

Building a disordered protein database: A case study in managing biological data

Arran D. Stewart

Xiuzhen Zhang

School of Computer Science and IT, RMIT University
Email: {arstewar, zhang}@cs.rmit.edu.au

Abstract

A huge diversity of biological databases is available via the Internet, but many of these databases have been developed in an ad hoc manner rather than in accordance with any data management principles. In addition, in the area of disordered protein databases, many of the databases have not been made publicly available. This poses challenges to researchers, since reliable protein databases are required in order to test and measure the accuracy of protein structure prediction software. In this paper, we describe our work developing a disordered protein database using data from the protein secondary structure database DSSPcont. In particular, we discuss the way in which we have addressed the issues of data cleaning, query processing and interoperability. This research is a pilot study in managing biological data.

Keywords: disordered proteins, biological data management

1 Introduction

The number of biological databases available via the Internet is large and constantly increasing. The number of publicly available databases that appear each year is so many that the Nucleic Acids Research Journal dedicates an entire issue to them annually (Srdanovic et al. 2005), and this listing does not include many more databases which are developed by researchers but not made publicly available.

These biological databases vary greatly in size and complexity (Luscombe et al. 2001). On the one hand one can find extremely large, “industrial scale” databases like the Protein Data Bank (PDB, <http://www.rcsb.org/pdb/>) (Berman et al. 2000), which stores data on the position of atoms within protein structures as determined by such techniques as X-ray crystallography or Nuclear Magnetic Resonance (NMR) spectroscopy. The PDB has a complicated, multi-tier infrastructure, an integrated workflow system, and dedicated staff to maintain and develop it. At the other extreme are small, temporary collections of flat files put together by individual bioinformatics researchers or small teams.

The multiplicity of biological databases has attracted considerable attention from members of the database research community. Some have called for more research into the data management issues of biological databases—for instance, Gupta (2004), Jagdish & Olken (2003) and Wooley & Lin (2005).

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at the Eighteenth Australasian Database Conference (ADC2007), Ballarat, Victoria, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 63. James Bailey and Alan Fekete, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

A number of important issues have been recognized, such as the complexity of queries and data models in biological databases (discussed further in section 2).

In this paper we report on how we have addressed data management issues in building a disordered protein database using data from the DSSPcont (<http://cubic.bioc.columbia.edu/services/DSSPcont/>) protein secondary structure database. We also suggest practical implementation tips for bioinformaticists working on structural biology database projects. We provide a more detailed explanation of what disordered protein segments are later in section 2.1. Roughly, however, they can be described as highly flexible regions of protein which do not adopt a regular, stable structure under normal physiological conditions. Our aim in the present research is to develop a reliable database of protein structural data which users can query in order to identify protein segments that meet particular criteria for being disordered.

In particular, we discuss the following issues:

- **Data cleaning and transformation.** High quality data is crucial to the success of any database. One important data quality issue is that of data semantics. For instance, the concept of solvent accessibility (SA) has been widely used to compute the hydrophobic contribution to the stability of proteins, and is used for defining protein disorder in our study. Raw SA scores derived from databases such as DSSPcont require normalization before they can be used as a reliable measure of protein flexibility. However, this normalization has been performed in a number of different ways in the biological literature. We propose normalization of SA according to the method of Ahmad et al. (2003) to avoid semantic disparity and ensure the reliability of our database.
- **Incremental query processing.** Queries run on biological databases are usually complex—the information needed is typically defined by compound criteria rather than a single condition, and are often performed on sequences or more complex data structures. In extracting disordered segments from DSSPcont, for instance, our criteria for identifying segments as disordered are based on a combined set of parameters. Furthermore, biologists often want to examine how changes in parameters affect the result data. We therefore design our system in such a way that queries are processed incrementally to improve efficiency.
- **Data interoperability and storage.** The text-based FASTA format is widely used for data exchange in bioinformatics. However interoperability considerations, and the ability to store metadata about sequences in a structured way,

make XML a preferable option. We discuss how data is organized in our system.

