$79.0 \pm 0.6\%$
	50%	$69.1 \pm 0.4\%$	$93.6 \pm 0.1\%$	$73.8 \pm 0.4\%$
$1NN$(AVL-CZ)	20%	$72.4 \pm 0.5\%$	$87.1 \pm 0.5\%$	$83.2 \pm 0.5\%$
	50%	$66.4 \pm 0.4\%$	$83.5 \pm 0.4\%$	$79.6 \pm 0.4\%$
$1NN$(SPC-K10-CZ)	20%	$69.4 \pm 0.6\%$	$82.1 \pm 0.5\%$	$84.6 \pm 0.5\%$
	50%	$63.0 \pm 0.4\%$	$76.0 \pm 0.4\%$	$83.0 \pm 0.3\%$
D(SPC-K10-CZ)	20%	$49.9 \pm 0.5\%$	$55.2 \pm 0.5\%$	$90.4 \pm 0.5\%$
	50%	$49.7 \pm 0.4\%$	$55.8 \pm 0.5\%$	$89.2 \pm 0.4\%$

Table 5.1: Success rates, non-rejection rates and purities of the selected algorithms at dilutions 20% and 50%.

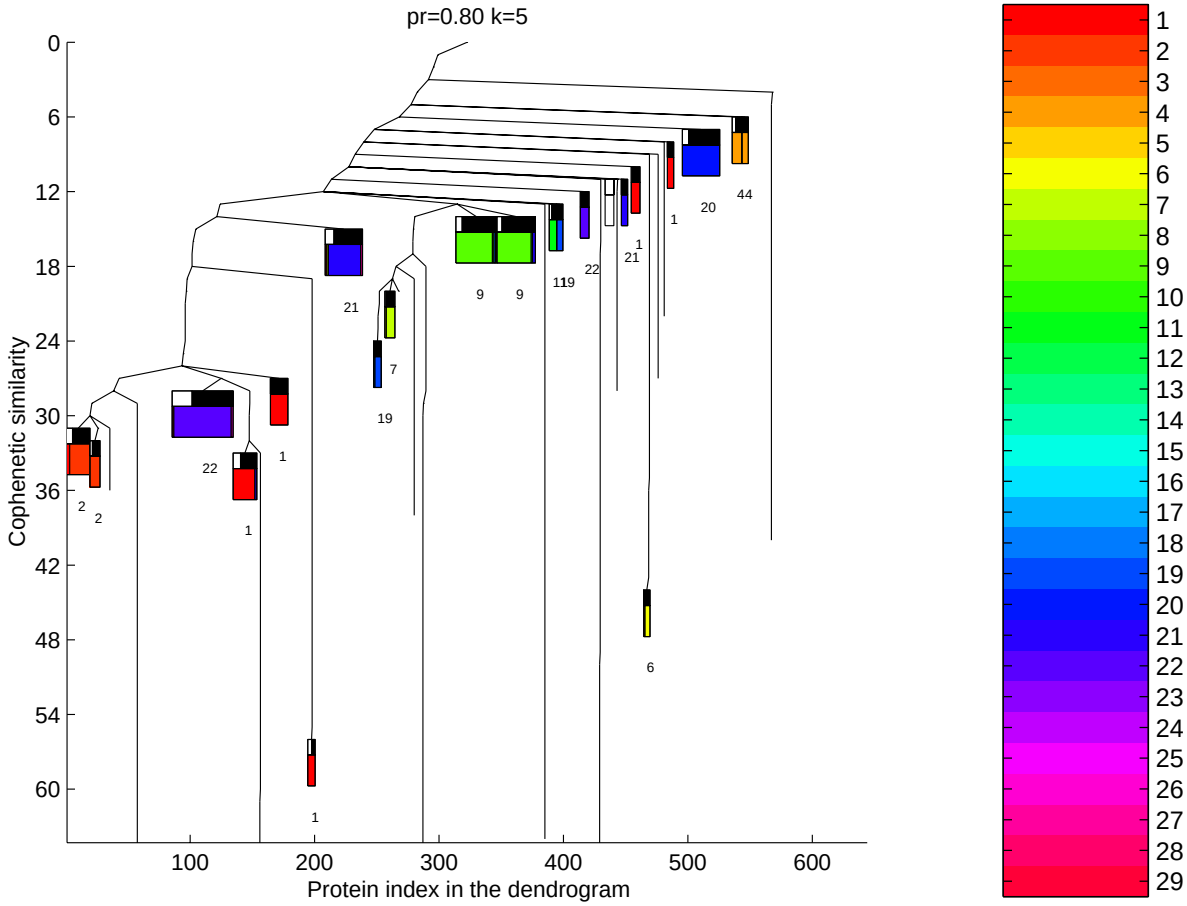
Viewing the figures, one can notice that the algorithms have different K dependence; the $1NN$(CZ) behaves in a monotonic manner, as K is increased the success rate increases and the purity decreases and large K values do not have much influence on the performance. The $1NN$(SPC-CZ), on the other hand, reaches the maximal success rate at an intermediate K value ($K = 10$).

In general, it is expected that as K decreases the classification has lower success rates and higher purity since the KMNV algorithm removes "inconsistent" connections, thus

keeping, with high probability, ones between proteins of the same class, but also leaving many proteins without any neighbors, which increases the rejection rate (and decreases the success rate). In the SPC case we have not studied the reasons for this dependency but we suggest that the KMNV helps to even out the number of connections a protein has. A large deviation in the number of connections between different regions in the data can change the SPC's behavior.

In order to appreciate the relations between the PR644 (PR479 together with PR165) proteins we use the SPC and AVL clustering algorithms to produce a dendrogram of the data. Figures 5.3 and 5.4 depict these dendrogram created using SPC-K10-CZ and AVL-CZ, respectively. In each cluster plotted in the dendrogram there are two horizontal bars; the top one, represents the number of classified (black) and unclassified proteins (white) in that cluster. The bottom bar represents the class distribution within the classified proteins in the cluster.

