Master Thesis

Exploring the role of amino-acid alphabets in protein-interface classification

Author(s):
Somody, Joseph C.

Publication Date:
2015

Permanent Link:
https://doi.org/10.3929/ethz-a-010415857

Rights / License:
In Copyright - Non-Commercial Use Permitted
Master’s Thesis

Exploring the Role of Amino-Acid Alphabets in Protein-Interface Classification

Supervisors:

PROF. DR. AMEDEO CAFLISCH
[Internal]

DR. GUIDO CAPITANI
[Lead, External]

DR. JOSÉ M. DUARTE
[External]

Author:

JOSEPH C. SOMODY

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Computational Biology & Bioinformatics Department of Computer Science ETH Zürich with research at Protein Crystallography and Structural Bioinformatics Laboratory of Biomolecular Research Paul Scherrer Institute

http://dx.doi.org/10.3929/ethz-a-010415857

1 April 2015
Declaration of Originality

I, JOSPEH C. SOMODY, hereby confirm that I am the sole author of the written work here enclosed, entitled “EXPLORING THE ROLE OF AMINO-ACID ALPHABETS IN PROTEIN-INTERFACE CLASSIFICATION”, and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisors, Prof. Dr. AMEDEO CAFLISCH, Dr. GUIDO CAPITANI, and Dr. JOSÉ M. DUARTE.

With my signature, I confirm that:

- I have committed none of the forms of plagiarism described on the information sheet entitled Citation Etiquette, which are as follows:
  - Using the exact words or ideas from another author’s intellectual property (text, ideas, structure, etc.) without clear citation of the source.
  - Using text from the Internet without citing the uniform resource locator (URL) and the date it was accessed.
  - Reusing one’s own written texts, from different course papers or performance assessments, or parts thereof, without explicitly identifying them as such.
  - Translating and using a foreign-language text without citing its source.
  - Submitting work that has been written by another under one’s own name (“ghostwriting”).
  - Using an extract of another author’s work, paraphrasing it, and citing the source elsewhere than in the context of that extract.
- I have documented all methods, data, and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that this work may be screened electronically for plagiarism.

Place and Date: Zürich, Switzerland, 1 April 2015

Signature:  

iii
Abstract

Exploring the Role of Amino-Acid Alphabets in Protein-Interface Classification

JOSEPH C. SOMODY

It is no trivial task to determine whether a protein–protein interface in a crystal structure is biologically relevant. A large alignment of homologous sequences can be constructed, and the evidence provided by evolution can be extracted. Residues involved in biological interfaces tend to demonstrate less variability than other residues on the surface of the protein, but how the variability is calculated has an interesting impact on how well the interfaces can be classified. This thesis begins by optimising the parameters for the algorithm described above, then continues on to an investigation of reduced alphabets for the amino acids. Results show that grouping the amino acids into a mere two families plays a beneficial role in discriminating between biologically-relevant interfaces and the artefacts of crystal packing; however, grouping similar amino acids together does not necessarily have a positive effect: placing aspartic acid and glutamic acid in the same group of the reduced amino-acid alphabet, for example, has a decidedly-harmful impact on the classification accuracy. Two tangential appendices are attached to the thesis to document research forays that, although unfruitful, have the potential to succeed given further investment.
# Table of Contents

Declaration of Originality iii

Abstract v

Table of Contents vii

Abbreviations xi

## 1 Introduction

1.1 Basic Molecular Biology and Biochemistry  . . . . . . . . . . . . . . . . . . 1

1.2 X-Ray Crystallography . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  1.2.1 Protein Crystallisation . . . . . . . . . . . . . . . . . . . . . . . . . 3
  1.2.2 Data Collection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  1.2.3 Structural Modelling, Model Building, and Model Refinement . . . 4

1.3 Interface-Classification Problem . . . . . . . . . . . . . . . . . . . . . . . 5
  1.3.1 Crystallographic Symmetry . . . . . . . . . . . . . . . . . . . . . . . 5
  1.3.2 The Problem . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  1.3.3 Early Approaches . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
  1.3.4 CRK . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

1.4 EPPIC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
  1.4.1 The Fall of CRK . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
  1.4.2 The Rise of EPPIC . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

1.5 Amino-Acid Alphabets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
  1.5.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
  1.5.2 Relevance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
Table of Contents

2 Parameter Optimisation 15
  2.1 Parameters in EPPIC .......................... 15
  2.2 Optimisation .................................. 16
    2.2.1 BioMany and XtalMany .................... 16
    2.2.2 Computation ................................ 17
  2.3 Results ...................................... 18
    2.3.1 Treatment of $nopred$ Calls .............. 18
    2.3.2 Prefiltering Results ....................... 18
    2.3.3 Measures of Classifier Performance ........ 18
    2.3.4 Optimised Parameters ...................... 20
  2.4 Conclusion ................................... 24

3 Reduced Amino-Acid Alphabets 27
  3.1 Rationale .................................... 27
  3.2 Published Alphabets .......................... 27
    3.2.1 Methods .................................. 27
    3.2.2 Results .................................. 29
    3.2.3 Receiver Operating Characteristic .......... 30
    3.2.4 Further Results ............................ 30
  3.3 Random 2-Alphabets .......................... 31
    3.3.1 Methods .................................. 32
    3.3.2 Results .................................. 32
  3.4 Transitioning 2-Alphabets .................... 34
    3.4.1 Methods .................................. 35
    3.4.2 Results .................................. 36
    3.4.3 Linear Model ............................... 38
  3.5 Conclusion ................................... 39

4 Surface-Residue Sampling 41
  4.1 Rationale .................................... 41
  4.2 Strategies for Surface-Residue Sampling .... 42
    4.2.1 Classic Strategy .......................... 42
    4.2.2 Novel Strategy ............................. 43
  4.3 Methods ...................................... 44
  4.4 Results ...................................... 44
  4.5 Conclusion ................................... 44

5 2-Alphabets 47
  5.1 Rationale .................................... 47
  5.2 Methods ...................................... 47
    5.2.1 Stirling Numbers of the Second Kind ...... 48
  5.3 Results ...................................... 50
  5.4 Linear Model .................................. 50
  5.5 Conclusion ................................... 53
6 Conclusion  
6.1 Summary ........................................... 55

A Hydrogen-Bond Detection  
A.1 Hydrogen Bonding .................................. 57
A.2 Rationale .............................................. 58
A.3 Definition ............................................ 58
A.4 Implementation ...................................... 58
A.5 Conclusion ............................................ 59

B Core-Residue Decay  
B.1 Theory ................................................. 61
B.2 Results ................................................ 62
B.3 Conclusion ............................................ 64

List of Figures  

List of Tables  

Acknowledgements  

References
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH</td>
<td>Eidgenössische Technische Hochschule</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>DNA</td>
<td>Deoxyribonucleic Acid</td>
</tr>
<tr>
<td>RNA</td>
<td>Ribonucleic Acid</td>
</tr>
<tr>
<td>A</td>
<td>Adenine</td>
</tr>
<tr>
<td>C</td>
<td>Cytosine</td>
</tr>
<tr>
<td>G</td>
<td>Guanine</td>
</tr>
<tr>
<td>T</td>
<td>Thymine</td>
</tr>
<tr>
<td>U</td>
<td>Uracil</td>
</tr>
<tr>
<td>PPI</td>
<td>Protein-Protein Interaction</td>
</tr>
<tr>
<td>PDB</td>
<td>Protein Data Bank</td>
</tr>
<tr>
<td>AU</td>
<td>Asymmetric Unit</td>
</tr>
<tr>
<td>PISA</td>
<td>Proteins, Interfaces, Structures and Assemblies</td>
</tr>
<tr>
<td>CRK</td>
<td>Core-Rim $K_a/K_s$ Ratio</td>
</tr>
<tr>
<td>MSA</td>
<td>Multiple Sequence Alignment</td>
</tr>
<tr>
<td>ASA</td>
<td>Accessible Surface Area</td>
</tr>
<tr>
<td>BSA</td>
<td>Buried Surface Area</td>
</tr>
<tr>
<td>EPPIC</td>
<td>Evolutionary Protein-Protein Interface Classifier</td>
</tr>
<tr>
<td>DC</td>
<td>Duarte-Capitani</td>
</tr>
<tr>
<td>NMR</td>
<td>Nuclear Magnetic Resonance</td>
</tr>
<tr>
<td>HPCC</td>
<td>High-Performance Computing Cluster</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>MCC</td>
<td>Matthews Correlation Coefficient</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic</td>
</tr>
<tr>
<td>AUROC</td>
<td>Area Under the ROC Curve</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>JPA</td>
<td>Java Persistence API</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>PDB ID</td>
<td>PDB Identifier</td>
</tr>
<tr>
<td>LBR</td>
<td>Laboratory of Biomolecular Research</td>
</tr>
<tr>
<td>PSI</td>
<td>Paul Scherrer Institute</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This master’s thesis is mainly an investigation into using reduced amino-acid alphabets to calculate conservation in alignments of homologous amino-acid sequences. A direct consequence of using computational methods to solve biological questions is that the reader should be well-versed in both basic biology and basic computer science. In order to lay the foundation for all readers to begin with equal footing, this introductory chapter will briefly describe the required background knowledge.

1.1 Basic Molecular Biology and Biochemistry

As with any scientific report with a focus on molecular biology, this master’s thesis must necessarily start with an overview of the central dogma of molecular biology. Deoxyribonucleic acid (DNA) is the genetic material that forms the inheritable basis for life. DNA can be replicated, an essential part of cell division, or transcribed into ribonucleic acid (RNA), the first step of gene expression. DNA and RNA are both chains of nucleotides, each nucleotide being one of adenine (A), cytosine (C), guanine (G), and thymine (T)—or uracil (U) for RNA—attached to a sugar–phosphate backbone, as shown in Figure 1.1. Once RNA has been transcribed from DNA, nascent RNA is processed by the cellular machinery, thereby regulating its transcription or splice pattern and forming mature RNA. The ribosome, itself composed of protein and RNA, is responsible for translating the RNA into the protein for which it codes, which then folds into an energetically-stable conformation as it comes off the ribosomal complex. A protein is a chain of amino acids, each being one of twenty possibilities that possess different properties. It is their sequence that determines the structure of the protein and, thus, the protein’s function [1].
Chapter 1 — Introduction

Chapter 1 — Introduction

Chapter 1 — Introduction

Figure 1.1: The composition of RNA and DNA. RNA is a single-stranded chain of nucleotides, each being one of adenine (A), cytosine (C), guanine (G), and uracil (U), attached to a sugar–phosphate backbone. DNA is of similar composition, but it is double-stranded and contains thymine (T) instead of uracil (U). This figure was adapted from graphics by Sponk and R. Mattern [2].

As stated by Hartwell et al., “a discrete biological function can only rarely be attributed to an individual molecule” [3]. Biological functions, which can be anything from signal transduction to cell-cycle regulation to enzymatic digestion, often require the formation of protein complexes [4, 5]. These protein–protein interactions (PPIs) can be between subunits of the same type, forming homooligomers, or of different types, forming heterooligomers. The interactions in homooligomers can be further subdivided into isologous and heterologous [6]. The subunits form an isologous association when they both contribute the same set of amino-acid residues to the interface, whereas they form a heterologous association when they contribute a different subset of amino-acid residues to the interface. Schematic examples of such oligomers can be seen in Figure 1.2. Furthermore, isologous interfaces have identical interacting surfaces, resulting in a necessarily dimeric and closed structure with a twofold axis of symmetry, while heterologous interfaces have non-identical interacting surfaces, which must be complementary but are not usually symmetric [7]. Heterologous interfaces can, but need not, give rise to infinite assemblies like the one shown in Figure 1.2(c). In the case that they do not form infinite assemblies, heterologous interfaces must display circular symmetry.
The amino-acid residues that point into the interface between the proteins are the interacting residues and are often required to maintain the PPI. Typically, there are also residues critical for the PPI that do not interact across the interface: internal support residues that stabilise the main-chain scaffold of the interface [8].

1.2 X-Ray Crystallography

1.2.1 Protein Crystallisation

The first step in protein X-ray crystallography is to crystallise the protein of interest, beginning with its purification. The protein is typically obtained by recombinant methods: the gene for the protein is cloned into an expression vector and transferred into a production organism, such as *Escherichia coli*, in order to express the protein in large quantities. The protein is then purified out of the culture, generally by means of affinity chromatography targeting a protein tag whose coding sequence had been inserted at one end of the gene of interest.

Once the protein has been purified, it must be crystallised. To crystallise, protein molecules must fall out from solution and assemble themselves into a crystal lattice, typically done by adding a precipitant to a concentrated solution of protein [9]. The formation of protein crystals is a finicky procedure with many variables, and, as it is not determinable in advance which solutions and conditions will yield protein crystals, let alone well-diffracting ones, many trials with varying characteristics must be carried out. In a popular method known as hanging-drop vapour diffusion, the reservoirs of microtitre plates are filled with solutions with various acidities, temperatures, ionic strengths,
buffers, precipitants, etc. A small amount of the protein solution is mixed with an equal amount of the reservoir solution and transferred on to a glass slide, which is then placed atop the reservoir, thereby sealing it, with the drop facing the interior of the reservoir [9]. Since the drop has a lower precipitant concentration than the reservoir solution, water evaporates from the drop, leading to a slow increase in protein concentration. Over time, the drop becomes supersaturated with protein, and a nucleation event may lead to the formation of protein crystals.

This screening process suffers from the curse of dimensionality: crystallisation is affected by many variables, which makes sampling the multidimensional solution space an onerous task, especially since the amount of protein available is generally limited. It is for this reason that sparse-matrix sampling was proposed by Jancarik and Kim in 1991 [10], which involves strategically screening crystallisation conditions that have been successful in the past and using non-repeating combinations of reagents [9]. This method has become a very popular technique in protein crystallisation and is now widely applied in commercial crystallisation kits [11].

1.2.2 Data Collection

After obtaining a satisfactory crystal of the protein of interest, the crystal is most commonly flash-frozen in liquid nitrogen and mounted to a goniometer head. The crystal, kept at 100 K by a stream of gaseous nitrogen, is then exposed to an incident beam of monochromatic X-ray radiation while slowly being rotated about a spindle axis. The X-rays are diffracted, and diffraction patterns are captured by a detector sensitive to X-ray photons [9].