Our prototype disordered database can be used to compare the performance of different disorder prediction models. Since it represents a set of disordered segments collected according to a clearly-defined definition of disorder, by comparing the properties of our collection with others compiled according to different criteria, our database can also be used to examine the relationships between different types of disorder. The database is currently in use by staff at the Walter and Eliza Hall Institute of Medical Research (WEHI, <http://www.wehi.edu.au/>) for analyzing the properties of existing disordered proteins. It is intended that a Web-based interface be developed to make the database available for public use.

1.1 Related work

As noted previously, several studies have included general discussions of various data management issues for biological databases, and the need for increased effort in this area from the database community—for instance, Gupta (2004), Jagadish & Olken (2003) and Wooley & Lin (2005). However, general guidelines on how to resolve such issues have not yet been developed. Although not a general solution, our work is a pilot study in applying general data management principles to building a specific purpose database.

Bry & Kroger (2003) have examined in detail various biological databases and describe the development of data management technology in such databases. They noted that developers of biological databases have typically taken an ad hoc approach to data management issues.

Herbert et al. (2004) proposed a framework for managing and integrating evolutionary experimental data, and discussed data cleaning as a necessary step in data integration. They used a conceptual model to resolve mismatches and achieve uniformity among different terms. In contrast, our approach is to physically clean the data itself.

Much work on protein disorder has focused on developing models which can predict disorder from protein sequences (e.g., (Linding et al. 2003a), (Linding et al. 2003b), and (Cheng et al. 2005)). Building such models and evaluating their accuracy requires reliable training databases of known disorder in existing proteins. However, the training databases used in these research efforts are generally not intended for re-use and are not made publicly available. Data management issues are not discussed in any of these works.

DisProt (<http://www.disprot.org/>) (Vucetic et al. 2005) is the only publicly available disordered protein database. It contains records of protein segments that have been reported in the biological literature as showing disorder in experiments. Since different definitions of disorder have been adopted by experimenters, and since several distinct types of protein disorder are believed to exist, the use of DisProt's data is not suitable for studies that require a single, consistent definition of disorder. Additionally, the exact extent of a disordered segment in DisProt will have been determined by experimenters, and thus contains a subjective element. For our purposes, it is preferable that disordered segments be identified by objectively applied criteria. Data management issues are not discussed in this work.

2 Background

Although highly diverse, biological databases do have some similarities. For instance, those available via

the World Wide Web typically follow a 3-tier architectural model (Smail-Tabbone et al. 2005). At the bottom level is the actual database, at the top is a web interface, and in between is a software layer which mediates between the two. The middle layer turns requests made via the Web interface into actual database queries, and presents database results in HTML format.

Biological databases typically store data of a complex and often hierarchical nature (Brusic et al. 1998). The data may include sequences (for instance, DNA or protein sequences), graphs (for instance, metabolic pathways) or spatial information (for instance, 3D protein structures). It is possible to store such structures in a relational DBMS (RDBMS) by decomposing them into tables, but the result is often an overly complex relational schema (Bry & Kroger 2003). As a result, the use of an RDBMS has been described as being a “clumsy and awkward” way of manipulating complex biological data (Wooley & Lin 2005).

Consequently, biological databases have typically taken alternative approaches to data management. The earliest approach (and still a common one) was to simply store data as text-based flat files (Nelson et al. 2003). Some biological databases adopted a format known as ASN.1, originally developed for describing telecommunications protocols (Bry & Kroger 2003). More recent approaches have been the use of object-oriented or object-relational DBMSs (Jagadish & Olken 2003), or XML- or other semi-structured DBMSs (Shui et al. 2003).