Viewing the dendrograms one can see that both algorithms find two separate "branches" surrounded by small clusters; one branch includes architectures 1,2,3,21,22, which are the mainly-α proteins together with two α-β architectures that include many α-helices. The other branch includes architectures 7,9,19 which are mainly-β Roll and Sandwich joined with α-β Roll. The AVL algorithm separates the first branch into mainly-α and α-β . The SPC algorithm, however, tends to seperate the data into smaller and purer clusters. One can see that the yet unclassified proteins are distributed among the clusters and many of them (40% for the SPC and 60% for the AVL) are part of pure clusters. This strongly suggests that they too belong to the architectures of these clusters. We conclude that both algorithms find non-trivial structure in the data that corresponds to the architectures defined by human observers.



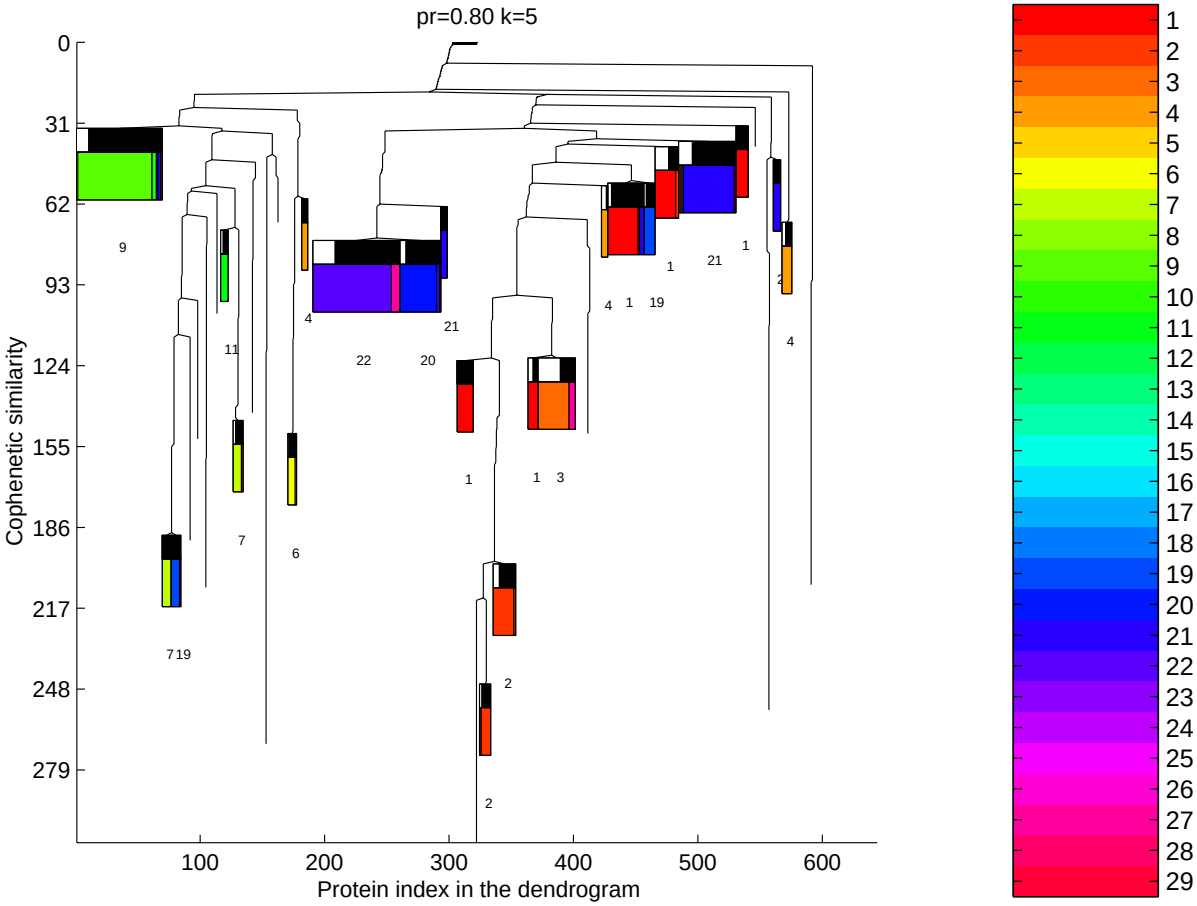
Mainly α
1 - Non-bundle
2 - Bundle
3 - Few SS

Mainly β
4 - Ribbon
5 - Single Sheet
6 - Roll
7 - Barrel
8 - Clam
9 - Sandwich
10 - Distorted sandwich
11 - Trefoil
12 - Orthogonal prism
13 - Aligned prism
14 - Four-propellor
15 - Six-propellor
16 - Seven-propellor
17 - Eight-propellor
18 - Complex

α-β
19 - Roll
20 - Barrel
21 - Two-layer sandwich
22 - Three-layer ($\alpha\beta\alpha$) sandwich
23 - Three-layer ($\beta\beta\alpha$) sandwich
24 - Four-layer sandwich
25 - Box
26 - Horseshoe
27 - Complex
28 - Few SS

Few SS
29 - Irregular

Figure 5.3: The dendrogram created by the SPC algorithm using K10-CZ similarity measure between the PR644 proteins.



Mainly α
1 - Non-bundle
2 - Bundle
3 - Few SS

Mainly β
4 - Ribbon
5 - Single Sheet
6 - Roll
7 - Barrel
8 - Clam
9 - Sandwich
10 - Distorted sandwich
11 - Trefoil
12 - Orthogonal prism
13 - Aligned prism
14 - Four-propellor
15 - Six-propellor
16 - Seven-propellor
17 - Eight-propellor
18 - Complex

α-β
19 - Roll
20 - Barrel
21 - Two-layer sandwich
22 - Three-layer ($\alpha\beta\alpha$) sandwich
23 - Three-layer ($\beta\beta\alpha$) sandwich
24 - Four-layer sandwich
25 - Box
26 - Horseshoe
27 - Complex
28 - Few SS

Few SS
29 - Irregular

Figure 5.4: The dendrogram created by the AVL algorithm using CZ similarity measure between the PR644 proteins.