1.2.3 Structural Modelling, Model Building, and Model Refinement

There is an inherent difficulty in X-ray crystallography: the phase problem. The information regarding the phase of any given spot, known as a reflection, on the diffraction pattern is lost when captured as an image. This is problematic, because the phases of the reflections are more crucial to structural modelling than their intensities [12]. One of the most common solutions to this problem is molecular replacement, where the phases from a similar, solved structure are grafted on to the intensities of the crystal of interest after computationally finding the correct orientation and position in the target asymmetric unit (defined in Subsection 1.3.1) [13]. In Figure 1.3, a schematic example (with artistic renderings of diffraction patterns, not the real ones) of molecular replacement as a solution to the phase problem is illustrated: the cat in Figure 1.3(a) represents the crystal structure to be solved, whose diffraction pattern is shown in Figure 1.3(b), but only
the phaseless (shown as colourless in this analogy) diffraction pattern in Figure 1.3(c) is available to crystallographers. The phaseless intensities from the reflections in Figure 1.3(c) are used, but the phases from a similar structure (not shown), appropriately positioned in the asymmetric unit, are applied on to the intensities. The combination, shown in Figure 1.3(d), yields the structure in Figure 1.3(e), an approximate solution.

The recorded series of two-dimensional diffraction patterns, with intensities and grafted phases, must then be converted into a three-dimensional model of the electron density. The Fourier transform is an operation that decomposes a periodic function into the frequencies that constitute it. The diffraction pattern of a crystal is composed of peaks as a result of the crystal’s periodicity, so the inverse of the Fourier transform can be used to recompose the electron density from the observed pattern [15]. A biochemical model of the protein can then be built into the electron density and iteratively refined until the calculated diffraction data from the model agree as much as possible with the experimental diffraction data. The final structure is then formatted appropriately and deposited into a database, such as the Protein Data Bank (PDB) [16].

1.3 Interface-Classification Problem

1.3.1 Crystallographic Symmetry

When a protein structure is determined by X-ray crystallography, its space group must be identified. The space group describes the symmetry of the crystal: it provides sufficient information to mathematically generate a unit cell from the asymmetric unit (AU) [17]. More specifically, the AU is the smallest unit of the crystal that contains sufficient information to produce the unit cell through the symmetry operations of the crystal space group, while the unit cell is the smallest unit of the crystal that contains sufficient information to produce the (infinite) crystal lattice by translation only [18]. A schematic example of crystallographic symmetry is shown in Figure 1.4, where the AU is any one duck, of either colour, and the unit cell is shown in green overlay.

1.3.2 The Problem

The crystal structure of a protein under investigation contains non-biological interactions, a direct consequence of the nature of crystals. (An exception to this is seen in the case of natural crystals like polyhedrins in insect viruses [19, 20] or like the Cry3A toxin found naturally crystallised in *Bacillus thuringiensis* [21], where all interfaces are biologically relevant since the protein’s crystal form evolved to perform a particular biological
Figure 1.3: An illustration of molecular replacement as an approach to solving the phase problem with artistic renderings of diffraction patterns, not the real Fourier transforms. The cat in Figure 1.3(a), having the diffraction pattern in Figure 1.3(b), where the colours represent phases, needs to be generated from the phaseless (colourless) diffraction pattern in Figure 1.3(c). Phases from a similar structure are appropriately aligned and grafted on to Figure 1.3(c), resulting in Figure 1.3(d). Figure 1.3(d) yields the approximate solution shown in Figure 1.3(e). This figure was inspired by a similar one by Kevin Cowtan [14].
The matter that constitutes each AU in a crystal interacts not only within the AU but also between adjacent AUs. The AU is defined without reference to the biological context of the complex, meaning the complexes in each AU are not necessarily the true biological complexes [22]. A schematic example is shown in Figure 1.5, where the AU is clearly any one subunit (of any colour), but the definition of the biological unit is debatable without any complementary information.

The crystal structure of a protein homooligomeric in solution is not restricted to symmetry operations based on the biological oligomeric interactions between amino-acid chains. For example, a protomer that forms a homodimer \textit{in vivo} does not necessarily only have that one isologous interaction in its crystalline form. As a result, when crystallographers investigate the crystal lattice, it is not usually obvious which contacts are due to biological interactions and which contacts are due simply to crystal packing. In the past, especially with simple structures, biological interactions have been discriminated from crystal contacts by mere manual inspection, an endeavour that has become more and more challenging as the complexity of the proteins being researched has risen over time [24]. This difficulty has become further exacerbated by a lack of classical biochemical background on new proteins: it is quite rare to see full biochemical characterisations for the majority of today’s crystallographic targets [24].
1.3.3 Early Approaches

Over the past twenty years, attempts to solve the above problem have varied widely in their approaches. In the 1970s, Janin started investigating protein–protein interactions [25, 26] and, in 1995, began researching crystal contacts [27]. Among the first was Janin’s method of discriminating between subunit contacts and crystal artefacts based solely on the area of the interface [28]. A few years later, Valdar and Thornton trained neural networks with both the area of the interface and the conservation scores of the amino-acid residues involved, which resulted in a highly-accurate predictor [29]. A couple more methods arose in the early 2000s, which focussed on analysing residue conservation in order to tell biological and non-biological interfaces apart: Elcock and McCammon...
determined that comparing the conservation of amino-acid residues involved in the interface with the conservation of residues elsewhere on the surface of the protein was an effective discriminator [30], and Guharoy and Chakrabarti found that subdividing the interface into a “core” region and a “rim” region and comparing the residue conservation between the two regions worked even better, since binding sites and other conserved patches on the rest of the protein surface led to anomalous results [31]. Researchers thereafter began using methods in machine learning in order to combine various parameters describing the interfaces of interest and extract statistical power from a trained classifier [32–34]. Of particular note is NOXclass, a support vector machine–based classifier that distinguishes between three classes of interactions by analysing six interface properties, including interface area and amino-acid composition [33]. These classes consist of obligate interactions, where protomers are not independently found as stable structures \textit{in vivo}; non-obligate interactions, where protomers are; and crystal-packing interactions [33].

In 2007, Krissinel and Henrick developed a thermodynamic method for estimating interface stability based on the binding energy of the interface and the entropy change due to complex formation [35]. This method is implemented in Proteins, Interfaces, Structures and Assemblies (PISA) [35], which has become the \textit{de facto} standard for interface classification in the community. In 2008, Xu \textit{et al.} performed an analysis of interface similarity in crystals of homologous proteins to examine whether having interfaces in common across different crystal forms can be used to calculate the likelihood that an interface is biological [36]. Three years later, Xu and Dunbrack released ProtCID [37], a database of homologous interfaces—both homodimeric and heterodimeric—observed in multiple crystal forms, grouped by Pfam domain [38] and crystal form.

1.3.4 CRK

The core–rim $K_a/K_s$ ratio (CRK) is a measure developed by Schärer \textit{et al.} in 2010 to distinguish biological interfaces from non-specific ones [39]. It is based on an idea originally proposed by Bahadur \textit{et al.} in a 2004 paper [40], which was to designate each residue involved in the interface as either a core residue or a rim residue, depending on the extent to which atoms in the residue are buried upon interface formation—core residues contain at least one fully-buried atom, while rim residues, which surround core residues, have only partially-buried atoms.

In 2001, Elcock and McCammon published a method based on comparing the information entropy at various positions, each designated as either “interface” or “non-interface”, in multiple sequence alignments (MSAs) [30]. It was found that the average sequence
entropy for interface residues is lower than that of non-interface residues. By restricting
the focus and looking only at interface residues, Guharoy and Chakrabarti refined this
method in 2005 and showed that the Bahadur et al. classifications—core residues and
rim residues—provided a better basis for comparisons of sequence entropy [31].

The CRK algorithm expands on these methods: given a PDB structure, the sequence
of each chain is used as a BLASTP [41, 42] query to search for homologous sequences
in the UniProt database [43], which are then filtered so as to have homology above a
threshold, appropriate taxonomy, and low pairwise redundancy. The protein sequences
for these homologues (as well as for the query) are then aligned with one another using
T-Coffee [44], the resulting protein MSA is backtranslated into an MSA of coding se-
quencies using RevTrans [45], and the selection pressure at each site is calculated with
Selecton [46, 47].

The accessible surface area (ASA) of an amino-acid residue is the size of the region
accessible to a solvent molecule, usually water [48]. This is generally calculated using
the Shrake–Rupley algorithm [49], where a probe sphere is “rolled” along the surface
of the region while checking for steric clashes that would imply a particular point on
the surface is not accessible. The buried surface area (BSA) is similar but measures
how accessible the residue is after interface formation [50]. The geometric descriptions
of the interfaces for the input PDB structure are retrieved from PISA, and the residues
are divided into a rim subset and a core subset, depending on what fraction of their
accessible surface area becomes inaccessible after interface formation. This fraction is
known as the burial, and the default burial cutoff for core residues in CRK is 95%.

The final steps in the CRK algorithm are calculating the ratio $\omega = K_a/K_s$ for each
residue, separating the $\omega$ values by class (either core or rim), and calculating the statistics
(both the unweighted mean and the mean weighted by the BSA) for both residue classes.
The ratio can then be used to determine whether the interface is biologically relevant:
a biological interface is expected to have more purifying selection for core residues than
rim residues, which would yield a ratio $\text{CRK} = \frac{\langle \omega \rangle_{\text{core}}}{\langle \omega \rangle_{\text{rim}}} < 1$. 
1.4 EPPIC

1.4.1 The Fall of CRK

CRK, as a proof-of-concept work, was successful but had its limitations. Firstly, the CRK pipeline was complicated: finding protein homologues, aligning them, backtranslating the alignment, calculating the selection pressure, classifying residues, calculating ratios, etc. This meant not only that any large-scale surveys of protein structures would be computationally infeasible but also that there were many welds in the pipeline where something could go wrong. In fact, mapping protein sequences back to their coding sequences turned out to be one such problem since there were often issues with cross-referencing between databases. A protein sequence would often map to multiple coding sequences or, even worse, to no coding sequences, which led to fewer sufficiently-significant predictions. In addition, the step in which CRK calculates selection pressures using Selecton was found to be exceedingly slow, further constraining the method’s usability.

1.4.2 The Rise of EPPIC

In 2012, the research group behind CRK published a new approach, the Evolutionary Protein–Protein Interface Classifier (EPPIC) [24]. Although grounded in the same concepts as CRK, EPPIC overcame CRK’s limitations. Two evolutionary predictors, core–rim and core–surface, were introduced, as well as a new geometric criterion for interface classification, which was based simply on the number of core residues in an interface. This allowed for a call to be made even when insufficient homologues were available for evolutionary analysis. Next, the use of $K_a/K_s$ ratios was abandoned in favour of sequence entropies. It was found that better performance was obtainable by using sequence entropies as long as homologues were stringently selected and had low redundancy. This change not only did away with the speed-limiting step involving Selecton but also meant that mapping protein sequences back to coding sequences was no longer necessary.

As mentioned in Subsection 1.3.4, in 2001, Elcock and McCammon pioneered a method comparing sequence entropy between interface and non-interface positions in MSAs [30]. EPPIC expands on this substantially: for the core–surface call, core residues of the interface are compared to surface residues outside the interface. More specifically, in a manner similar to that described by Valdar and Thornton [51], EPPIC randomly samples the surface residues to be compared from a pool of surface residues in order to maximise statistical robustness.
When searching for homologues, EPPIC uses a conservative sequence-identity cutoff of 60%, which is lowered to 50% in the event that insufficient homologues are found. These high cutoffs ensure that homologues have a high structural similarity at both tertiary and quaternary levels [24].

EPPIC uses the CRK definition for determining burial, as defined in the paper by Schärer et al. in 2010 [39]. For each of the three predictors contained in EPPIC, a different burial cutoff can be used. EPPIC determines the solvent accessibility of residues before and after the formation of the protein–protein interface, and the ratio is used to classify them as either “core” (residues fully buried upon interface formation), “rim” (residues partially buried upon interface formation), “surface” (surface residues not involved in an interface), or “other” (any non-interface, non-surface residues; for example, residues in a subunit’s core). The geometry-based predictor simply returns the number of core residues in the interface as its score, the core–rim evolutionary predictor returns the ratio of sequence entropies for core interface residues compared to rim interface residues, and, finally, the core–surface evolutionary predictor uses random sampling of surface residues and returns the “distance” of the average entropy for core residues to the mean of the samples of surface residues in terms of their standard deviation, a Z-score–like measure. This will be explained in greater detail in Chapter 4.

These three scores are converted to calls by comparison to a threshold unique to each predictor. Each call can be one of bio, meaning that the predictor believes the interface to be biologically relevant; xtal, meaning that the predictor believes it to be a crystal contact; or nopred, meaning that the predictor has insufficient data to go on. The three calls are combined to form a consensus call through a simple-majority voting scheme with some modifications. Since the geometric predictor always returns a call, if both evolutionary predictors give no prediction, the geometry call is taken as the consensus. Also, in certain cases, more weight is given to the calls by the evolutionary predictors: when there is exactly one each of bio, xtal, and nopred, the non-nopred evolutionary call is made the consensus call.

1.5 Amino-Acid Alphabets

1.5.1 Background

There are twenty different naturally-occurring amino acids, complex strings of which form proteins. Anfinsen’s dogma states that the native structure of a protein is determined solely by the protein sequence of amino acids [52]. It has been shown, however, that point mutations in a protein’s sequence are often tolerated [53] and that proteins
with far fewer types of amino acids can still form stable structures [54, 55]. In one case, thirty-eight residues of a fifty-seven–residue protein domain were each replaced with one of five possible residues, and the domain retained its fold [54]. This suggests that the standard twenty-letter amino-acid alphabet could, perhaps, be reduced.

Investigations into the degeneracy of amino-acid sequences and the possibility to condense the amino-acid alphabet were published as early as 1979 [56]. In 1994, Strelets et al. developed a model of combinatorial sequence space and broke down known protein sequences into overlapping subsequences of a fixed length, which led to an unexpectedly-high clusterisation that “looked as if certain protein sequence variations occurred and were fixed in the early course of evolution” [57], further propelling the idea that there were simplifications that could be made in attempts to model the complexity of protein sequence.

The simplest reduced protein alphabet is a two-letter one where amino acids are grouped as either hydrophobic or polar. This has been dubbed the HP model, but, as argued by Wolynes in 1997, two-letter alphabets result in too many local minima and insufficient funnelling in the energy landscape of protein folding [58]. In 1999, Wang and Wang took a computational approach to simplifying the protein-folding alphabet using minimised mismatch between properties of amino acids as a function of the size of the amino-acid alphabet and determined that, depending on how many properties are to be modelled, various plateaus exist to describe proteins at different coarse-grained levels [59].