The sorts of queries made on biological databases can typically be expressed succinctly in English, but lead to complex processing requirements. Singh (2003) gives several illustrative examples: “find all genes that are structurally similar to a given gene and express similarly over a specific DNA microarray dataset”; “find all proteins that are structurally similar to a given protein, used in a given pathway, and are expressed similarly as another given protein in a given experiment”; and “find all protein pairs that are less than 30% similar at a string level, share a given active site, and co-occur in some metabolic pathway”. All of these queries would lead to complex processing of non-atomic data types.

Our area of research—disordered protein databases—is a good example of the challenges faced. As will be seen in the next section, protein data is complex and hierarchical in nature, and querying it requires more complex processing than can be expressed in relational queries.

2.1 Proteins and their structure

Proteins are complex biological molecules made up of long sequences of small molecules—amino acids or residues—linked together in a chain. A single protein can contain a number of chains of amino acids (Branden & Tooze 1991); a protein will typically contain hundreds of amino acids, and some contain thousands.

Proteins have several levels of structure (Baxevanis & Ouellette 2005). At the level of *primary structure*, a protein chain can simply be viewed as a string of letters, with each letter representing one of 20 naturally-occurring amino acids (listed in Table 1).

For instance, the primary sequence of the human myoglobin protein, used to carry oxygen in the muscles, begins “GLSDGEWQLVLNVWGKVEA”.¹ However, each amino acid has chemical and physical properties which cause it to interact with nearby

¹This sequence was obtained from the SWISS-PROT database, at <http://ca.expasy.org/sprot/>, accession number P02144.

Amino acid	Abbreviation	Amino acid	Abbreviation
Alanine	A	Leucine	L
Arginine	R	Lysine	K
Asparagine	N	Methionine	M
Aspartic acid	D	Phenylalanine	F
Cysteine	C	Proline	P
Glutamic acid	E	Serine	S
Glutamine	Q	Threonine	T
Glycine	G	Tryptophan	W
Histidine	H	Tyrosine	Y
Isoleucine	I	Valine	V

Table 1: Naturally occurring amino acids. Source: Lesk (2002).

```

HEADER OXYGEN TRANSPORT          19-FEB-91      2MM1          2MM1  2
COMPND MYOGLOBIN MUTANT WITH LYS 45 REPLACED BY ARG AND CYS 110      2MM1  3
COMPND 2 REPLACED BY ALA (K45R, C110A MUTANT)                          2MM1  4
SOURCE HUMAN (HOMO $SAPIENS) RECOMBINANT FORM EXPRESSED IN          2MM1  5
SOURCE 2 (ESCHERICHIA $COLI)                                           2MM1  6
AUTHOR S.R.HUBBARD,W.A.HENDRICKSON,D.G.LAMBRIGHT,S.G.BOXER          2MM1  7
REVDAT 1 15-JAN-93 2MM1 0                                             2MM1  8
JRNL   AUTH S.R.HUBBARD,W.A.HENDRICKSON,D.G.LAMBRIGHT,S.G.BOXER      2MM1  9
JRNL   TITL X-RAY CRYSTAL STRUCTURE OF A RECOMBINANT HUMAN            2MM1 10
JRNL   TITL 2 MYOGLOBIN MUTANT AT 2.8 ANGSTROMS RESOLUTION           2MM1 11
JRNL   REF J.MOL.BIOL. V. 213 215 1990                                2MM1 12
JRNL   REFN ASTM JMOBAK UK ISSN 0022-2836 070                         2MM1 13
REMARK 1                                                                2MM1 14
REMARK 2                                                                2MM1 15
REMARK 2 RESOLUTION. 2.8 ANGSTROMS.                                   2MM1 16
REMARK 3                                                                2MM1 17
REMARK 3 REFINEMENT. BY THE RESTRAINED LEAST SQUARES PROCEDURE OF J. 2MM1 18
REMARK 3 KONNERT AND W. HENDRICKSON (PROGRAM *PROLSQ*. THE R          2MM1 19
REMARK 3 VALUE IS 0.158.                                             2MM1 20
REMARK 3                                                                2MM1 21
...
SEQRES 1 153 GLY LEU SER ASP GLY GLU TRP GLN LEU VAL LEU ASN VAL      2MM1 43
SEQRES 2 153 TRP GLY LYS VAL GLU ALA ASP ILE PRO GLY HIS GLY GLN      2MM1 44
SEQRES 3 153 GLU VAL LEU ILE ARG LEU PHE LYS GLY HIS PRO GLU THR      2MM1 45
...
ATOM   1 N   GLY   1      -5.817 17.320 15.842 1.00 18.38              2MM1 66
ATOM   2 CA  GLY   1      -4.704 17.705 14.942 1.00 17.81              2MM1 67
ATOM   3 C   GLY   1      -3.356 17.578 15.656 1.00 17.07              2MM1 68
ATOM   4 O   GLY   1      -3.081 16.578 16.353 1.00 17.43              2MM1 69
ATOM   5 N   LEU   2      -2.521 18.595 15.488 1.00 16.23              2MM1 70
ATOM   6 CA  LEU   2      -1.187 18.608 16.091 1.00 15.37              2MM1 71
ATOM   7 C   LEU   2      -1.322 18.457 17.619 1.00 14.87              2MM1 72