5.2 Prediction

In this section we report our architecture prediction for the PR165 proteins. Tables 5.2 to 5.6 list all 165 proteins with their prediction using the selected methods (rejections appear as blanks). Each protein is specified by its PDB entry and chain identifier, and the architectures are given by their index (see table 3.1). Further details about each protein can be found in the PDB web site [27]. We ordered the proteins according to the architectures predicted by the HR(CZ) algorithm, since it had no rejections. At the top of each prediction column we stated the purity of that algorithm as was estimated on PR479 at dilution 20%. This gives an estimate of what is the probability that a prediction written in that column is a correct one. The total number of predictions appear at the bottom of each column, and the overall total appears at the bottom of the table 5.6.

One can see that the correlation between the different methods is very high; for 46 out of 165 (28%) all the methods yield the same prediction (meaning also that these proteins are not rejected). In 128 out of the 165 (77.6%) all the non-rejection predictions are the same. The remaining 37 proteins usually have a majority voting for a certain architecture.

number	PDB entry	$1NN$(CZ)	HR (K20-CZ)	$1NN$ (AVL-CZ)	$1NN$(SPC-K10-CZ)	D(SPC-K10-CZ)
	purity	$79.3 \pm 0.5\%$	$79.0 \pm 0.6\%$	$83.2 \pm 0.5\%$	$84.6 \pm 0.5\%$	$90.4 \pm 0.5\%$
1	1fct	1	1			
2	1hph	1	1			
3	1iba	1	1	1	1	
4	1ktx		1			
5	1thjA	1	1		1	1
6	2bpa3		1			
7	2mrb		1			
8	1tmf4		1			
9	1vnc	2	1	2		
10	1ron	1	1		1	1
11	1ctdA	1	1	1	2	
12	1dmc	1	1			
13	1eciA	1	1	22	1	1
14	1lbd	1	1			
15	1ponA	1	1	1		
16	1ponB	1	1	1		
17	1ppbL	1	1		1	
18	1tfe	2	1	1	1	
19	1tfr	1	1	1	1	21
20	1c53	1	1	1	1	1
21	1pcl	22	1			
22	1occE	1	1	26	1	1
23	1occH	3	1	1		
24	1tafA	1	1	1	1	
25	1tafB	1	1	1	1	
26	1mhu		1			
27	1cem		1			
28	1xsm	1	1	1	1	
29	1roo		1			
30	1bgk		1			
31	1pueE	1	1	1	1	
32	1fre		1			
33	1cpo	1	1	1		
	total	24/33	33/33	16/33	14/33	6/33

Table 5.2: Prediction of PR165 - Part 1.

number	PDB entry	$1NN$(CZ)	HR (K20-CZ)	$1NN$ (AVL-CZ)	$1NN$(SPC-K10-CZ)	D(SPC-K10-CZ)
	purity	$79.3 \pm 0.5\%$	$79.0 \pm 0.6\%$	$83.2 \pm 0.5\%$	$84.6 \pm 0.5\%$	$90.4 \pm 0.5\%$
34	1pbwA	2	1		1	
35	1cei	2	1		1	1
36	1nox	1	1	21	1	
37	1lefA	1	1	1	1	1
38	1lrv	26	1	26	1	1
39	1zwa	2	1			
40	1zwb	1	1			
41	1zwc	1	1			
42	1fow	1	1	1	1	1
43	1zwd	1	1			
44	1zwe	1	1			
45	1jvr	1	1			
46	1ery	2	2	2		
47	2brd	2	2	2	2	2
48	1rmi	2	2	2	2	2
49	1hmcB	2	2	2	2	2
50	1bmfG	3	2	3	3	
51	1dkzA	3	2	2	2	2
52	1occA	2	2	2	2	2
53	1occC	2	2	2	2	2
54	1higB	1	2	1		
55	1lre	3	2	2	2	2
56	1hdj	3	3	22	3	
57	1ifi	3	3	3	3	
58	1ppt	3	3			
59	1bba		3			
60	1tiiC	3	3	3	3	
61	1gcmA	3	3	3	3	
62	1fosE	3	3	3	3	
63	1fosF	3	3	3	3	
64	1lyp	3	3	3	3	
65	1mof	3	3	3	3	
66	1psm	3	3	3	3	
	total	32/33	33/33	23/33	23/33	11/33

Table 5.3: Prediction of PR165 - Part 2.

number	PDB entry	$1NN$(CZ)	HR (K20-CZ)	$1NN$ (AVL-CZ)	$1NN$(SPC-K10-CZ)	D(SPC-K10-CZ)
	purity	$79.3 \pm 0.5\%$	$79.0 \pm 0.6\%$	$83.2 \pm 0.5\%$	$84.6 \pm 0.5\%$	$90.4 \pm 0.5\%$
67	1spf	3	3	3	3	
68	1wfbA	3	3	3	3	
69	1occD	3	3			
70	1occI	3	3	3	3	
71	1occG	2	3			
72	1occJ	3	3	3	3	
73	1occK	3	3	3	3	
74	1occL	3	3	3	3	
75	1occM	3	3	3	3	
76	1peh	3	3	3	3	
77	2ifm	3	3	3	3	
78	4ifm	3	3	3	3	
79	1edmB	4	4	4	4	4
80	2cbh	4	4	4	4	
81	1hleB	7	4	4	4	
82	7apiB	4	4	4	4	
83	1emn	4	4	4	4	4
84	1pfxL	4	4	4	4	4
85	1pft	5	5	5	5	
86	1bnb	5	5	5	5	
87	1whi	6	6	6	6	
88	1mai	6	6	6	6	
89	1lepA	6	6	6	6	
90	1qbeA	21	7	21	21	
91	1tiiD	7	7	7		
92	1fmb	7	7	7	7	
93	1ghj	7	7	7		
94	1znbA	22	7			
95	1pfsA	7	7	7		
96	1iyu	7	7	7		
97	1cwpA	9	9	9	9	9
98	1ahsA	9	9	9	9	9
99	1wkt	9	9	9	9	9
	total	33/33	33/33	30/33	26/33	6/33