In 2000, Murphy et al. set out to determine whether there is a minimal protein alphabet with which all proteins could still fold [60]. They used the BLOSUM50 similarity matrix [61], which was derived from gapless blocks in large alignments of protein sequences, to correlate the most-related amino acids, merge them, and construct a simplified similarity matrix. This process, applied iteratively, showed clear tendencies: residues with similar physicochemical properties first group together, the small amino acids then come together to form a group, and the alphabet ultimately reduces to the HP model. The authors tested the effects by applying the reduced alphabets on protein sequences and aligning them to the SCOP40 clustered database [62] to determine how much of the protein fold mapping is lost. An amino-acid alphabet of ten or twelve letters only reduced the percentage coverage retained by approximately 10%; however, a steep loss of fold recognition was observed with any simpler alphabet [60].

Three years later, in a very similar approach, Li et al. used the BLOSUM62 similarity matrix [61] to guide an iterative condensation of the protein alphabet and compare them for information loss just as Murphy et al. did. The authors noted that ten-letter alphabets allow for nearly the same ability to detect distantly related folds as with
twenty-letter alphabet, which may mean that ten amino acids would suffice to characterise the complexity in proteins [63]. Research into reduced protein alphabets has also continued in recent years with a greater focus on their applications in protein design [64].

1.5.2 Relevance

In EPPIC, as described in Subsection 1.4.2, an MSA of putative homologues is generated for the two evolutionary predictors. The sequence entropy is then calculated at each position, and each position is categorised as “core”, “rim”, “surface”, or “other”. The scores are based on the differences in sequence entropy between these classes. It is for these calculations that reduced amino-acid alphabets are needed. The signal-to-noise ratio of a column is a delicate balance in that variation should be detected but not taken into account if the variations do not matter at a biochemical level. Part of this could be resolved through cleverly choosing an appropriate amino-acid alphabet. As a result, a large part of this thesis is an investigation into the effects of various reduced amino-acid alphabets on EPPIC’s output.
Chapter 2

Parameter Optimisation

2.1 Parameters in EPPIC

The strategy behind EPPIC is fairly fixed: given a structure, for each possible interface, the program classifies residues based on solvent accessibility before and after the formation of the interface; creates a large, non-redundant MSA of putative homologues; and uses this information to return three different measures by which to classify the interface as either biologically relevant or a crystal contact. This does not, however, mean that EPPIC does not allow for any customisability. In fact, the gamut of adjustable parameters that EPPIC offers proves just the opposite.

EPPIC allows the user to choose the size of the amino-acid alphabet used for the sequence entropy calculations (from the ones proposed by Murphy et al. [60]), the BSA-to-ASA burial cutoff for core-residue assignment for each of the three classifiers, the calling thresholds for determining bio or xtal from the raw scores, the hard and soft sequence identity cutoffs, the maximum number of homologues to use, the number of sphere points for ASA calculations, as well as several other parameters.

Of the aforementioned parameters, this chapter of the thesis will concern the optimisation of seven of them: the three burial cutoffs, the three call thresholds, and the reduced amino-acid alphabet.
2.2 Optimisation

2.2.1 BioMany and XtalMany

In order to optimise parameters, a method of evaluating the accuracy of the classifier is required. Therein lies one of the most challenging obstacles: gold-standard datasets with experimental backing are key to properly training a classifier, but the manually-curated ones available to date have taken a long time to compile, been small (on the order of $10^2$ entries), and been prone to human error [65].

In 2012, when EPPIC was originally released, two small datasets were compiled by the authors. These Duarte–Capitani (DC) datasets, DCbio and DCxtal, were used to optimise the EPPIC parameters [24]. Since it had long been known that biological interfaces tend to be much larger than crystal contacts [28], Duarte et al. decided to focus their curation efforts on the region harder to classify. The DCbio dataset, therefore, consisted of small interfaces validated as biological, while the DCxtal dataset contained large crystal contacts.

For a true determination of the optimal parameters, the DC datasets, each having only eighty-one entries, were too small. To this end, Baskaran et al. generated the Many datasets, BioMany and XtalMany [65]. BioMany was based on the ProtCID database [36, 37], which infers that an interface is biological when it is present across multiple crystal forms. The ProtCID database also clusters interfaces by Pfam architecture [38], which allowed for a selection of interfaces of minimal redundancy to be pulled from the database. Using conservative thresholds, 2666 such interfaces were added to the BioMany dataset. Additionally, 171 further biological interfaces that had been validated by nuclear magnetic resonance (NMR), another method of determining protein structure, were added to the dataset. Any interfaces larger than $2000 \text{Å}^2$ were removed from BioMany in order to keep the dataset focused on the range of interfaces harder to classify.

In 1965, Monod et al. first discussed interfaces that lead to infinite assemblies [6]. As mentioned in Section 1.1, homomeric interfaces can be either isologous or heterologous, the latter of which has the potential to assemble infinitely. Baskaran et al. assert that, when considering a protein crystal, any interfaces produced by pure translation or a screw axis cannot result in closed assemblies, and, therefore, any such observed interfaces can be assumed to be crystal contacts [65]. The XtalMany dataset was populated using this assumption, the interfaces were clustered by sequence to decrease redundancy, and any interfaces smaller than $600 \text{Å}^2$ were removed. This resulted in a dataset of 2913 crystal contacts.
2.2.2 Computation

It was determined early on that the optimisation of parameters would have to involve a full search of the parameter space. Not all parameters are interdependent, which meant that there could be a certain degree of parallelisation in the optimisation. Additionally, the call thresholds only come into play during the conversion of scores to calls, allowing these parameters to be handled \textit{a posteriori}.

The burial cutoff for the geometric predictor was evaluated from 50\% to 100\% in increments of 5\%. The burial cutoffs for the core–rim and core–surface evolutionary predictors were each evaluated from 50\% to 100\% in increments of 10\% in combination with the reduced amino-acid alphabet being varied across the two-, four-, eight-, ten-, and fifteen-letter alphabets proposed by Murphy \textit{et al.} \cite{60} and the six-letter alphabet proposed by Mirny and Shakhnovich \cite{66}. The reduced amino-acid alphabets used for the optimisations are shown in Table 2.1.

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>Source</th>
<th>Amino-Acid Alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>\cite{60}</td>
<td>ACFGILMPSTWVY:DEHKNQR</td>
</tr>
<tr>
<td>4</td>
<td>\cite{60}</td>
<td>AGPST:CILNV:DEHKNQR:FWY</td>
</tr>
<tr>
<td>6</td>
<td>\cite{66}</td>
<td>ACILMV:DE:FHWY:GP:KR:NQST</td>
</tr>
<tr>
<td>8</td>
<td>\cite{60}</td>
<td>AG:CILMV:DENQ:FWY:H:KR:P:ST</td>
</tr>
<tr>
<td>10</td>
<td>\cite{60}</td>
<td>A:C:DENQ:FWY:G:H:ILMV:KR:P:ST</td>
</tr>
</tbody>
</table>

\textbf{Table 2.1:} The seven amino-acid alphabets originally released with EPPIC. For each alphabet, the alphabet identifier—simply the number of letters in the alphabet—is shown, as well as the authors who engineered it and the alphabet itself. The twenty-letter alphabet is an unreduced alphabet and, therefore, has no source.

Table 2.1 also introduces a standardised way to describe amino-acid alphabets. Each group of amino acids represented by one letter in the alphabet is written as a string of one-letter codes corresponding to the amino acids in that group. The one-letter codes are written in alphabetical order, and then the full strings are written out from left to right, also sorted alphabetically, using the colon character (:) as a separator.

For each set of parameters, EPPIC was run on 5607 PDB structures from the BioMany and XtalMany datasets, which led to a grand total of $5607 \times (11 + 6 \times 7 + 6 \times 7) = 532,665$ runs of the program. Clearly, with each run lasting approximately half a minute, this would have taken somewhere on the order of months to compute on a standard personal computer. As a result, computations were carried out using two high-performance computing clusters (HPCCs): Brutus at ETH Zürich and Merlin 4 at the Paul Scherrer Institute.
2.3 Results

2.3.1 Treatment of nopred Calls

As discussed in Subsection 1.4.2, there are three different calls that EPPIC’s predictors can return: bio, xtal, or nopred. As a classifier, EPPIC’s output should be analysed in the typical manner, which involves tallying correct and incorrect classifications for both the positive and negative datasets (BioMany and XtalMany, respectively). The handling of nopred calls in this binary scheme, however, is not trivial. This is because, no matter how they are treated, some bias will necessarily appear in the results. For example, treating them with the same “contempt” as incorrect classifications discourages their occurrence, which is contrary to the fundamental idea of having nopred calls. Ideally, nopred calls should be preferred to misclassification. On the other hand, treating them favourably encourages EPPIC to return fewer bio and xtal calls, which, clearly, is flawed behaviour for software whose purpose is to make predictions.

It was decided that the nopred calls could be treated as xtal calls. This converts the hard-to-handle tertiary classification problem into a binary one. A possible downside to this approach is that nopred calls are now considered as suitable to be returned for crystal interfaces. This outcome is easily defended: a typical user of EPPIC is searching for biological interfaces in a sea of crystal contacts and would likely view any non-bio call similarly. As of February 2015, EPPIC’s final calls across the entire PDB show that approximately six times more crystal interfaces exist than biological interfaces.

2.3.2 Prefiltering Results

For the core–rim and core–surface evolutionary predictors, two different stringencies for the list of putative homologues were applied: a minimum of ten homologues, each having at least 50% sequence identity to the query, and a minimum of thirty homologues, each having at least 60% sequence identity. Imposing these homologue stringencies helps ensure that the MSA used for calculations is of statistical significance, since too few or too dissimilar homologues in the alignment is likely to undermine the accuracy of classifications made.

2.3.3 Measures of Classifier Performance

As mentioned in Subsection 2.3.1, each of EPPIC’s predictors is considered a binary classifier for analysis and optimisation, and nopred calls are simply treated as xtal
calls. Since biological interfaces are typically of interest to the user, \texttt{bio} and \texttt{xtal} calls shall be positives and negatives, respectively. True positives (TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs) are briefly defined in Table 2.2.

<table>
<thead>
<tr>
<th>Call</th>
<th>Interface</th>
<th>Biological</th>
<th>Crystal</th>
</tr>
</thead>
<tbody>
<tr>
<td>bio</td>
<td>TP</td>
<td>FP</td>
<td></td>
</tr>
<tr>
<td>xtal</td>
<td>FN</td>
<td>TN</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: The definitions of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) with respect to the factual interface type, either biological or crystal, and the call made by the binary classifier, either \texttt{bio} or \texttt{xtal}.

Various measures exist to assess the performance of a binary classifier given the definitions in Table 2.2 [67]. Equation (2.1) defines the sensitivity of the classifier, which is the proportion of positives correctly labelled as positive. The denominator is expanded for clarity: datapoints correctly labelled as positive and datapoints incorrectly labelled as negative together form the set of all positives.

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \quad (2.1)
\]

Similarly, Equation (2.2) defines the specificity of the classifier, which is the proportion of negatives correctly labelled as negative. Again, the denominator is expanded for clarity: datapoints correctly labelled as negative and datapoints incorrectly labelled as positive together form the set of all negatives.

\[
\text{Specificity} = \frac{TN}{TN + FP} \quad (2.2)
\]

In Equation (2.3), the accuracy of the classifier is defined as the correct proportion of all predictions. There is, however, an issue with all hitherto-presented measures of performance: maximising the sensitivity or specificity alone has the side-effect of minimising the other. This can be illustrated by an extreme case wherein the classifier always makes a \texttt{bio} call: perfect sensitivity is obtained at the expense of any specificity. The accuracy is also flawed in that maximising it promotes skewing the classifier’s predictions to obtain as many correct predictions as possible, which is not necessarily a desirable result, especially for imbalanced datasets. Considering, for example, a hypothetical dataset with ninety positive cases and ten negative cases, a binary classifier trained by maximising accuracy would value sensitivity—the ability to identify those ninety positives—much
more highly than specificity, since even a near-zero specificity would only reduce the accuracy by approximately 10%.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.3)
\]

It follows that any useful measure of performance would have to impartially balance the sensitivity and the specificity. One such measure is presented in Equation (2.4), which defines the balanced accuracy as the arithmetic mean of the sensitivity and the specificity.

\[
\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \quad (2.4)
\]

Another interesting measure of performance is the Matthews correlation coefficient (MCC) [68], presented in Equation (2.5). The MCC was introduced in 1975 as a balanced measure of performance that works with datasets of imbalanced character. It is effectively a correlation coefficient measuring the agreement between the actual and the predicted classifications.

\[
\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.5)
\]

Over the course of the experiments carried out to generate the results presented below, it was found that the best method of assessing a classifier’s performance was to use the balanced accuracy. This measure, therefore, is used as the objective function for the optimisation of EPPIC’s parameters.

2.3.4 Optimised Parameters

After varying and testing sets of parameters for each of EPPIC’s classifiers, the results were sorted by the balanced accuracy achieved on the BioMany and XtalMany datasets. The top ten sets of parameters for each classifier are presented in the tables below.

In Table 2.3, the ten best parameter sets for the geometric classifier are presented. Only two parameters can be varied for this classifier: the burial cutoff, which is the proportion of solvent-accessible surface that has to be buried upon interface formation for a residue to be assigned to the interface core, and the call threshold, which is the minimum number of core residues in the interface that have to be identified for an interface to be deemed biological and given a \text{bio} call. The parameter optimisation that preceded the original
Table 2.3: The top ten parameter sets of burial cutoffs and call thresholds for the geometric classifier in EPPIC. While many measures of classifier performance are shown, the balanced accuracy is used for sorting purposes.

<table>
<thead>
<tr>
<th>Burial Cutoff</th>
<th>Call Threshold</th>
<th>Balanced Accuracy</th>
<th>Accuracy</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>90%</td>
<td>8</td>
<td>0.8635</td>
<td>0.8647</td>
<td>0.7992</td>
<td>0.9279</td>
<td>0.7344</td>
</tr>
<tr>
<td>85%</td>
<td>10</td>
<td>0.8612</td>
<td>0.8621</td>
<td>0.8152</td>
<td>0.9072</td>
<td>0.7264</td>
</tr>
<tr>
<td>90%</td>
<td>9</td>
<td>0.8555</td>
<td>0.8573</td>
<td>0.7533</td>
<td>0.9577</td>
<td>0.7283</td>
</tr>
<tr>
<td>95%</td>
<td>5</td>
<td>0.8544</td>
<td>0.8552</td>
<td>0.8085</td>
<td>0.9004</td>
<td>0.7126</td>
</tr>
<tr>
<td>90%</td>
<td>7</td>
<td>0.8525</td>
<td>0.8530</td>
<td>0.8266</td>
<td>0.8784</td>
<td>0.7064</td>
</tr>
<tr>
<td>85%</td>
<td>9</td>
<td>0.8524</td>
<td>0.8526</td>
<td>0.8419</td>
<td>0.8629</td>
<td>0.7052</td>
</tr>
<tr>
<td>95%</td>
<td>6</td>
<td>0.8501</td>
<td>0.8517</td>
<td>0.7558</td>
<td>0.9443</td>
<td>0.7146</td>
</tr>
<tr>
<td>90%</td>
<td>10</td>
<td>0.8484</td>
<td>0.8507</td>
<td>0.7213</td>
<td>0.9756</td>
<td>0.7230</td>
</tr>
<tr>
<td>95%</td>
<td>4</td>
<td>0.8469</td>
<td>0.8465</td>
<td>0.8704</td>
<td>0.8234</td>
<td>0.6942</td>
</tr>
<tr>
<td>90%</td>
<td>6</td>
<td>0.8445</td>
<td>0.8439</td>
<td>0.8793</td>
<td>0.8097</td>
<td>0.6900</td>
</tr>
</tbody>
</table>

release of EPPIC in 2012 showed an optimal burial cutoff of 95% and an optimal call threshold of 6 for the geometric classifier [24]. From the table, it can be observed that a burial cutoff of 90% and a call threshold of 8 achieve better performance. The result is also intuitive in the sense that relaxing the requirement to be considered a core residue comes alongside an increase in how many core residues must be present to consider an interface biological.