```

Figure 1: The PDB file format. Extracts from a sample PDB file for the human myoglobin protein.

amino acids, and result in protein chains folding into repetitive structures such as helices or sheets (Baxevanis & Ouellette 2005, Berg et al. 2002). These local, repetitive structures are known as the protein’s *secondary structure*.

The repetitive structures are a compact configuration for the protein chain, and this helps the protein achieve stability. The main forms of secondary structure are *alpha helices* and *beta strands* (which can also join together to form *beta sheets*). Some researchers consider alpha helices and beta strands/sheets to be the only “ordered” secondary structures (for instance, see (Uversky 2002)), whereas others include less common structures such as “3/10 helices” (for instance, (Linding et al. 2003a)). Helices other than alpha helices are rarely observed in proteins except at the ends of protein chains. Hence, we have adopted the approach of considering only alpha helices and beta strands/sheets as “ordered”.

The ordered secondary structures (helices and sheets) are typically linked to each other by unstructured lengths of protein that are typically classified as “coils”, “loops” and “turns”. In normal circumstances, the secondary structures of proteins further fold up into a complex three-dimensional structure, specific to each protein (Berg et al. 2002, Luscombe et al. 2001); this three-dimensional arrangements of the protein chains is known as the protein’s *tertiary*

structure (Branden & Tooze 1991). The final shape a protein assumes depends on the exact sequence of amino acids which compose it, as well as on the properties of its environment (for instance, a protein may fold differently depending on whether it is an acid or alkaline environment, or whether the ambient temperature is warm or cool) (Raven & Johnson 1989).

One of the central tenets of structural biology is that the function of a protein is determined by its three-dimensional structure. However, it has recently been recognized that unstructured or disordered regions of protein also play critical roles in protein function (Dunker et al. 2002, Dyson & Wright 2005, Linding et al. 2003a). Although no single consistent definition has yet been developed, loosely speaking, disordered protein segments are highly flexible regions of protein whose structures are difficult to investigate experimentally. Segments of protein which form stable three-dimensional structures are described as *ordered*, and those that do not as *disordered* (Cheng et al. 2005). It is also believed that there are several different types of protein disorder (Dunker et al. 2002, Linding et al. 2003a).

#	RESIDUE	AA	...	ACC	G	H	I	T	E	B	S	L	...
31	71	A	G	...	0	0	0	0	0	0	0	100	...
32	72	A	V	...	0	0	0	0	0	0	100	0	...
33	73	A	R	...	36	0	0	10	0	0	90	0	...
34	74	A	V	...	0	0	0	0	0	87	13	0	...
35	75	A	D	...	5	0	0	0	0	0	0	100	...
36	76	A	L	...	14	0	0	0	0	0	0	100	...
37	77	A	G	...	19	0	0	0	0	0	0	100	...
38	78	A	E	...	50	0	0	0	100	0	0	0	...