Table 5.4: Prediction of PR165 - Part 3.

number	PDB entry	$1NN$(CZ)	HR (K20-CZ)	$1NN$ (AVL-CZ)	$1NN$(SPC-K10-CZ)	D(SPC-K10-CZ)
	purity	$79.3 \pm 0.5\%$	$79.0 \pm 0.6\%$	$83.2 \pm 0.5\%$	$84.6 \pm 0.5\%$	$90.4 \pm 0.5\%$
100	1mspA	9	9	10	9	9
101	1occB	9	9	9	9	9
102	1occF	9	9			
103	1wit	9	9	9	9	9
104	1lcl	9	9	9	9	9
105	1stmA	9	9	9	9	9
106	1tul	10	9	10	9	9
107	1dec	10	10	10	10	
108	1dutA	10	10	10	10	
109	2ila	11	11	11	11	11
110	1wba	11	11	11	11	11
111	1pmc	14	14		14	
112	1npoA	17	17			
113	1ema	7	19	19	7	7
114	1cby	21	19			
115	1alo	19	19	19	1	1
116	1lit	19	19	19	19	
117	1cb2A	20	20	20	20	20
118	1eceA	20	20	20	20	20
119	1dhpA	20	20	20	20	20
120	1nsj	20	20	20	20	20
121	1onrA	20	20	20	20	20
122	1pnh	21	21	21		
123	4cpaI	4	21			
124	1tys	21	21	19	21	21
125	1sis	21	21	21		
126	2fua	21	21	21	21	21
127	1mli	21	21	21	21	21
128	1hqi	21	21	21	21	21
129	1vhiA	21	21	21	21	21
130	1fleI		21			
131	1pytA	21	21	21	21	21
132	1far	21	21	21	21	
	total	32/33	33/33	27/33	26/33	21/33

Table 5.5: Prediction of PR165 - Part 4.

number	PDB entry	$1NN$(CZ)	HR (K20-CZ)	$1NN$ (AVL-CZ)	$1NN$(SPC-K10-CZ)	D(SPC-K10-CZ)
	purity	$79.3 \pm 0.5\%$	$79.0 \pm 0.6\%$	$83.2 \pm 0.5\%$	$84.6 \pm 0.5\%$	$90.4 \pm 0.5\%$
133	1sco	21	21	21		
134	2crd	21	21	21		
135	1din	22	22	22	22	22
136	1frvA	22	22	22	22	22
137	1frvB	4	22	4		
138	1gdoA	22	22	22	22	
139	1hgxA	22	22	22	22	22
140	1qrdA	22	22	22	22	22
141	1rnl	22	22	22	22	22
142	1kte	22	22	22	22	22
143	1cydA	22	22	22	22	22
144	1enp	22	22	22	22	22
145	1srsA	24	22	22		
146	1efm	22	22	22	22	22
147	1pmaA	22	22	21		
148	1pmaP	21	22	21		
149	1broA	22	22	22	22	22
150	1jud	2	22	22	22	22
151	1rie	22	22			
152	1rvv1	22	22	22	22	22
153	1apyA	22	22			
154	1apyB	21	22	21		
155	1mil	22	22	22	22	
156	3monG	16	22	4		
157	1cfr	22	22	22	22	22
158	1xvaA	22	22	22	22	22
159	1xel	22	22	22	22	22
160	1fds	22	22	22	22	22
161	1kuh	22	22	22	22	
162	1kinA	22	22	22	22	22
163	1zfd	21	28	21	21	
164	1gur	29	29	29	29	
165	1eit	29	29	29	29	
	total	33/33	33/33	31/33	23/33	17/33
	TOTAL	154/165	165/165	127/165	112/165	61/165

Table 5.6: Prediction of PR165 - Part 5.

Chapter 6

Summary

This chapter summarizes the work we performed regarding prediction of protein architectures. As described in the Introduction (Sec. 2.2), we tried to automatically predict the CATH architecture of a set of proteins that were not yet processed by CATH, using the similarity measures taken from the FSSP database. The architecture classification in the CATH database is performed manually. As far as we know this is the first attempt to correlate the FSSP similarity measures with the CATH classification.

We performed the classification using several standard and newly suggested techniques and gave predictions for 165 yet unclassified proteins (see Sec. 5.2). The classification algorithms were tested and evaluated on an already classified set of proteins. Reliability of 80% and above (depending on the preferred rejection rate) was reached. Using these automatic classification methods one can easily enlarge the CATH database with less human effort, or none at all.

We tested several techniques for the classifications task. We first used standard nearest–neighbor type algorithms using the original similarity measures taken from the FSSP database. Next we introduced an improved similarity measure based on local normalization of the original measures (CS and CZ). Using this measure improved the performance of all the classification methods that were tested.

A second type of similarity measures, which we called *indirect*, were calculated using the outcome of hierarchical clustering techniques. We use the clustering techniques in

order to introduce global relationships between the proteins into the local similarity measures. When classifying using the indirect similarities, we encountered many rejections and, therefore, we could not increase the overall success rate. On the other hand the classification purity (the success rate within the non-rejected proteins) increased by 10%.