In Table 2.4, the ten best parameter sets for the core–rim evolutionary classifier are presented. In addition to the burial cutoff (as described for the geometric classifier), two further parameters can be varied: the reduced alphabet, which is an abbreviated way of referring to the number of letters in the reduced amino-acid alphabet used to calculate the entropy at sites in the MSA (taken from the possibilities proposed by Murphy et al. [60], as well as the six-letter alphabet proposed by Mirny and Shakhnovich [66]), and the call threshold, which is the scoring boundary below which an interface is given a bio call. These results are for datapoints where at least ten putative homologues, each having at least 50% sequence identity to the query, were identifiable.

Table 2.5 again shows the ten best parameter sets for the core–rim evolutionary classifier, much like in Table 2.4, but only considering datapoints for which a minimum of thirty putative homologues, each having at least 60% sequence identity to the query, was identifiable. The results in Table 2.5 can be accepted with greater confidence, since more data are used per considered datapoint to make the prediction. The typical values for the balanced accuracies also contrast between tables: selecting putative homologues more stringently allows for higher scores, further strengthening the case for the use of Table 2.5. EPPIC’s original parameter optimisation showed an optimal burial cutoff of 70% and an optimal call threshold of 0.75 for the core–rim evolutionary predictor [24].
Chapter 2 — Parameter Optimisation

Reduced amino-acid alphabets were not tested, and the ten-letter alphabet proposed by Murphy et al. [60] was used. From the table, it can be observed that the six-letter reduced amino-acid alphabet proposed by Mirny and Shakhnovich [66], a burial cutoff of 80%, and a call threshold of 0.90 produce better results.

<table>
<thead>
<tr>
<th>Reduced Alphabet</th>
<th>Burial Cutoff</th>
<th>Call Threshold</th>
<th>Balanced Accuracy</th>
<th>Accuracy</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>80%</td>
<td>0.90</td>
<td>0.8745</td>
<td>0.8727</td>
<td>0.8873</td>
<td>0.8617</td>
<td>0.7437</td>
</tr>
<tr>
<td>2</td>
<td>90%</td>
<td>0.90</td>
<td>0.8725</td>
<td>0.8874</td>
<td>0.7684</td>
<td>0.9766</td>
<td>0.7768</td>
</tr>
<tr>
<td>2</td>
<td>90%</td>
<td>0.85</td>
<td>0.8715</td>
<td>0.8867</td>
<td>0.7653</td>
<td>0.9777</td>
<td>0.7758</td>
</tr>
<tr>
<td>2</td>
<td>90%</td>
<td>0.80</td>
<td>0.8705</td>
<td>0.8861</td>
<td>0.7621</td>
<td>0.9789</td>
<td>0.7749</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>0.85</td>
<td>0.8700</td>
<td>0.8686</td>
<td>0.8795</td>
<td>0.8605</td>
<td>0.7351</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>0.75</td>
<td>0.8692</td>
<td>0.8706</td>
<td>0.8592</td>
<td>0.8792</td>
<td>0.7366</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>0.90</td>
<td>0.8688</td>
<td>0.8660</td>
<td>0.8899</td>
<td>0.8488</td>
<td>0.7316</td>
</tr>
<tr>
<td>6</td>
<td>80%</td>
<td>0.85</td>
<td>0.8680</td>
<td>0.8685</td>
<td>0.8675</td>
<td>0.8675</td>
<td>0.7325</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>0.80</td>
<td>0.8674</td>
<td>0.8673</td>
<td>0.8685</td>
<td>0.8664</td>
<td>0.7312</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>0.70</td>
<td>0.8670</td>
<td>0.8700</td>
<td>0.8466</td>
<td>0.8875</td>
<td>0.7344</td>
</tr>
</tbody>
</table>

Table 2.4: The top ten parameter sets of reduced alphabets, burial cutoffs, and call thresholds for the core–rim evolutionary classifier in EPPIC. These are the results obtained when using the less-stringent requirements for putative homologues: a minimum of ten, each having at least 50% sequence identity to the query. While many measures of classifier performance are shown, the balanced accuracy is used for sorting purposes.

In Table 2.6, the ten best parameter sets for the core–surface evolutionary classifier are presented. The parameters varied for this analysis are exactly as those varied in Tables 2.4 and 2.5, albeit for the core–surface predictor. These results are for datapoints where at least ten putative homologues, each having at least 50% sequence identity to the query, were identifiable.
Table 2.6: The top ten parameter sets of reduced alphabets, burial cutoffs, and call thresholds for the core-surface evolutionary classifier in EPPIC. These are the results obtained when using the less-stringent requirements for putative homologues: a minimum of ten, each having at least 50% sequence identity to the query. While many measures of classifier performance are shown, the balanced accuracy is used for sorting purposes.

Table 2.7 again shows the ten best parameter sets for the core-surface evolutionary classifier, much like in Table 2.6, but only considering datapoints for which a minimum of thirty putative homologues, each having at least 60% sequence identity to the query, was identifiable. The high-stringency results in Table 2.7, again, can be accepted with greater confidence and exhibit higher values of balanced accuracy. EPPIC’s original parameter optimisation showed an optimal burial cutoff of 70% and an optimal call threshold of $-1.00$ for the core-surface evolutionary predictor [24]. Reduced amino-acid alphabets were not tested, and the ten-letter alphabet proposed by Murphy et al. [60] was used. From the table, it can be observed that the six-letter reduced amino-acid alphabet proposed by Mirny and Shakhnovich [66], a burial cutoff of 80%, and a call threshold of $-0.90$ produce better results.

Perhaps more interesting than the slight differences in optimised burial cutoffs and call thresholds is the fact that the two-letter alphabet proposed by Murphy et al. appears so frequently in the top parameter sets. Indeed, if Tables 2.4 and 2.6 were to be extended to each show the top twenty-five parameter sets, the two-letter alphabet would be present in fourteen and eleven of them, respectively. This is not just the case for the low-stringency homologues, either: similarly extending Tables 2.5 and 2.7 to each show twenty-five parameter sets would result in twelve and nine appearances, respectively, of the two-letter alphabet.

In Figure 2.1, the effect of different amino-acid alphabets—the two-, four-, eight-, ten-, and fifteen-letter reduced amino-acid alphabets proposed by Murphy et al. [60]; the six-letter one proposed by Mirny and Shakhnovich [66]; and the standard, unreduced twenty-letter alphabet—on the balanced accuracy of the core-surface evolutionary predictor is
Table 2.7: The top ten parameter sets of reduced alphabets, burial cutoffs, and call thresholds for the core–surface evolutionary classifier in EPPIC. These are the results obtained when using the more-stringent requirements for putative homologues: a minimum of thirty, each having at least 60% sequence identity to the query. While many measures of classifier performance are shown, the balanced accuracy is used for sorting purposes.

<table>
<thead>
<tr>
<th>Reduced Alphabet</th>
<th>Burial Cutoff</th>
<th>Call Threshold</th>
<th>Balanced Accuracy</th>
<th>Accuracy</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>80%</td>
<td>−0.90</td>
<td>0.9142</td>
<td>0.9169</td>
<td>0.8951</td>
<td>0.9332</td>
<td>0.8300</td>
</tr>
<tr>
<td>15</td>
<td>80%</td>
<td>−0.90</td>
<td>0.9132</td>
<td>0.9155</td>
<td>0.8967</td>
<td>0.9297</td>
<td>0.8274</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>−0.85</td>
<td>0.9122</td>
<td>0.9149</td>
<td>0.8936</td>
<td>0.9308</td>
<td>0.8259</td>
</tr>
<tr>
<td>15</td>
<td>80%</td>
<td>−0.85</td>
<td>0.9114</td>
<td>0.9135</td>
<td>0.8967</td>
<td>0.9261</td>
<td>0.8233</td>
</tr>
<tr>
<td>6</td>
<td>80%</td>
<td>−0.95</td>
<td>0.9110</td>
<td>0.9142</td>
<td>0.8889</td>
<td>0.9332</td>
<td>0.8245</td>
</tr>
<tr>
<td>6</td>
<td>80%</td>
<td>−0.95</td>
<td>0.9110</td>
<td>0.9142</td>
<td>0.8889</td>
<td>0.9332</td>
<td>0.8245</td>
</tr>
<tr>
<td>6</td>
<td>80%</td>
<td>−1.05</td>
<td>0.9110</td>
<td>0.9155</td>
<td>0.8795</td>
<td>0.9426</td>
<td>0.8272</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>−0.80</td>
<td>0.9108</td>
<td>0.9129</td>
<td>0.8967</td>
<td>0.9250</td>
<td>0.8220</td>
</tr>
<tr>
<td>6</td>
<td>80%</td>
<td>−0.85</td>
<td>0.9106</td>
<td>0.9129</td>
<td>0.8951</td>
<td>0.9261</td>
<td>0.8219</td>
</tr>
<tr>
<td>2</td>
<td>80%</td>
<td>−0.90</td>
<td>0.9104</td>
<td>0.9142</td>
<td>0.8842</td>
<td>0.9367</td>
<td>0.8244</td>
</tr>
</tbody>
</table>

shown for various call thresholds. No matter the call threshold, the trends remain quite consistent. The twenty-letter alphabet does quite poorly, which is to be expected, since this maximal level of detail is likely to bring more noise than signal into the balance during the entropy calculations. The four-letter alphabet, however, does particularly badly, especially in comparison to its two- and six-letter neighbours, while the two-, six-, and fifteen-letter alphabets perform best.

It is puzzling, or at least unintuitive, that a complete disregard of an arguable 90% of the information in the MSA—going from twenty amino acids to two—is so beneficial to the predictions made by the evolutionary classifiers in EPPIC. With a view to elucidate this observation, the topic of reduced amino-acid alphabets forms much of the remainder of this thesis. For the sake of conciseness, a terminological adjustment should be made: henceforth, let an “n-alphabet” refer to an n-letter reduced amino-acid alphabet, where n can be any natural number from 2 to 20.

### 2.4 Conclusion

All three of EPPIC’s predictors—geometric, core–rim evolutionary, and core–surface evolutionary—have a large number of parameters that can be varied. While the default values for these parameters were chosen based on an optimisation carried out just prior to the original release of EPPIC in 2012 [24], this optimisation was not an exhaustive search and was performed using a small dataset. Two new datasets, BioMany and XtalMany, compiled computationally by Baskaran et al. [65], allowed for a more thorough analysis of
the effects of parameter values on EPPIC’s predictions. Through this, it was found that the geometric predictor performs best with a burial cutoff of 90% and a call threshold of 8, as shown in Table 2.3; that the core–rim evolutionary predictor performs best with the 6-alphabet proposed by Mirny and Shakhnovich [66], a burial cutoff of 80%, and a call threshold of 0.90, as shown in Table 2.5; and that the core–surface evolutionary predictor performs best with the 6-alphabet proposed by Mirny and Shakhnovich [66], a burial cutoff of 80%, and a call threshold of $-0.90$, as shown in Table 2.7. Additionally, 2-alphabets are surprisingly prominent in the highest-ranking parameter sets for the evolutionary predictors, a fact that drove much of the remainder of this thesis toward further investigation of the importance of reduced amino-acid alphabets on EPPIC’s predictive performance.
Chapter 3

Reduced Amino-Acid Alphabets

3.1 Rationale

In Section 2.4, Chapter 2 was concluded with some interesting findings: the 2-alphabet proposed by Murphy et al. [60] seems to come up surprisingly often in the highest-ranking parameter sets for the evolutionary predictors in EPPIC, and the 6-alphabet proposed by Mirny and Shakhnovich [66] is present in the best parameter sets. Clearly, further investigation into the effects that different amino-acid alphabets have on EPPIC’s performance was needed. EPPIC, however, had its seven amino-acid alphabets, those defined in Table 2.1, hardcoded in its source code. This meant that some changes to EPPIC had to first be made.

3.2 Published Alphabets

The first step was to collect reduced amino-acid alphabets that had already been proposed by the scientific community. The VisCoSe tool, published by Spitzer et al. [69], allows users to select from a list of compiled amino-acid alphabets and includes descriptions and citations for each. These alphabets, as well as the ones from Chapter 2, are listed in Table 3.1. Due to the style in which these alphabets were described by the VisCoSe tool, it was not immediately obvious that Alphabets 25 and 29 were, in fact, the same.