We tested two clustering techniques; one is the average–linkage algorithm (AVL), which is a widely used hierarchical clustering method and the super paramagnetic clustering method (SPC) which is a more robust clustering method introduced recently. We had to modify the SPC algorithm in order to use it as a hierarchical clustering method.

Finally, we suggested a heuristic approach (HR) to the classification problem which searches for a global minimum of the SPC cost function. This method uses relationships between all proteins, much like indirect similarity measures, in order to perform the classification. Using this method we succeeded to improve the correct prediction rate.

We made predictions using 5 classification methods, all based on the CZ similarity measure. Each of the methods has different success and rejection rates. As a byproduct of the classification we identified a list of proteins, which are already classified by CATH, that we suspect as misclassifications (see table 4.2). We intend to publish our predictions and let researchers from the CATH, or other groups, examine them.

On the basis of the work presented here, further research can be conducted in several directions; in the biological direction, one can use automatic algorithms to separate multi-domain proteins into domains, calculate the CZ score between them and apply the architecture prediction to each of the domains. Such a procedure would make the classification methods presented here applicable for all proteins. Regarding the classification and clustering techniques, one can study the different K behavior of the algorithms and suggest ways to find the optimal one. Furthermore, one can search for improved algorithms to find the global minimum of the SPC cost function and ways to alter the cost function so that it will better reflect the true correlations between the proteins.

Appendix A

The FSSP Database

In this section the FSSP database and similarity measure are described based on papers published by Holm and Sander [15][18][16] and on the FSSP web site **http://www.ebi.ac.uk/dali/fssp**. The FSSP database includes all protein chains from the Protein Data Bank that are longer than 30 residues. The chains are identified using the PDB entry code plus a chain identifier. A representative set is selected out of sequence homologies, which have more than 25% sequence identity. Above 25% sequence identity assures very similar structure and thus only one representative from each of the sequence homologies is needed for fold classification. Within the representatives there are still many similar structures despite the low sequence similarity. The FSSP version from December 25, 1997, which was used in this work, included 1188 representatives out of 9153 chain structures that were available in the PDB (the latest update from August 15, 1998 included 1371 representatives out of 11,912 structures).

For the representatives, an all-against-all structural comparison is done and all pairwise similarity measures are calculated. The similarity measure is statistically calibrated and is given in terms of Z-score (standard deviations above the mean). The results of the exhaustive comparison are reported as a fold tree (dendrogram) created by an average linkage hierarchical clustering algorithm. The fold tree divides the representatives into a series of sets at different structural similarity levels.

For each representative there is a file containing structure-structure alignments and

(a) 1bcfA (b) 1lki

Figure A.1: The alignment of 1bcfA and 1lki. Aligned residues of the proteins are red and unaligned are blue. The calculated Z-score for them is 7.3.

similarity score with its neighbors in the representative set and its sequence homologs in the PDB. Figure A.1 shows the alignment of two proteins; 1bcfA and 1lki. Aligned residues are colored in red and unaligned are colored in blue. The Z-score for the pair (1bcfA, 1lki) is 7.3, meaning 7.3 standard deviations above background average.

The rest of this appendix describes how the structures are compared and how the similarity scores and Z-scores are calculated. The all-against-all Z-score between the representatives is part of the data used in this work.

The structure comparison is done using the DALI algorithm [14], which finds optimal pairwise 3D alignment of protein structures using an elastic similarity score. In the DALI algorithm, the 3D structure of a protein chain is represented using a distance matrix between all its C_α atoms which represent the distance between the residues. This representation, which is called a distance matrix or map, has the advantage of being invariant under rotation and translation and has more than enough information to reconstruct the three dimensional position of the C_α atoms, besides overall chirality (a mirror image of the protein will have the same distance matrix). Therefore, two similar structures will have similar distance maps regardless of the orientation and position of their reference

frames.

The DALI algorithm is an optimization algorithm which tries to find an optimal alignment between two chains. An alignment is a list of L pairs of indices of C_α atoms, $\{(x^A, x^B)_i\}_{i=1}^{L}$. Each pair, (x^A, x^B), represents that the x_A C_α atom in chain A is aligned with the x_B C_α atom in chain B. The optimization procedure tries to maximize a similarity score, S-score, between the distance maps of the aligned regions. The similarity score measures the pointwise resemblance of the distance maps. The exact calculation is as follows: Take two chains A and B with n^A and n^B C_α atoms respectively and an alignment of $L < n_A, n_B$ C_α atoms between them. The similarity score for chain A and B using this alignment is:

$$S_{AB} = \sum_{i}^{L} \sum_{j}^{L} \left(0.2 - \frac{\left| d_{ij}^{A} - d_{ij}^{B} \right|}{d_{ij}^{*}} \right) e^{-(d_{ij}^{*}/20\text{Å})^2} \tag{A.1}$$

The summation is done on all aligned pairs, $i, j = 1, \ldots, L$. Where d_{ij}^{*} is the mean of the residue distances d_{ij}^{A} and d_{ij}^{B} in chains A and B, each of which is the distance between the i-th and j-th aligned C_α atom in the corresponding chain. The distances are measured in Angstrom units.

This similarity score is an elastic score because it uses a relative, $\left| d_{ij}^{A} - d_{ij}^{B} \right| / d_{ij}^{*}$, rather than absolute deviations. This means that large deviation for distant pairs contribute the same as small deviations for close pairs. 0.2 is a relative deviation threshold. The term threshold is used because if an aligned pair has an overall negative contribution to the S-score, it will be profitable to remove it from the optimal alignment. Therefore, some of the terms within the optimal alignment can be negative, i.e. their relative deviation is larger than 0.2, only if they are compensated by the other positive terms of the aligned residue. The exponential factor down-weights the contribution of distant pairs. This factor is needed to reduce the contribution of distant pairs which are abundant but less discriminative between folds. The scale of 20Å is a typical domain size.