3.2.1 Methods

The eleven additional alphabets in Table 3.1, when compared to Table 2.1, were manually added into EPPIC’s source code, and EPPIC was run on the BioMany and XtalMany
Chapter 3 — Reduced Amino-Acid Alphabets

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>Source</th>
<th>Amino-Acid Alphabet</th>
<th>Size of Alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[60]</td>
<td>ACFGILMPSTVWY:DEHKNQR</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>[60]</td>
<td>AGPST:CILMV:DEHKNQR:FWY</td>
<td>4</td>
</tr>
<tr>
<td>21</td>
<td>[63]</td>
<td>AGPST:CFILMVVY:DEHKNQR</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>[63]</td>
<td>AGPST:CFWY:DEHKNQR:ILMV</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>[59]</td>
<td>ADEGHKNPQRST:CFILMVWY</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>[59]</td>
<td>AGHPRT:CFILMVVY:DEKNSQ</td>
<td>3</td>
</tr>
<tr>
<td>29</td>
<td>[69]</td>
<td>ADEGHKNPQRST:CFILMVWY</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.1: Sixteen amino-acid alphabets proposed by the scientific community. For each alphabet, the alphabet identifier—no longer necessarily the number of letters in the alphabet—is shown, as well as the authors who proposed it, the alphabet itself, and the number of groups in the alphabet. Alphabet 20 is an unreduced alphabet and, therefore, has no source. Note that Alphabets 25 and 29 are identical.

datasets once per alphabet. The burial cutoff used for core-residue assignment was 80%. When the authors of a PDB entry collect new data or rerefine a structure, the PDB entry is made obsolete and superseded by a new PDB entry. When this occurs to PDB entries in the BioMany or XtalMany datasets, the entry must be removed, since it is no longer considered a valid structure. The new, superseding structure does not necessarily conform to the standards applied when generating the datasets and, hence, cannot be added to either dataset. When this experiment was carried out, four PDB entries had been superseded: 2H7I, 2HCY, 3DMA, and 4M7N. The more-stringent criteria for putative homologues, a minimum of thirty, each having at least 60% sequence identity to the query, were imposed. Additionally, at this point, it was decided that, for these and future experiments, there is little use investigating both the core–rim and core–surface evolutionary predictors. The core–surface one offers a much more robust measure and will, thus, represent both predictors. Computations were carried out using the Merlin 4 HPCC at the Paul Scherrer Institute.
3.2.2 Results

Figure 3.1 shows the results of running the EPPIC core–surface evolutionary predictor on the BioMany and XtalMany datasets for each of the alphabets in Table 3.1. The results are shown for seven different core–surface call thresholds. No matter the threshold, Alphabet 2 clearly comes out on top, with Alphabet 15 following close behind, while Alphabet 25, identical to Alphabet 29, is the worst by a significant margin.

In Figure 3.1, it is also interesting that the alphabets contributing to high performance tend to have less variation for different call thresholds. Ideally, there would be a measure that could summarise not only the balanced accuracy, which, it might be helpful to recall, is the arithmetic mean of the sensitivity and specificity, but also the variation across call thresholds. Such a measure is described below in Subsection 3.2.3, and the remainder of the results from this chapter then follow in Subsection 3.2.4.
3.2.3 Receiver Operating Characteristic

The receiver operating characteristic (ROC) curve is a method of plotting particular aspects of the performance of a binary classifier against one another, all while varying the call threshold. Specifically, the true-positive rate, which is equivalent to the sensitivity, is plotted as a function of the false-positive rate, which is equivalent to unity minus the specificity, at various threshold settings. The ROC curve necessarily passes through two points: \((0, 0)\), since designating all interfaces as \textit{xtal} correctly identifies all negatives but also labels all positives as false negatives, and \((1, 1)\), since designating all interfaces as \textit{bio} correctly identifies all positives but also labels all negatives as false positives. The diagonal of the plot, stretching between the two aforementioned points, is the line below which the performance of the binary classifier is mathematically worse than that of random classification.

The best result for a binary classifier would be to have an ROC curve that rises sharply from \((0, 0)\) to \((0 + \epsilon, 1)\), stays flat across \(y = 1\), and ends at \((1, 1)\). Thus, in general, an ROC curve that tends to be higher or more left-lying is considered to be demonstrative of a better classifier. The area under the ROC curve (AUROC) is a measure of such characteristics: integrating the curve over \(x \in [0, 1]\) results in a number from 0 to 1, which represents the performance of the classifier irrespective of the call threshold chosen. A binary classifier with a higher AUROC value can be said to be a better classifier; however, when two binary classifiers have similar AUROC values but dissimilar ROC curves, one must consider the prevalence of positives \textit{versus} the prevalence of negatives, as well as the cost of false positives \textit{versus} the cost of false negatives, before deciding which classifier is more appropriate [70].

3.2.4 Further Results

In Figure 3.2, the ROC curves for the sixteen amino-acid alphabets from Table 3.1 are shown. The integration of each ROC curve has also been calculated, and the AUROC values are displayed in Table 3.2. With the highest AUROC, 0.9563, Alphabet 2 is confirmed as the best alphabet for the core–surface evolutionary predictor, and the identical Alphabets 25 and 29, with AUROCs of 0.7602 and 0.7601, respectively, are confirmed to be the worst alphabet. Note that the AUROC values for the two identical alphabets are not exactly the same. This is due to some stochasticity in the algorithm of the core–surface evolutionary predictor and will be addressed in more detail in Chapter 4.

Arguably, the most intriguing result of this section is that two different 2-alphabets manage to achieve both first- and last-place rankings. It would be promising to delve
Figure 3.2: The ROC curves for EPPIC’s core–surface evolutionary predictor for the published alphabets being evaluated in Section 3.2. The alphabet identifiers are those defined in Table 3.1.

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9536</td>
</tr>
<tr>
<td>4</td>
<td>0.9091</td>
</tr>
<tr>
<td>6</td>
<td>0.9360</td>
</tr>
<tr>
<td>8</td>
<td>0.9295</td>
</tr>
<tr>
<td>10</td>
<td>0.9217</td>
</tr>
<tr>
<td>15</td>
<td>0.9357</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.9045</td>
</tr>
<tr>
<td>21</td>
<td>0.9262</td>
</tr>
<tr>
<td>22</td>
<td>0.9048</td>
</tr>
<tr>
<td>23</td>
<td>0.9021</td>
</tr>
<tr>
<td>24</td>
<td>0.9126</td>
</tr>
<tr>
<td>25</td>
<td>0.7602</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>0.9127</td>
</tr>
<tr>
<td>27</td>
<td>0.9280</td>
</tr>
<tr>
<td>28</td>
<td>0.8921</td>
</tr>
<tr>
<td>29</td>
<td>0.7601</td>
</tr>
<tr>
<td>30</td>
<td>0.9167</td>
</tr>
<tr>
<td>31</td>
<td>0.9263</td>
</tr>
</tbody>
</table>

Table 3.2: The AUROC values for EPPIC’s core–surface evolutionary predictor for the published alphabets being evaluated in Section 3.2. The alphabet identifiers are those defined in Table 3.1.

Further into the topic of 2-alphabets by seeing how randomly-generated 2-alphabets fare in comparison.

3.3 Random 2-Alphabets

In order to test the effects of randomly-generated 2-alphabets, EPPIC had to be redesigned once again. The hardcoded amino-acid alphabets were kept for convenience,
but the necessity of their use was eliminated by allowing for custom alphabets to be defined in EPPIC’s configuration file. Once this was accomplished, it became much easier to benchmark the software using random 2-alphabets.

### 3.3.1 Methods

The process begins by randomly generating ten 2-alphabets. These alphabets are listed in Table 3.3. Note that the alphabets are all fairly balanced: no fewer than five (and, hence, no more than fifteen) amino acids are in any one category.

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>2-Alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ACDEFGIKLPSTVW:HMNQRY</td>
</tr>
<tr>
<td>1</td>
<td>ACDEFGIKMNPRST:HLQVWY</td>
</tr>
<tr>
<td>2</td>
<td>ACDEFGKLMQRTVW:HINSY</td>
</tr>
<tr>
<td>3</td>
<td>ACDEFGKNRS:HILMPQTVW</td>
</tr>
<tr>
<td>4</td>
<td>ACDEFGHIMNPWY:GKLQRSTV</td>
</tr>
<tr>
<td>5</td>
<td>ACDEFGHKNPQRSV:QIMTY</td>
</tr>
<tr>
<td>6</td>
<td>ACDEFGHKLQPSWY:GIMNRTV</td>
</tr>
<tr>
<td>7</td>
<td>ACDEHIKLNPQRSTV:FGNQWY</td>
</tr>
<tr>
<td>8</td>
<td>ACDELMNPTWY:FHIKQRSV</td>
</tr>
<tr>
<td>9</td>
<td>ACDFGHILQRSWY:EKNMFTV</td>
</tr>
</tbody>
</table>

Table 3.3: Ten randomly-generated 2-alphabets. For each alphabet, the alphabet identifier—which has been reset and now starts from 0—is shown, as well as the alphabet itself.

For each alphabet in Table 3.3, the EPPIC core–surface evolutionary predictor was run on the BioMany and XtalMany datasets. The burial cutoff used for core-residue assignment was 80%. The more-stringent criteria for putative homologues, a minimum of thirty, each having at least 60% sequence identity to the query, were imposed. Computations were carried out using the Merlin 4 HPCC at the Paul Scherrer Institute.

### 3.3.2 Results

Figure 3.3 shows the results of running the EPPIC core–surface evolutionary predictor on the BioMany and XtalMany datasets for each of the alphabets in Table 3.3. The results are shown for seven different core–surface call thresholds. No matter the threshold, it is fairly clear that there is not much variability in the balanced accuracy as a function of the random 2-alphabet used. Alphabet 4 ranks best, Alphabet 9 follows by a hair’s breadth, and Alphabet 7 is worst by a fair amount. Just as for the previous section, the ROC curves for these alphabets can be generated and their AUROC values used to better summarise the performance.
Chapter 3 — Reduced Amino-Acid Alphabets

In Figure 3.4, the ROC curves for the ten randomly-generated amino-acid alphabets from Table 3.3 are shown. The integration of each ROC curve has also been calculated, and the AUROC values are displayed in Table 3.4. With the highest AUROC, 0.9070, Alphabet 4 is confirmed as the best of the alphabets in Table 3.3, and Alphabet 7, with an AUROC of 0.7861, is confirmed to be the worst alphabet.

Although only ten random 2-alphabets were looked at, they were all well-balanced, and there does not appear to be much variability amongst them, using either the balanced accuracy or AUROC value as a measure of performance. The average AUROC value for the random 2-alphabets in this section is 0.8600 ± 0.0119. Why, then, does the 2-alphabet proposed by Murphy et al. [60] perform so much better than these (0.9536), and why does the 2-alphabet proposed by Wang and Wang [59] perform so much worse (0.7602)?
Figure 3.4: The ROC curves for EPPIC’s core–surface evolutionary predictor for the random 2-alphabets being evaluated in Section 3.3. The alphabet identifiers are those defined in Table 3.3.

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.8541</td>
</tr>
<tr>
<td>1</td>
<td>0.8706</td>
</tr>
<tr>
<td>2</td>
<td>0.8168</td>
</tr>
<tr>
<td>3</td>
<td>0.8818</td>
</tr>
<tr>
<td>4</td>
<td>0.9070</td>
</tr>
</tbody>
</table>

Table 3.4: The AUROC values for EPPIC’s core–surface evolutionary predictor for the random 2-alphabets being evaluated in Section 3.3. The alphabet identifiers are those defined in Table 3.3.

Section 3.4 - Transitioning 2-Alphabets

In Sections 3.2 and 3.3, it was found that the 2-alphabet proposed by Murphy et al. [60] performs about 10% better than random 2-alphabets, whereas the 2-alphabet proposed by Wang and Wang [59] performs about 10% worse than random 2-alphabets. How is it that two alphabets published by the scientific community could span such a range? In fact, there are only five differently-classified amino acids between the two alphabets—alanine, glycine, proline, serine, and threonine. Thus, to mutate one alphabet into the other requires five one-letter recategorisations. Every alphabet between these two
proposed 2-alphabets—in other words, the power set of these five changes—can be tested with EPPIC’s core–surface evolutionary predictor.

### 3.4.1 Methods

All intermediary 2-alphabets can be thought of as a presence or absence of the shift for each of the five amino acids whose categorisations differ between the alphabets. This allows for \(2^5 = 32\) possible 2-alphabets, which have been listed in Table 3.5. Note that Alphabet 0 is defined as the 2-alphabet proposed by Murphy et al. [60], while Alphabet V is defined as the 2-alphabet proposed by Wang and Wang [59]. Any alphabet between Alphabets 0 and V correspond to systematically-interpolated alphabets. For example, Alphabets 1 through 5 correspond to taking Alphabet 0 and moving one of alanine, glycine, proline, serine, or threonine, in that order, to the other group.

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>2-Alphabet</th>
<th>Alphabet Identifier</th>
<th>2-Alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ACFGILMPSTVWY:DEHKNQR</td>
<td>G</td>
<td>ADEGHKNPQR:CFILMSTVWY</td>
</tr>
<tr>
<td>1</td>
<td>ADEHKNQRCFGILMSTVWY</td>
<td>H</td>
<td>ADEGHKNQRS:CFILMPTVWY</td>
</tr>
<tr>
<td>2</td>
<td>ACFGILMSTVWY:DEGHKNQR</td>
<td>I</td>
<td>ADEGHKNQRT:CFILMPSVWY</td>
</tr>
<tr>
<td>3</td>
<td>ACFGILMSTVWY:DEHKNQRS</td>
<td>J</td>
<td>ADEHKNPQRS:CFILMTVWY</td>
</tr>
<tr>
<td>4</td>
<td>ACFGILMPTVWY:DEHKNQRS</td>
<td>K</td>
<td>ADEHKNPQRT:CFGILMSVWY</td>
</tr>
<tr>
<td>5</td>
<td>ACFGILMSVWY:DEHKNQRT</td>
<td>L</td>
<td>ADEHKQRTS:CFGILMPVWY</td>
</tr>
<tr>
<td>6</td>
<td>ADEGHKNQRCFILMSTVWY</td>
<td>M</td>
<td>ACFILMTVWY:DEGHKNPQRS</td>
</tr>
<tr>
<td>7</td>
<td>ADEHKNPQR:CFGILMSTVWY</td>
<td>N</td>
<td>ACFILMSVWY:DEGHKNPQRT</td>
</tr>
<tr>
<td>8</td>
<td>ADEHKNPQRS:CFGILMPTVWY</td>
<td>O</td>
<td>ACFILMPVWY:DEGHKNQRT</td>
</tr>
<tr>
<td>9</td>
<td>ADEHKNPQRT:CFGILMPSVWY</td>
<td>P</td>
<td>ACFGILMPSVY:DEHKNPQRS</td>
</tr>
<tr>
<td>A</td>
<td>ACFILMSTVWY:DEGHKNPQR</td>
<td>Q</td>
<td>ADEGHKNPQRS:CFILMTVWY</td>
</tr>
<tr>
<td>B</td>
<td>ACFILMPTVWY:DEGHKNQRS</td>
<td>R</td>
<td>ADEHKNPQRT:CFILMTVY</td>
</tr>
<tr>
<td>C</td>
<td>ACFILMPSVWY:DEHKNQRT</td>
<td>S</td>
<td>ADEGHKNQRTS:CFILMPVWY</td>
</tr>
<tr>
<td>D</td>
<td>ACFGILMPTVWY:DEHKNPQRS</td>
<td>T</td>
<td>ADEHKNPQRTS:CFGILMVWY</td>
</tr>
<tr>
<td>E</td>
<td>ACFGILMSVWY:DEHKNPQRT</td>
<td>U</td>
<td>ACFILMVWY:DEGHKNPQRS</td>
</tr>
<tr>
<td>F</td>
<td>ACFGILMPVWY:DEHKNPQRT</td>
<td>V</td>
<td>ADEGHKNPQRTS:CFILMVWY</td>
</tr>
</tbody>
</table>

Table 3.5: Thirty-two 2-alphabets, the first and last of which correspond to the 2-alphabets proposed by Murphy et al. [60] and Wang and Wang [59], respectively. Since there are only five differences in the categorisation of amino acids between the two 2-alphabets, alphabets between the first and last represent all possible alphabets “between” the two. For each alphabet, the alphabet identifier—which has, again, been reset—is shown, as well as the alphabet itself.