Notice that the similarity score is an overall sum of a pointwise symmetric calculation done between two L-by-L distance maps created by extracting the rows and columns of the

aligned residues form the distance maps of each of the chains. This operation resembles a dot product between the two matrices. In section 4.4, this resemblance is used to define a new normalization of the S-score.

The S-score of the optimal alignment has several important properties. First, it is always positive because, otherwise, the best alignment would be the empty alignment for which the S-score is 0. The optimal alignment between a chain to itself is, most probably, an alignment between all residues to themselves because all the terms in the sum contribute a large positive number. The S-score of this self-alignment is

$$S_{\mathrm{AA}} = 0.2 \sum_{i}^{n^{\mathrm{A}}} \sum_{j}^{n^{\mathrm{A}}} e^{-(d_{ij}^{\mathrm{A}}/20\text{Å})^2} \ .$$

This score is bound by a linear function of the length of the chain because there is a maximal physically possible value for the inner sum, K, which is obtained for an infinite maximally compact chain. Therefore, $\sum_{j}^{n^{\mathrm{A}}} e^{-(d_{ij}^{\mathrm{A}}/20\text{Å})^2} < K$ for all i, and $S_{\mathrm{AA}} < K n^{\mathrm{A}}$.

From the above treatment, it is clear that the S-score value depends on the length of the compared chains and thus can not be used directly to compare the pairwise similarity of chain A and B to the pairwise similarity of chain C and D. In order to overcome this, a normalization or calibration step is needed and is discussed later.

The S-score has another disadvantage. It is possible, though not very likely, that the S-score between chain A and B will be higher than the S-score between A and itself. This property contradicts the notion of S being a similarity score. For any similarity score, one would expect that $S_{\mathrm{AA}} > S_{\mathrm{AB}}$. This situation can arise because of the multiplication of the distortion factor by the down-weighting factor. For example, consider two structures, A and B, where B is a similar but slightly shrinked version of A. In both alignments, AA and AB, the best score is obtained by aligning each residue to itself or to its shrinked counterpart. The S-score of the AA alignment is not penalized for distortion but uses larger distances in the exponent. On the other hand, the AB alignment score is penalized for the distortion but uses smaller distances in the exponents. In cases where the contribution of the exponents is greater than the distortion penalty, one gets that $S_{\mathrm{AB}} > S_{\mathrm{AA}}$.

As discussed above, the S-score value depends on the number of aligned residues which depend on the length of the chains being compared. In order to put all scores on a common scale and to assign them a statistical meaning they were calibrated against pairwise all-on-all comparisons in the database, as a function of chain size. The calibrated score is expressed in terms of normalized Z-scores, that is the number of standard deviations above the mean.

$$Z_{\mathrm{AB}} = \frac{S_{\mathrm{AB}} - \overline{S}_{\mathrm{AB}}}{\sigma_{\mathrm{AB}}}$$

Note that the exact population on which the average and standard deviation are calculated is not clear from the articles and not from a discussion with Liisa Holm [33]. The Z-score, as the S-score, is a similarity measure, namely, a large Z-score between two structures means that they are very similar.

The Z-scores between all the representatives are used to create a dendrogram by applying an average linkage clustering algorithm (the algorithm is explained in details in 4.5.5). In the FSSP the dendrogram is called the "fold tree". Basically, the clustering algorithm is a bottom-up iterative algorithm. It starts with all representatives in different clusters and in each iteration it joins the two clusters with highest average Z-score between their components. The algorithm stops when all the chains are grouped into one cluster.

The dendrogram is a tree that represents the clusters unifications made during the algorithm. Cutting the dendrogram at different values of average Z-score creates a hierarchy of sets that have structural resemblance with different statistical significance.

In the FSSP database the dendrogram is cut in at six different levels $Z = 2, 3, 4, 5, 10, 15$. Cutting the 1188 representatives fold tree at $Z = 2$, in the December 1997 version of FSSP, yielded 364 fold classes. In the fold tree, each of the representatives is assigned six indices representing the sub-family number for each cut. In addition, each representative has an associated file which includes the alignments to all other representatives for which $Z \geq 2$ and the exact value of Z. Therefore, taking the Z-scores from all the files reproduces all pairwise comparison scores with $Z \geq 2$.

The DALI algorithm is a heuristic optimization algorithm that finds the alignment with maximal S-score [14]. The algorithm allows gaps with any length and also supports reversal of chain direction, but the last option is not used in the creation of the FSSP database. The algorithm consists of three stages: In the first stage, the distance matrices of the two chains are divided into overlapping sub-matrices of hexapeptide-hexapeptide contact patterns and pairs of similar contact patterns are found. Each contact pattern pair represents a subalignment of two fragments on each chain. The next stage is a Monte Carlo Metropolis algorithm that connects subalingments together. The search is done in parallel for several trajectories. In the last stage, the best alignments that were found are locally refined.

Appendix B

The CATH Database

B.1 General description

This section describes the CATH database. The database is maintained by Orengo *et al.* and is publicly available at the CATH web site [8] (see Appendix C for list of web sites). This section is based on articles [26][32]. The CATH database classifies domain structures into a hierarchy of families and sub-families. The four main levels of classification of a protein are: class (C), architecture (A), topology (T), and homologous superfamily (H). The top level, class, describes the secondary structure composition of the domain, e.g. mainly-α. The next level, the architecture level, describes the shape revealed by the orientation of the secondary structure units, e.g. barrel. The topology level uses sequential connectivity to distinguish members with similar shape but *different order* of secondary structure units. The homologous superfamily level groups domains which have high structural similarity and similar biological function. These domains are considered to be evolutionarily related. The homologous superfamily level is further divided into levels determined by sequence similarity alone.

B.2 Creating the database

The database is created using a semi-automatic procedure. Low level sequence and structure similarities are calculated automatically. The top class level and the domain separation are performed automatically, leaving the architecture level for manual inspection.

The database is, generally, created in a bottom-up manner, besides the top level which is performed at an earlier stage in order to save computation. Following is a list of the creation steps:

1. *Selection of structures for CATH* - CATH contains only well-resolved crystal structures (below 3.0Å resolution) and NMR structures from the PDB.

2. *Sequence comparison* - Pairwise sequence comparison between all selected sequences is performed using a standard Needleman and Wunsch algorithm [23]. Each pair is characterized by two ratios; *sequence identity* and *sequence overlap*. The sequence identity is the number of identical residues as a percent of the smaller protein, and sequence overlap is the percent of the larger protein which is equivalent to the smaller.

   Proteins are grouped, using a single linkage clustering algorithm (see 4.5.5), to form hierarchical sequence based families;

I family Identical proteins (100% sequence similarity and 100% overlap).

N family Near-identical proteins (more than 95% sequence similarity, and at least 85% overlap).

S family Groups proteins having 35% or more sequence identity and above 60% of the larger protein is equivalent to the smaller one.

   The 35% sequence identity ensures that all proteins within an S group have similar structure. Higher level similarities are revealed by structure comparison.

3. *Assignment of domain boundaries for multidomain proteins* - Up to this stage, proteins may be either single or multi-domained. At this point, multi-domain protein are divided into domains. One representative from each N group is analyzed to identify domains using three automatic methods. Domains are generally identified

as compact sub-strucurtures within a protein. Further details on domain identification are beyond the scope of this work and can be found in references 22-24 of [26]. If the identified domains are highly overlapping, they are used to chop the structure. Otherwise, the boundaries are manually assigned using visual inspection and by comparing to other databases, such as SCOP [22] and 3DEE [30].

Next, the domain boundaries are inherited by the other members of the N-family using the sequence alignment between them. A validation procedure alerts whenever a multi-domained protein is not properly covered by the domains defined on it. Finally, the classification into sequence similarity families (I,N,S families) is repeated (step 2) using the sequences of single domains.

4. *Automatic assignment of class (C)* - The class level is the highest level of the CATH classification and represents the secondary structures content of the domain (α helices and β sheets). There are four different classes: mainly-α, mainly-β, α-β, and few secondary structures (FSS) which groups all domains with very low secondary structure content. The class assignment is performed at this stage in order to save structural comparisons in the next stage. Structural comparisons are computationally intensive and are only calculated between structures within the same C-level family. Therefore, all inter-class comparisons are saved.

   The class is assigned using an automatic procedure that is applied to each S-level representative. The procedure (reference 30 of [26]) examines the secondary structure composition and identifies if the structure is mainly α or mainly β or α-β. 90% of the domain structures can be confidently assigned to the correct class. The remaining structures are treated manually.

5. *Structure comparison (H and T)* - Structural comparisons are performed between N-level representatives within each class. The comparison is performed using a structural alignment algorithm (SSAP) [32]. The SSAP algorithm uses dynamic

programming techniques, as done for sequence alignment, but instead of using an amino-acid substitution score it uses a score which measures resemblance of the structural environment of the residue. The algorithm is not discussed in this work. First, a fast version of the SSAP algorithm [24] finds pairs scoring above 75 (on a scale from 0 to 100). Next, a more sensitive version of the algorithm [32] re-aligns pairs with low initial score.

The SSAP scores and an overlap threshold are used by a single linkage algorithm (as in step 2) to cluster the domains into structural families. Two cutoffs on the SSAP score are applied, above 80 SSAP score and more than 60% overlap define the H-level and above 70 SSAP score with more than 60% overlap define the T-level. In addition to these criteria, proteins are assigned to a specific homologous superfamily (H-level) if their biological function is similar to the other members of the family. The function is determined using other databases, e.g. SWISSPROT [2], or the literature. When in doubt, the protein is assigned a separate H-level family.

6. *Assigning architecture* - The last stage in the classification process is assigning the architecture (A-level family). As described above, the architecture represents the shape created by the arrangement of the secondary structures in the domain, e.g. barrel, sandwich, roll. This classification is done manually using the commonly used classification of Richardson [28] and other well-known shapes from the literature. In case there is no specific shape, as in the mainly-α class, the architectures are divided into bundle and non-bundle. Complex arrangements, which can not be easily described, are placed in a general complex architecture.

7. *Assigning CATH numbers* - Each domain in the CATH database is assigned a set of numbers representing its family association at each level. For example, the single-domained protein Amylase (PDB entry - 1smd) is assigned the numbers 3.20.20.70 which represent that it is an α-β class (3), Barrel architecture (20), TIM Barrel

topology (20), and a member of the Narbonin (70) homologous superfamily. The numbers associated with the families (besides the class level) are multiplications of 10 to allow future insertions.

Appendix C

Used web sites

Following is a lists of all the web sites used during the work:

- FSSP database - http://www2.ebi.ac.uk/dali/fssp/
- CATH database - http://www.biochem.ucl.ac.uk/bsm/cath/
- 3Dee database - http://circinus.ebi.ac.uk:8080/3Dee/
- PDB - http://pdb.pdb.bnl.gov/

Bibliography

[1] Abola, E.E., Bernstein, F.C., Bryant, S.H., Koetzle, T.F., and Weng, J. 1987. "Protein Data Bank. *In Crystallographic Databases - Information Content, Software Systems, Scientific Applications* (Allen, F.H., Bergerhoff, G. and Sievers, R., eds), 107–132.

[2] Bairoch, A., and Boeckmann, B. 1992. "The SWISS-PROT protein sequence data bank", *Nucleic Acid Research* **20**, 2019–2022.

[3] Bernstein, F.C., *et. al.*, and Tasumi, M. 1977. "The protein data bank: a computer based archival file for macromoleculecular structures", *Journal of Molecular Biology* **112**, 535–542.