For each alphabet in Table 3.5, the EPPIC core–surface evolutionary predictor was run on the BioMany and XtalMany datasets. The burial cutoff used for core-residue assignment was 80%. The more-stringent criteria for putative homologues, a minimum of thirty, each having at least 60% sequence identity to the query, were imposed. Computations were carried out using the Merlin 4 HPCC at the Paul Scherrer Institute.
3.4.2 Results

Figure 3.5 shows the results of running the EPPIC core–surface evolutionary predictor on the BioMany and XtalMany datasets for each of the alphabets in Table 3.5. The results are shown for seven different core–surface call thresholds. No matter the threshold, the trend is the same: Alphabet 0 fares best, Alphabet V fares worst, and the transitioning alphabets between them drop off as more amino acids are recategorised. Just as for previous sections, the ROC curves for these alphabets can be generated and their AUROC values used to better summarise the performance.

In Figure 3.6, the ROC curves for the thirty-two transitioning 2-alphabets from Table 3.5 are shown. The integration of each ROC curve has also been calculated, and the AUROC values are displayed in Table 3.6.

Unlike in the previous sections, however, the noteworthy results here are not immediately visible from either Figure 3.5 or Figure 3.6. It was obvious that there would be some sort of decrease in performance as Alphabet 0 transitioned into Alphabet V, but the real problem was to identify which step in the transitioning was the most detrimental.
Figure 3.6: The ROC curves for EPPIC’s core–surface evolutionary predictor for the transitioning 2-alphabets being evaluated in Section 3.4. The alphabet identifiers are those defined in Table 3.5.

<table>
<thead>
<tr>
<th>Alphabet Identifier</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.9535</td>
</tr>
<tr>
<td>1</td>
<td>0.8960</td>
</tr>
<tr>
<td>2</td>
<td>0.9392</td>
</tr>
<tr>
<td>3</td>
<td>0.9465</td>
</tr>
<tr>
<td>4</td>
<td>0.9240</td>
</tr>
<tr>
<td>5</td>
<td>0.9287</td>
</tr>
<tr>
<td>6</td>
<td>0.8902</td>
</tr>
<tr>
<td>7</td>
<td>0.8891</td>
</tr>
<tr>
<td>8</td>
<td>0.8730</td>
</tr>
<tr>
<td>9</td>
<td>0.8643</td>
</tr>
<tr>
<td>A</td>
<td>0.9332</td>
</tr>
<tr>
<td>B</td>
<td>0.8975</td>
</tr>
<tr>
<td>C</td>
<td>0.9069</td>
</tr>
<tr>
<td>D</td>
<td>0.9169</td>
</tr>
<tr>
<td>E</td>
<td>0.9255</td>
</tr>
<tr>
<td>F</td>
<td>0.9002</td>
</tr>
</tbody>
</table>

Table 3.6: The AUROC values for EPPIC’s core–surface evolutionary predictor for the transitioning 2-alphabets being evaluated in Section 3.4. The alphabet identifiers are those defined in Table 3.5.
Looking at Figure 3.5, one could note that going from Alphabet 0 to Alphabet 1, which corresponds to recategorising the alanine, brings with it a sizeable drop in performance. But what effect does a recategorisation of the alanine have on alphabets that have already had some other amino acids recategorised? For example, the only difference between Alphabet M and Alphabet Q is the alanine, but both alphabets also already had glycine, proline, and serine recategorised. Looking at the difference between Alphabet M and Alphabet Q in Figure 3.5 shows, as expected, a decrease in performance. In fact, the decrease in the AUROC value from Alphabet 0 to Alphabet 1 is $0.0575$, and the decrease in balanced accuracy from Alphabet M to Alphabet Q is $0.0532$. The proximity of these values to one another acted as inspiration to model the AUROC as a function of the presence or absence of each part of the full transition from Alphabet 0 to Alphabet V.

### 3.4.3 Linear Model

A linear model was fit to the data, such that the AUROC value for EPPIC’s core-surface evolutionary predictor could be predicted given only the status of each amino acid having been recategorised. Equation (3.1) shows the linear model after training, where $\delta_X$ is the status of $X$ having been recategorised in the amino-acid alphabet being used. For example, Alphabet 0 has $\delta_A = \delta_G = \delta_P = \delta_S = \delta_T = 0$, while Alphabet V has $\delta_A = \delta_G = \delta_P = \delta_S = \delta_T = 1$.

$$\text{AUROC} \approx 0.0596\delta_A + 0.0255\delta_G + 0.0143\delta_P + 0.0385\delta_S + 0.0368\delta_T + 0.7926 \quad (3.1)$$

Figure 3.7 shows the result of using the linear model to predict the AUROC values, in blue, which, when compared to the true AUROC values, in pink, seems to match quite well. Indeed, the coefficient of determination, $R^2$, was 0.96. This is a very good result for a linear model, since the classification of amino acids in a 2-alphabet is certainly not linearly independent, which becomes evident when one considers that the only difference in having an amino acid change category is the change in the other amino acids with which it shares its category.

Further conclusions can be drawn by examining the coefficients in Equation (3.1). The coefficient of largest magnitude is that for alanine, which means that its categorisation is the most important of the five differences between Alphabet 0 and Alphabet V. The next most important is serine, followed closely by threonine, which makes sense considering their similar properties and prevalences [71]. The coefficient for glycine places fourth, and, finally, proline follows in the least-important position.
3.5 Conclusion

In this chapter, reduced amino-acid alphabets published by the scientific community were tested by running EPPIC’s core–surface evolutionary predictor on the BioMany and XtalMany datasets as a measure of performance and using the AUROC values as measures of classifier performance. The performance of random 2-alphabets was also tested, and an interesting finding was that, while random 2-alphabets exhibit little variability (0.8600 ± 0.0119), the 2-alphabet proposed by Murphy et al. [60] performs much better (0.9536), and the 2-alphabet proposed by Wang and Wang [59] performs much worse (0.7602). Despite this, the only difference between the two aforementioned alphabets is the categorisation of five amino acids: alanine, glycine, proline, serine, and threonine. All possible permutations between these alphabets were tested, the performance was shown to be more-or-less linearly dependent on the amino-acid categorisation, and alanine was identified as the amino acid whose categorisation was of greatest significance.
Chapter 4

Surface-Residue Sampling

4.1 Rationale

This chapter deviates from the investigative flow of the thesis but is a necessary explanatory tangent, since the following chapter will be predicated on the results from this one. EPPIC’s core–surface evolutionary predictor was introduced in Subsection 1.4.2, but was not described in detail.

EPPIC determines the solvent accessibility of residues before and after the formation of the protein–protein interface, and the ratio—the BSA-to-ASA ratio—is used to classify them as either “core” (residues fully buried upon interface formation), “rim” (residues partially buried upon interface formation), “surface” (surface residues not involved in an interface), or “other” (any non-interface, non-surface residues; for example, residues in a subunit’s core). The burial cutoffs used in classification are variable parameters. Once the categories for every residue have been assigned, EPPIC proceeds by searching for homologues. A minimum number of homologues, each having a minimum sequence identity to the query, are required for the evolutionary predictor to continue. The homologues are then input into a clustering algorithm for the purpose of reducing redundancy. This is done to ensure that each cluster of similar homologues is fairly characterised by one representative cluster member. These representatives are then aligned with one another in a large MSA. The information entropy for each column in the MSA is calculated, making sure to consider the (reduced) amino-acid alphabet of choice. The entropies of the core residues must now be compared to the entropies of the surface residues, and this forms the topic for this chapter.
4.2 Strategies for Surface-Residue Sampling

The first problem faced when comparing core-residue entropies to surface-residue entropies is that there tend to be far fewer core residues than surface residues. To handle this, the surface residues have to be sampled at random. The exact method by which this is carried out is not trivial, a fact which has led to the inclusion of two different strategies in EPPIC. Subsections 4.2.1 and 4.2.2 describe these two strategies, and their main difference is summarised in Table 4.1.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Core</th>
<th>Rim</th>
<th>Surface</th>
<th>Other</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classic</td>
<td>✘</td>
<td>✘</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Novel</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✘</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 4.1: A comparison of the two strategies for surface-residue sampling. In this table, “core” and “rim” residues are core and rim residues of the interface of interest, “surface” residues are residues on the surface of the protein, “other” residues are residues that do not fit in any other category (for example, residues in the protein’s hydrophobic core), and “interface” residues are residues on the surface that are involved in other interfaces of a minimum surface area. For each strategy and residue type, a ✓ or ✘ is shown to represent whether that strategy is permitted to add residues of that type to the surface-residue pool.

4.2.1 Classic Strategy

The classic strategy was the first to be implemented, and its algorithm for calculating the core–surface score is as follows. First, a non-zero core size is checked for: if the interface has no core residues, no prediction is returned. The average entropy of all core residues is computed. Then, surface residues are added to a pool as long as they are not involved in an interface (of at least some minimum surface area). It is ensured that a sufficient number of surface residues exist in the structure: if there are fewer than 1.5 times as many non-interface surface residues as core residues, the algorithm does not proceed. A number of surface residues equal to the number of existing core residues are then drawn from the pool, without replacement, and this sampling is carried out 10 000 times. The arithmetic mean of each sample is recorded, providing a “distribution” of 10 000 values. The mean and standard deviation of this distribution are used to compare the surface residues to the core residues through the Z-score test statistic.

Equation (4.1) defines the Z-score test statistic: the population mean of the core-residue entropies minus the sample mean of the surface-residue entropies, all divided by the sample standard deviation of the non-interface surface-residue entropies. Note that the
sample mean and sample standard deviation are different than usual, since they represent statistics from a sample that overrepresents the population due to the large number of samples.

\[
Z = \frac{\mu_{\text{core}} - \bar{x}_{\text{surface}}}{s_{\text{surface}}}
\]

(4.1)

This classic strategy for surface-residue sampling and score calculation has a few downsides. Firstly, it places a lot of weight on the parameters that define what constitutes a non-interface surface residue. While it is true that the surface itself will generally contain other interfaces or binding sites, it would be presumptuous to exclude surface residues simply because they lie in a region that may or may not be part of a biological interface. Secondly, since this strategy needs to refer to what is considered an interface-related residue for all other interfaces of the protein, all these objects need to have persistent references in the data model. This is not an issue when running an instance of EPPIC from start to finish, but it quickly becomes problematic when working directly with the database, which will be the case in the next chapter.

4.2.2 Novel Strategy

The novel strategy is vastly simpler than the classic one, but arguably contradicts the name of the predictor. Instead of meddling with the surface residues in the pool, the novel strategy leaves them as they are. In fact, it also considers the core residues and the rim residues to be part of the surface, and the only residues not part of the pool are those labelled as “other” residues. Of course, the definition of core residues is that they become buried upon interface formation, which means that they have to be exposed prior to interface formation and, hence, must also be on the surface. This is the interpretation of “surface” for this strategy, which is a superset of the meaning of “surface” in residue classification.

Other than this major difference, things are mostly as they are with the classic strategy. A non-zero core size is checked for, the core-residue entropies are averaged, all non-other residues are added to a pool, and, since there are no longer any removals from the pool, it is ensured that at least 2.0 times (rather than 1.5) as many surface residues are in the pool as there are core residues in the interface. The sampling is then performed as before, and Equation (4.1) is used to calculate a test statistic. The test statistic is now more like a real Z-score: it represents how likely it is that the values, which come from the population and are being compared to the population as a whole—in
this case, the core of the interface compared to the entire surface of the protein—have a statistically-differing mean.

4.3 Methods

For both of the strategies detailed in Section 4.2, the EPPIC core–surface evolutionary predictor was run on the BioMany and XtalMany datasets. The burial cutoff used for core-residue assignment was 80%, and the reduced amino-acid alphabet used was the 6-alphabet proposed by Mirny and Shakhnovich [66]. The more-stringent criteria for putative homologues, a minimum of thirty, each having at least 60% sequence identity to the query, were imposed. Computations were carried out using the Merlin 4 HPCC at the Paul Scherrer Institute.

4.4 Results

Presenting the performance of the classifier using the balanced accuracy as a measure no longer makes sense, since different scoring systems cannot be compared using the same call thresholds. For this reason, the AUROC values must be used as a measure of performance, which ensures that the potential of each classifier is being tested, without bias, across all thresholds.

In Figure 4.1, the ROC curves for each surface-sampling strategy in Section 4.2 are shown. The integration of each ROC curve has also been calculated, and these AUROC values are displayed in the figure’s legend. The classic strategy for sampling surface residues resulted in an AUROC value of 0.9338, performing slightly better than the novel strategy, which obtained an AUROC value of 0.9265. This is an acceptable result: the slight decrease in performance, which may well be within the noise of the data, is more than made up for by the simplicity of the strategy, its reduced computational complexity, and the possibility of tackling cases where the classic strategy has failed due to an insufficient number of surface residues (for example, in coiled-coil structures). For this reason, the next chapter, which will involve working directly with the database behind EPPIC, will adopt the novel strategy for the sampling of surface residues.

4.5 Conclusion

Due to both the sheer number and the variability in character of the residues on a protein’s surface, EPPIC’s core–surface evolutionary predictor requires some statistical
sampling to take place before the sequence entropies for the surface residues can be compared to those of the core residues. The classic strategy for this was to exclude regions of the surface that could be involved in binding sites or interfaces before sampling, whereas the novel strategy takes the entire surface of the protein, even the core and rim residues in the interface of interest, into account. A comparison of these two strategies in this chapter showed that the classic one outperforms the novel one by a very narrow margin of 0.78%, which is an acceptable loss to incur when weighed against the gains: increased simplicity, reduced computational complexity, and handling structures with few surface residues. Thus, the novel method is adopted in the following chapter of this thesis.
Chapter 5

2-Alphabets

5.1 Rationale

In Chapter 2, the parameters in EPPIC were optimised based on a maximisation of performance for the BioMany and XtalMany datasets. Of EPPIC’s built-in amino-acid alphabets, shown in Table 2.1, the 6-alphabet proposed by Mirny and Shakhnovich [66] performed best. In Chapter 3, more reduced amino-acid alphabets, which had been proposed by the scientific community, were tested and compared to randomised 2-alphabets. The results showed that the 2-alphabet proposed by Murphy et al. [60] performs far better than random 2-alphabets, but the 2-alphabet proposed by Wang and Wang [59] performs much worse.

The linear model from Subsection 3.4.3 set out to explain this by deconvolving the five differentially-grouped amino acids—alanine, glycine, proline, serine, and threonine—between the two 2-alphabets, and it was found that alanine was the amino acid whose categorisation had the greatest effect on the classifier’s performance. However, this linear model failed to take into consideration the interdependence of amino-acid categorisations: it is the other amino acids with which a category is shared that characterises an alphabet, not the individual categorisation of each amino acid. For this reason, this final chapter of the thesis focusses on generating and benchmarking all possible 2-alphabets and fitting a model of greater precision to the data.

5.2 Methods

Unfortunately, even given the modified version of EPPIC that allows for custom alphabets, any brute-force approach that tests all conceivable amino-acid alphabets would be
an insurmountable task. Therefore, the first step was to engineer a faster way of evaluating 2-alphabets. Working directly with the database that contains the computations for all current PDB entries would make the procedure much faster, since EPPIC’s preliminary calculations (finding interfaces, calculating ASAs, searching for homologues, creating MSAs, etc.), normally repeated for each run, would be bypassed. Using an application programming interface (API) called the Java Persistence API (JPA), a connection to the relational database management system (RDBMS) is made, a list of all Protein Data Bank identifiers (PDB IDs) in the BioMany and XtalMany datasets is passed to the database as a set of queries, and the referenced objects are returned.

For each PDB ID in the datasets, three stages of computation need to be carried out: firstly, acquiring the aligned homologues from the database, adding them to an MSA structure, and calculating the sequence entropy (with the selected amino-acid alphabet) for each column; secondly, appropriately mapping the PDB ID’s associated interface from the dataset to the database, classifying the residues geometrically by their burial ratio; and, finally, pooling and sampling the residues to calculate the final score for the interface. Once these scores are available, a bio or xtal call is obtained by comparing the score to the call threshold. Everything but the sequence-entropy calculation and sampling is done only once: the MSAs and residue classifications for both entire datasets, as well as any other required data, are serialised and saved to file. Further amino-acid alphabet evaluations begin by reading in this serialised file to save time.

Due to the time that had elapsed between this experiment and previous experiments, some PDB IDs had to be removed from the BioMany and XtalMany datasets. 2ELF, 2FRM, and 2O37 were removed because they had been superseded, while 2PYE, 2VLM, 3QH3, 3RFN, 3RY9, 3VVU, 4FQ2, 4L4V, and 4MAY were removed because one or more of the chains referred to by the datasets had no UniProt sequence mapping. For all other PDB IDs, the computations were carried out using the Euler HPCC at ETH Zürich.

5.2.1 Stirling Numbers of the Second Kind

Prior to beginning computations for this chapter, even listing all of the potential amino-acid alphabets proved difficult. To illustrate the complexity of this, even once the number of categories for the alphabet is chosen, the question of how many amino acids to add to each category remains, not to mention determining which particular amino acids to add.

James Stirling, an eighteenth-century Scottish mathematician, discovered the so-called “Stirling numbers of the second kind”, which define the number of ways to partition a set of \( n \) objects into \( k \) non-empty subsets. The explicit formula for their calculation is
shown in Equation (5.1) [72]. These numbers, as well as the formula to generate them, are very useful for the purposes of this experiment, since amino-acid alphabets are, in fact, partitions of a twenty-element set into $k$ non-empty subsets, where $2 \leq k \leq 20$ [72].

\[
S(n, k) = \binom{n}{k} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n
\]  

(5.1)

Figure 5.1 shows all possible partitions of a set of four elements. The topmost partition, in red, is not applicable to amino-acid alphabets, because $k = 1$. The green partitions correspond to partitions where $k = 2$ with three elements in one subset and one element in the other subset. The orange partitions also correspond to partitions where $k = 2$, but with two elements in each subset. The six purple partitions are partitions where $k = 3$, which can only be distributed by placing two elements in one subset and one element in each of the other two subsets. Finally, the grey partition at the bottom of the figure is a partition with $k = n = 4$, which has only one possible distribution: one element in each subset.

It is clear that, for increasing $n$, these partitions become increasingly complex. Using the formula in Equation (5.1), it can be shown that just the 2-alphabets, which must
have \( n = 20 \) and \( k = 2 \), amount to 524,287 possibilities \([74]\). Extending the scope to all possibilities produces an incredible 51,724,158,235,372 amino-acid alphabets \([74]\). Clearly, it would be infeasible to test all of these, and, for this reason, the scope of this chapter is restricted to the 2-alphabets.

### 5.3 Results

After writing the program as described in Section 5.2, it was now possible to evaluate the performance of one amino-acid alphabet in approximately one second (on one core). After about one week of compute time, all 2-alphabets had had their balanced accuracies calculated, and the results were quite surprising. The 2-alphabet proposed by Murphy et al. \([60]\) fares relatively well against all possible 2-alphabets, achieving a rank of 98. This 2-alphabet, as well as the one proposed by Wang and Wang \([59]\), which achieved a rank of 513,180, are shown alongside the ten best and ten worst 2-alphabets in Table 5.1.

### 5.4 Linear Model

Much like with the results from Section 3.4, a linear model can be fit to the results of this chapter. As mentioned in Section 5.1, amino-acid categorisations in amino-acid alphabets are not independent of one another: it is the other amino acids with which a category is shared that characterises an alphabet, not the individual categorisation of each amino acid. Therefore, a more elaborate form of linear model was used in this section. Instead of looking for the presence or absence of a particular amino acid, the presence or absence of all possible duos of amino acids were considered. Let a duo of amino acids be defined as two amino acids, and let the presence of any such duo indicate that the two component amino acids share the same categorisation in the amino-acid alphabet. For example, the presence of the ST duo—always written in alphabetical order—would mean that serine and threonine are grouped together in the alphabet in question.

A linear model was fit to the data, such that the balanced accuracy for EPPIC’s core–surface evolutionary predictor could be predicted given only the status of each amino-acid duo’s presence. Equation (5.2) shows the general structure of the linear model, where \( \alpha_{XY} \) is the coefficient of the duo \( XY \) after training and \( \delta_{XY} \) its status: 0 if absent, and 1 if present. \( \alpha_0 \) is the bias of the model, referring to the theoretical balanced accuracy that would be present if all amino acids were segregated into their own categories in the
alphabet. Terms in the model whose coefficients had magnitudes of less than $2 \times 10^{-3}$ were removed. Even after these simplifications, the coefficient of determination, $R^2$, was 0.96.

\[ \text{Balanced Accuracy} \approx \sum_{x<y} \alpha_{xy}\delta_{xy} + \alpha_0 \]  

(5.2)

After training, the coefficients of this model were examined. Figure 5.2 shows the ten coefficients of highest magnitude on each side of the spectrum, with the duos whose presences are beneficial in green and the duos whose presences are detrimental in red. It makes sense that duos composed of similar amino acids—such as isoleucine and valine, isoleucine and leucine, or leucine and valine—would be beneficial to group together, given

<table>
<thead>
<tr>
<th>Alphabet Rank</th>
<th>2-Alphabet</th>
<th>Balanced Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACDFGILMPSTVWY:EHKNQR</td>
<td>0.9014</td>
</tr>
<tr>
<td>2</td>
<td>ACDFGILMPSTVWY:EKQR</td>
<td>0.9001</td>
</tr>
<tr>
<td>3</td>
<td>ACDFGILMPSTVWY:EN</td>
<td>0.8995</td>
</tr>
<tr>
<td>4</td>
<td>ACDFGILMPSTVWY:DEHN</td>
<td>0.8988</td>
</tr>
<tr>
<td>5</td>
<td>ACDFGILMPSTVWY:ENW</td>
<td>0.8986</td>
</tr>
<tr>
<td>6</td>
<td>ACDFGILMPSTVWY:EHKNQR</td>
<td>0.8985</td>
</tr>
<tr>
<td>7</td>
<td>ACDFGILMPSTVWY:EKQR</td>
<td>0.8985</td>
</tr>
<tr>
<td>8</td>
<td>ACDFGILMPSTVWY:ENW</td>
<td>0.8982</td>
</tr>
<tr>
<td>9</td>
<td>ACDFGILMPSTVWY:EHKNQR</td>
<td>0.8982</td>
</tr>
<tr>
<td>10</td>
<td>ACDFGILMPSTVWY:EKQR</td>
<td>0.8979</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>98</td>
<td>ACDFGILMPSTVWY:DEHK</td>
<td>0.8938</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>513 180</td>
<td>ADEGHKLPQRSTVW:CFILMV</td>
<td>0.7277</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>524 278</td>
<td>ADEFGKLMNPQRSTVW:IW</td>
<td>0.5680</td>
</tr>
<tr>
<td>524 279</td>
<td>ADEFGKLMNPQRSTVW:IM</td>
<td>0.5664</td>
</tr>
<tr>
<td>524 280</td>
<td>ADEFGKLMNPQRSTVW:IMW</td>
<td>0.5648</td>
</tr>
<tr>
<td>524 281</td>
<td>ADEFGKLMNPQRSTVW:CIW</td>
<td>0.5645</td>
</tr>
<tr>
<td>524 282</td>
<td>ADEFGKLMNPQRSTVW:FI</td>
<td>0.5642</td>
</tr>
<tr>
<td>524 283</td>
<td>ADEFGKLMNPQRSTVW:IM</td>
<td>0.5632</td>
</tr>
<tr>
<td>524 284</td>
<td>ADEFGKLMNPQRSTVW:IMW</td>
<td>0.5630</td>
</tr>
<tr>
<td>524 285</td>
<td>ADEFGKLMNPQRSTVW:CI</td>
<td>0.5629</td>
</tr>
<tr>
<td>524 286</td>
<td>ADEFGKLMNPQRSTVW:IM</td>
<td>0.5618</td>
</tr>
<tr>
<td>524 287</td>
<td>ADEFGKLMNPQRSTVW:IMW</td>
<td>0.5605</td>
</tr>
</tbody>
</table>

Table 5.1: The ten best and ten worst 2-alphabets, as well as two 2-alphabets proposed by the scientific community for comparison. The rank of each alphabet and the balanced accuracy (resulting from its use in EPPIC’s core-surface predictor) are shown alongside the 2-alphabet itself.
Figure 5.2: The ten best (in green) and ten worst (in red) duos of amino acids. For each of these duos, the presence (for the ones in green) or absence (for the ones in red) of the duo—in other words, having the duo's component amino acids in the same or in different groups in the 2-alphabet, respectively—is beneficial to the performance of EPPIC’s core–surface predictor across the BioMany and XtalMany datasets.

Although anecdotal in nature, this result can be demonstrated to, indeed, be the case for the 2-alphabet proposed by Murphy et al. [60], at least for the sake of applying the finding to a known baseline. The proposed alphabet, unchanged, achieves a balanced accuracy of 0.8938. Moving aspartic acid or glutamic acid to the other group increases the result to 0.9014 or 0.8959, respectively. Moving both amino acids to the other group, however, results in a balanced accuracy of 0.8910, similar to the original value.
5.5 Conclusion

In this chapter, the algorithm behind EPPIC was reimplemented using a direct channel to communicate with the precomputed database, all 524,287 possible 2-alphabets were found and benchmarked using the core–surface predictor across the BioMany and Xtal-Many datasets, and a linear model that considered duos of amino acids being present in the same group was trained. As expected, it was found that, when similar amino acids were grouped together, the predictions were improved. By the same token, it was found that grouping dissimilar amino acids together caused detriment to the predictions. Surprisingly, the DE and KR duos were the most harmful to the predictions, despite the undeniable similarity between the members of each of them. Future work should address the basis for this observation.
Chapter 6

Conclusion

6.1 Summary

All three of EPPIC’s predictors—geometric, core–rim evolutionary, and core–surface evolutionary—have a large number of variable parameters. While the default values for these parameters were chosen based on an optimisation carried out just prior to the original release of EPPIC in 2012 [24], this optimisation was not an exhaustive search and was performed using a small dataset. Two new datasets, BioMany and XtalMany, compiled computationally by Baskaran et al. [65], allowed for a more thorough analysis of the effects of parameter values on EPPIC’s predictions. Through this, it was found that the geometric predictor performs best with a burial cutoff of 90% and a call threshold of 8; that the core–rim evolutionary predictor performs best with the 6-alphabet proposed by Mirny and Shakhnovich [66], a burial cutoff of 80%, and a call threshold of 0.90; and that the core–surface evolutionary predictor performs best with the 6-alphabet proposed by Mirny and Shakhnovich [66], a burial cutoff of 80%, and a call threshold of $-0.90$. Additionally, 2-alphabets are surprisingly prominent in the highest-ranking parameter sets for the evolutionary predictors, a fact that drove much of the remainder of this thesis toward further investigation of the importance of reduced amino-acid alphabets on EPPIC’s predictive performance.

Next, reduced amino-acid alphabets published by the scientific community were tested by running EPPIC’s core–surface evolutionary predictor on the BioMany and XtalMany datasets and using the AUROC values as measures of classifier performance. The performance of random 2-alphabets was also tested, and an interesting finding was that, while random 2-alphabets exhibit little variability ($0.8600 \pm 0.0119$), the 2-alphabet proposed by Murphy et al. [60] performs much better (0.9536), and the 2-alphabet proposed by Wang and Wang [59] performs much worse (0.7602). Despite this, the only difference

55
between the two aforementioned alphabets is the categorisation of five amino acids: alanine, glycine, proline, serine, and threonine. All possible “transitioning” 2-alphabets between these alphabets were tested: the performance was shown to be more-or-less linearly dependent on each of the five amino-acid categorisations, and alanine was identified as the amino acid whose categorisation was of the most significance.

As a bit of a tangent, the thesis then sought to focus on simplifying the method by which surface residues were sampled for the scoring component of EPPIC’s core–surface predictor. Due to both the sheer number and the variability in character of the residues on a protein’s surface, statistical sampling is required before the sequence entropies for the surface residues can be compared to those of the core residues. In the classic strategy, regions of the surface that could be involved in binding sites or interfaces were excluded before sampling, whereas the novel strategy takes the entire surface of the protein into consideration, even the core and rim residues in the interface of interest. A comparison of these two strategies in this chapter shows that the classic one outperforms the novel one by a very narrow margin of 0.78%, which is an acceptable loss to incur when weighed against the upsides: increased simplicity, reduced computational complexity, and the possibility of tackling cases where the classic strategy has failed due to an insufficient number of surface residues (for example, in coiled-coil structures). Thus, the novel method was adopted for Chapter 5.