[4] Blatt, M., Wiseman, S., and Domany, E. 1996a. "Super–paramagnetic clustering of data", *Physical Review Letters* **76**, 3251–3255.

[5] Blatt, M., Wiseman, S., and Domany, E. 1996b. "Clustering data through an analogy to the Potts model", *Advances in Neural Information Processing Systems* **8**, 416–422 (1996), Touretzky, Mozer, Hasselmo, eds., MIT Press.

[6] Blatt, M. 1997. "Clustering of Data, Computational Capabilities and Applications of Neural Networks", Ph.D. Thesis.

[7] Branden, C., and Tooze, J. 1991. *Introduction to protein structure*. Gerald Publishing, New York and London.

[8] CATH web site, **http://www.biochem.ucl.ac.uk/bsm/cath/**.

[9] Duda, R.O., and Hart, P.E. 1973. *Pattern Classification and Scene Analysis*. Wiley–Interscience, New York.

[10] FSSP web site, **http://www2.ebi.ac.uk/dali/fssp/**.

[11] Fukunaga, K. 1990. *Introduction to statistical Pattern Recognition*. Academic Press, San Diego.

[12] Golan, Y., Linial, N., Tishby, N., and Linial, M. 1997. "A map of the protein space - An automatic hierarchical classification of all protein sequences", Preprint.

[13] Griffiths, R.B., 1967. "Correlations in Ising ferromagnets, I*", *Journal of Mathematical Physics* **8**, 473–483.

[14] Holm, L., and Sander, C. 1993. "Protein Structure Comparison by Alignment of Distance Matrices", *Journal of Molecular Biology* **233**, 123–138.

[15] Holm, L., and Sander, C. 1994. "The FSSP database of structurally aligned protein fold families", *Nucleic Acids Research* **22**, 3600–3609.

[16] Holm, L., and Sander, C. 1996. "The FSSP database: fold classification based on structure-structure alignment of proteins", *Nucleic Acids Research* **24**, 206–209.

[17] Holm, L., and Sander, C. 1996. "Mapping the Protein Universe", *Science* **273**, 595–602.

[18] Holm, L., and Sander, C. 1997. "Decision support system for the evolutionary classification of protein structures", To be published.

[19] Jain, A.K., and Dubes, R.C. 1988. *Algorithms for Clustering Data*. Prentice–Hall, Englewood Cliffs.

[20] Jaynes, E.T. 1989 in *Papers on Probability, Statistics and Statistical Physics*, edited by R.D. Rosenkrantz, Kluwer, Dordrecht, The Netherlands.

[21] Johnson, S.C. 1967. "Hierarchical clustering schemes", *Psychometrika* **32** 241.

[22] Murzin, A.G., Brenner, S.E., Hubbard, T., and Chotia, C. 1995. "SCOP: A Structural Classification of Proteins Database for the Investigation of Sequences and Structures", *Journal of Molecular Biology* **247**, 536–540.

[23] Needleman, S.B., and Wunsch, C.D. 1970. "A general method applicable to the search for similarities in the amino acid sequence of twp proteins", *Journal of Molecular Biology* **48**, 443.

[24] Orengo, C.A., Brown, N.P., and Taylor, W.R. 1992. "Fast Structure Alignment for Protein Databank Searching", *Proteins: Structure, Function, and Genetics* **14**, 139–167.

[25] Orengo, C.A., Flores, T.P., Taylor, W.R., and Thornton, J.M. 1993. "Identification and classification of protein fold famalies", *Protein Engineering* **6**, 485–500.

[26] Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., and Thornton, J.M. 1997. "CATH - a hierarchic classification of protein domain structures" *Structure* **5** 1093–1108.

[27] PDB web site, **http://www.pdb.bnl.gov/**.

[28] Richardson, J.S. 1981. "The anatomy and taxonomy of protein structure", *Advanced Protein Chemistry* **34**, 167–339.

[29] Sander, C., and Schneider, R. 1991. "Database of Homology-Driven Protein Structures and the Structural Meaning of Sequence Alignment", *Proteins: Structure, Function, and Genetics* **9**, 56–68.

[30] Siddiqui, A.S., and Barton, G.J. 1995. "Continuous and discontinuous domains: an algorithm for the automatic generation of reliable protein domain definitions", *Protein Sceince* **4**, 872–884.

[31] Stone, M. 1974. "Cross validatory choice and the assessment of statistical prediction", *Journal of the Royal Statistical Society* Series B **36**, 111–113.

[32] Taylor, W.R., and Orengo, C.A. 1989. "Protein Structure Alignment", *Journal of Molecular Biology* **208**, 1–22.

[33] Vendruscolo, M. private discussion with Liisa Holm, 1998.

[34] Wiseman, S., Blatt, M., and Domany, E. 1996. Preprint.

[35] Wu, F.Y. 1982. "The Potts model", *Reviews of modern physics*, **54**(1)235–268.

[36] 3Dee web site, **http://circinus.ebi.ac.uk:8080/3Dee/**.

אוניברסיטת תל-אביב
הפקולטה למדעים מדוייקים
ע"ש ריימונד ובברלי סאקלר
בית הספר לפיסיקה ואסטרונומיה

TEL AVIV UNIVERSITY
RAYMOND AND BEVERLY SACKLER
FACULTY OF EXACT SCIENCES
SCHOOL OF PHYSICS & ASTRONOMY

זיהוי וקיבוץ מבנים של חלבונים

חיבור זה מוגש כחלק מן הדרישות לקבלת תואר "מוסמך למדעים" M.Sc.

בית הספר לפיסיקה ואסטרונומיה

אוניברסיטת תל–אביב

על–ידי

גדי גץ

העבודה הוכנה בהדרכתם של

פרופסור איתן דומאני (מכון ויצמן למדע)

ופרופסור אמנון אהרוני (אוניברסיטת תל-אביב)

ספטמבר 1998