In the final chapter, the algorithm behind EPPIC was reimplemented using a direct channel of communication with the precomputed database. All 524,287 possible 2-alphabets were then found and benchmarked using the EPPIC core–surface predictor across the BioMany and XtalMany datasets, and a linear model that considered duos of amino acids being present in the same group was trained. Confirming expectations, it was found that grouping similar amino acids together benefited predictions, while grouping dissimilar amino acids together was decidedly harmful to the predictions. Surprisingly, the DE and KR duos were the most detrimental to the predictions, despite the undeniable similarity between the members of each of them.
Appendix A

Hydrogen-Bond Detection

A.1 Hydrogen Bonding

A hydrogen bond is “an attractive interaction between a hydrogen atom from a molecule or a molecular fragment D–H, in which D is is more electronegative than H, and an atom or a group of atoms in the same or a different molecule, in which there is evidence of bond formation” [75]. Typically, a hydrogen bond is denoted as $D\cdot\cdot\cdot A\cdot\cdot\cdot A$, where the three dots represent the bond [75]. The name “hydrogen bond” is a bit misleading: hydrogen bonds are dipole–dipole interactions more than anything else and should not be confused with true bonds such as covalent bonds. The heavy atom to which the hydrogen involved in the bond is connected is called the donor atom and is represented by D. The heavy atom to which the hydrogen is paired—via the hydrogen bond—is called the acceptor atom and is represented by A. The parent heavy atoms of the donor and acceptor are represented by $DD$ and $AA$, respectively.

Hydrogen bonding is a very important phenomenon in protein structure, for it is because of these interactions that a subunit of a protein remains folded. The primary structure of the protein—its amino-acid sequence—dictates the secondary structures that form [1]. An alpha helix’s formation is stabilised by hydrogen bonding between the N–H group of an amino acid and the O=C group of the amino acid four residues earlier [76]. Similarly, parallel and antiparallel beta strands are held together to form beta sheets by hydrogen bonding between the N–H group of an amino acid and the O=C group of another on the adjacent strand [77]. Protein tertiary structure and quaternary structure are also impacted by hydrogen bonding: energetic minimisation drives proteins to their most stable conformation in space, given the secondary structures that first formed, and different subunits form protein–protein interfaces due to interaction partly caused by hydrogen bonding across the interface [78].
A.2 Rationale

Since hydrogen bonding is so key to protein structure, hydrogen-bond detection would be very useful in the prediction of biological interfaces. Proteins, Interfaces, Structures and Assemblies (PISA) [35] already, albeit indirectly, takes hydrogen bonding into consideration, since their algorithm to predict whether an interface is stable in solution considers the estimated free energy of the interaction. A lower energy supports the designation of the interface as biological, but the lower energy is, at least in part, caused by the stability that hydrogen bonds across the interface impart. The Evolutionary Protein–Protein Interface Classifier (EPPIC) [24] could also use hydrogen bonding to help in its geometric prediction. If one can detect hydrogen bond across an interface, it would be a good indication that the interface is biological. For this reason, this appendix documents the creation of a hydrogen-bond detector for EPPIC.

A.3 Definition

The first step was to determine what defines a hydrogen bond, which proved to be particular challenging, since there exists so much variation in the definitions of hydrogen bonding. Arunan et al., in 2011, provided a list of specific criteria to define what constitutes and characterises a hydrogen bond [75]. The downside, however, was that many of these rules, such as ones based on attributes unobtainable from the PDB file format, were not possible to implement in EPPIC. In 1984, Baker and Hubbard documented hydrogen bonding in secondary structure and showed that the main chain of a polypeptide can form hydrogen bonds through the N-H and O=C groups, with the O=C group able to accept two hydrogen bonds [79]. They also noted that “all polar sidechains, with the exception of tryptophan, can form multiple hydrogen bonds” [79]. But these definitions still did not address the required geometry for hydrogen bonding. In 2010, Hubbard and Haider provided some such information: the H···O=C angle is usually between 130° and 170°, with a clear preference at approximately 150°; the H···O distance is always between 1.6 Å and 2.5 Å; and the N-H···O angle is nearly always between 120° and 180° [80].

A.4 Implementation

Using only the three ranges of geometric values from Section A.3 to detect hydrogen bonds resulted in far too many being detected. At this point, another approach was taken. HBPLUS is a software package developed by McDonald and Thornton in 1994 [81]. They undertook an empirical approach to hydrogen-bond detection by
looking at protein structures in the PDB and creating particular criteria specific to hydrogen bonding in proteins. The rules listed in the documentation for HBPLUS [82] were added to EPPIC, such that, for each interface calculation, hydrogen bonds from one side of the interface to the other are identified and counted. These are attributed to the residues involved in the hydrogen bond, which are data that can be used in the future in EPPIC’s geometric predictor and possibly incorporated into a model that combines the core-residue count for the interface (and, eventually, even the insight from core-residue decay presented in Appendix B).

Upon validation, however, it appeared that the EPPIC implementation of the HBPLUS rules did not achieve the same results. The EPPIC implementation output some hydrogen bonds that HBPLUS did not find, but HBPLUS tended to find quite a few more than EPPIC. At this point, another direction was pursued, and it was decided that HBPLUS should simply be called by EPPIC. Some of EPPIC was recoded to allow the program to run HBPLUS for the PDB file. HBPLUS’s output is parsed by EPPIC’s internal code, and the hydrogen bonds are added to the component residues appropriately.

An issue with this implementation is how to ensure that the user of EPPIC also has HBPLUS installed. Due to licensing restrictions, HBPLUS cannot be packaged with EPPIC; thus, HBPLUS was implemented as follows: EPPIC checks for HBPLUS (whose path must be added to the EPPIC parameter file), and, if present, it is run and its output is parsed. If HBPLUS is not available for use, the EPPIC implementation of HBPLUS’s hydrogen-bonding criteria is used.

A.5 Conclusion

This appendix’s purpose was to document the steps taken to implement a hydrogen-bonding detector into EPPIC’s geometric predictor. In the end, EPPIC tries to run HBPLUS, but, if HBPLUS is unavailable, an alternative implementation of HBPLUS’s hydrogen-bonding criteria is used within EPPIC. The results with the non-HBPLUS implementation do not overlap completely (data not shown). More investigation would be required to determine the cause of the differences between the EPPIC implementation and the HBPLUS implementation. EPPIC could then be trained to use the hydrogen-bond information as a criterion for classification. This would likely be built alongside the core-size discriminator in the geometric classifier (and, eventually, even the insight from core-residue decay presented in Appendix B).
Appendix B

Core-Residue Decay

B.1 Theory

While experiments were being carried out for Chapter 2, a brief investigation into the potential for core-residue decay to act as a criterion for interface classification in EPPIC was undertaken. The theory is simple and goes as follows. The geometric classifier in EPPIC only considers the number of core residues when predicting whether an interface is bio or xtal. While performing the parameter optimisation in Subsection 2.3.4, information was available for core-residue assignment across a spectrum of burial cutoffs. Recall that core residues are assigned based on their BSA-to-ASA ratio, the proportion of solvent-accessible surface area that becomes buried upon interface formation. Recording the number of core residues as a function of the burial cutoff allows for a gradient to be calculated, which, it was theorised, could be telling of how likely the interface is to be biologically relevant. For example, it could be that, since biologically-relevant interfaces are conserved by evolution, they have more highly-buried residues involved in the interface and fewer modestly-buried residues, whereas a crystal contact may well exhibit a more linear or stochastic dropoff.

Every interface in the BioMany and XtalMany datasets had its number of core residues computed for each of the following burial cutoffs: 50%, 55%, 60%, 65%, 70%, 75%, 80%, 85%, 90%, 95%, and 100%. Linear models, with the number of core residues as a function of the burial cutoff, were then fit, one per datapoint. If the coefficient of determination, $R^2$, was above 0.96, the slope and $y$-intercept of the model were noted down. Let the slopes and $y$-intercepts of these models be termed the slopes and intercepts of core-residue decay.
B.2 Results

Figure B.1 is a histogram that has been included in this appendix for the sake of comparison. It shows the distribution of the number of core residues in an interface for a burial cutoff of 95%, which is the criterion used (with a call threshold of 6) in EPPIC’s geometric classifier. Note that the datapoints from the XtalMany dataset, shown in red, are from a visibly-different distribution than the datapoints from the BioMany dataset, shown in green. After choosing a criterion, the less overlap there is between these two distributions, the more powerful the classifier can be.

Figure B.2 is a histogram showing the distribution of decay slopes, that is, the distribution of the slopes from the models with $R^2 > 0.96$. Figure B.3 is a similar histogram showing the distribution of decay intercepts. Note that Figures B.2 and B.3 do not have as many datapoints as Figure B.1, because only datapoints whose linear model was fit with an $R^2$ of at least 0.96 had their slopes and $y$-intercepts added to the histogram.

Upon visual inspection of Figures B.2 and B.3, it is plain to see that there is more overlap between bio and xtal distributions than there is in Figure B.1. This likely means that a simple linear discriminator based on either of these criteria would not
Figure B.2: The distribution of the slopes from the linear models of the core size as a function of the burial cutoff for biological interfaces from the BioMany dataset, shown in green, and crystal contacts from the XtalMany dataset, shown in red. Note that the slopes are all negative because increasing the burial cutoff results in a more stringent criterion for core-residue assignment, which decreases the core size.

be able to distinguish between biological interfaces and crystal contacts in the range of values where the result is not clear. Indeed, Table B.1 shows the enumeration of non-overlapping bios (as a fraction of all bios), non-overlapping xtalss (as a fraction of all xtalss), and overlapping bios and xtalss (as a fraction of all datapoints) there are in each of the histograms. The simple count of core residues has a lower overlap than the other criteria and has more balance between non-overlapping categories. Both of these are properties that would help with classification. Unfortunately, there was insufficient time available to continue exploring options, but it is very possible that, using an approach

<table>
<thead>
<tr>
<th>Criterion</th>
<th>bio</th>
<th>xtal</th>
<th>bio + xtal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Size</td>
<td>0.7052</td>
<td>0.7156</td>
<td>0.2895</td>
</tr>
<tr>
<td>Decay Slopes</td>
<td>0.5698</td>
<td>0.6650</td>
<td>0.3767</td>
</tr>
<tr>
<td>Decay Intercepts</td>
<td>0.6199</td>
<td>0.7040</td>
<td>0.3329</td>
</tr>
</tbody>
</table>

Table B.1: For three different criteria—the core size, the decay slope, and the decay intercept—the non-overlapping bios (as a fraction of all bios), non-overlapping xtalss (as a fraction of all xtalss), and overlapping bios and xtalss (as a fraction of all datapoints) are listed.
Figure B.3: The distribution of the $y$-intercepts from the linear models of the core size as a function of the burial cutoff for biological interfaces from the BioMany dataset, shown in green, and crystal contacts from the XtalMany dataset, shown in red.

rooted in machine learning of higher complexity, these characteristics could achieve a predictive power higher than that of any individual one.

B.3 Conclusion

Although the histograms of the decay slopes and intercepts do not show a clear linear separability between groups, the results are still inconclusive. Some more experimentation, perhaps with a greater number of characteristics and by using better classifiers, should be able to improve the geometric classifier in EPPIC.
List of Figures

1.1 Composition of RNA and DNA ........................................... 2
1.2 Types of Protein–Protein Interactions ................................. 3
1.3 The Phase Problem .......................................................... 6
1.4 Crystallographic Symmetry ............................................... 7
1.5 The Interface-Classification Problem ................................. 8

2.1 Effects of Reduced Amino-Acid Alphabets .......................... 25

3.1 Published Reduced Amino-Acid Alphabets ........................... 29
3.2 ROC Curves for Published Alphabets ................................. 31
3.3 Random Two-Letter Amino-Acid Alphabets ......................... 33
3.4 ROC Curves for Random 2-Alphabets ................................ 34
3.5 Transitioning Two-Letter Amino-Acid Alphabets .................. 36
3.6 ROC Curves for Transitioning 2-Alphabets ......................... 37
3.7 Linear Model for Transitioning 2-Alphabets ......................... 39

4.1 ROC Curves for Surface-Residue Sampling ........................ 45

5.1 Hasse Diagram of Set Partitioning .................................... 49
5.2 Ten Best and Ten Worst Amino-Acid Duos .......................... 52

B.1 Distribution of Core Sizes ............................................... 62
B.2 Distribution of Decay Slopes ............................................. 63
B.3 Distribution of Decay Intercepts ....................................... 64
List of Tables

2.1 The Seven Amino-Acid Alphabets of EPPIC .......................... 17
2.2 The Confusion Matrix .................................................. 19
2.3 Top Parameters for Geometric Classifier .............................. 21
2.4 Top Low-Stringency Parameters for Core–Rim Evolutionary Classifier 22
2.5 Top High-Stringency Parameters for Core–Rim Evolutionary Classifier 22
2.6 Top Low-Stringency Parameters for Core–Surface Evolutionary Classifier 23
2.7 Top High-Stringency Parameters for Core–Surface Evolutionary Classifier 24
3.1 The Sixteen Published Amino-Acid Alphabets ......................... 28
3.2 AUROC Values for Published Alphabets ............................... 31
3.3 The Ten Random Two-Letter Amino-Acid Alphabets ................ 32
3.4 AUROC Values for Random 2-Alphabets .............................. 34
3.5 The Thirty-Two Transitioning Two-Letter Amino-Acid Alphabets .... 35
3.6 AUROC Values for Transitioning 2-Alphabets ....................... 37
4.1 Comparison of Surface-Sampling Strategies .......................... 42
5.1 Ten Best and Ten Worst 2-Alphabets ................................. 51
B.1 Separability by Geometric Criterion ................................. 63
Acknowledgements

First and foremost, I would like to thank Guido, for being such a caring and understanding supervisor, and José, for always helpfully answering any semblance of a question I had for him. Thanks also to Spencer, to Kumaran, to Aleix, and to all my other LBR colleagues, for making my stay at PSI an enjoyable time. Next, I would like to extend my thanks to Amedeo, for agreeing to be my internal supervisor; to my father, mother, and brother, for their unwavering support; and to Deena, for her constant words of encouragement and help with editing. Finally, I would like to thank Martin, Petra, Dania, and Aral, for always being around for those much-needed beers. Merci à tous.
References


[73] T. Piesk. File:Set partitions 4; Hasse; circles.svg — Wikimedia Commons, the free media repository, 2015. URL http://commons.wikimedia.org/w/index.php?
References