Figure 2: Extracts from a sample DSSPcont file.

2.2 The Protein Data Bank and derived databases

The main source of protein structural data is the PDB. The PDB contains data on over 32,000 proteins (over 21 gigabytes of data), and new proteins are added weekly. The primary format for exchange of PDB data is a text-based flat-file format. Each file contains data on the coordinates of the atoms in one protein, and details of the experimental procedures which created that data (Baxevanis & Ouellette 2005, Berman et al. 2000, Cohen 2004). The files consist of a *body*, containing the 3D coordinates of all atoms in the protein (usually about 2000 atoms per protein), and a *header* containing metadata about the protein and how the coordinates were derived. Figure 1 shows extracts from a PDB file.

The DSSP database² is derived from PDB data (Carter et al. 1983). It uses atomic coordinate data from PDB to derive a description of a protein’s secondary structure. Each amino acid in a protein is described as being part of various sorts of helix, sheet, or loop. A later database which extends the approach of DSSP is the DSSPcont database³ (Carter et al. 1983). Whereas DSSP assigns amino acids to a set of discrete structural categories (a process called “structural assignment”), DSSPcont performs *continuous* structural assignment—for each structural category, a *probability* is given that the amino acid falls into that category. DSSPcont is particularly useful in a study of disordered proteins, because the continuous assignment process captures flexibility in the positions of residues.

At the time of initial development of our database, the DSSPcont database contained 20,216 text-based flat files. Each file in the database records information on one protein; the files are named using the PDB accession number for the protein, with a “.dssp” extension. A single protein may consist of multiple protein chains, each identified by a single-letter abbreviation (“A”, “B”, and so on), and chains are separated by a delimiter record in which the amino acid type is marked as “!” (an exclamation mark). The total set of files is 1815 MB in size.

For each residue, DSSP and DSSPcont assign it to one of eight categories of structure, as shown in Table 2. Such a detailed classification is more than we need, however, since our aim is simply to divide residues into the two categories of “structured” or “unstructured”. For our purposes, only three of the categories are considered “structured”, namely alpha helices, beta strands, and beta sheets. All the other types of structure are considered “unstructured”.

²<http://swift.cmbi.ru.nl/gv/dssp/>

³Available at <http://cubic.bioc.columbia.edu/services/DSSPcont/>.

3 Extracting Disordered Protein Segments

In the present research, we aim to develop a database of protein structural data (ultimately derived from PDB data), and query it to locate disordered protein segments.

At the time of writing, we have developed a prototype which implements the bottom (database) level of the system, and processes queries made on the data. The top (presentation) level remains to be implemented.

As noted previously, it is generally accepted that a low degree of secondary structure combined with high flexibility implies disorder. Our query can be expressed as follows: given a threshold T_u of secondary-level unstructuredness, and a threshold T_f of flexibility, which residues in a set of protein chains exceeded both these thresholds? That is, which residues r have unstructuredness $r_u \geq T_u$ and flexibility $r_f \geq T_f$?

The DSSPcont database was selected as the most easily usable source of reliable structure and flexibility data. For any specified amino acid, the DSSPcont format provides us with information on its propensity for secondary structure or lack of structure, and a measurement of its SA.

3.1 Data cleaning and transformation: computing relative solvent accessibility

Solvent accessibility (SA) measures the proportion of an amino acid’s surface which is exposed to the solvent surrounding a protein (for instance, water). The interior of a protein is densely packed; thus, SA is low for amino acids in the interior of a protein, and high for those on the surface. SA is measured in units of square Ångströms.⁴ In general, the higher the SA of a region of residues, the more flexible it is.

DSSPcont provides raw SA measurements for amino acids. However, since amino acids vary in size, this figure is not comparable between different amino acids: a certain number of square Ångströms might constitute a very small proportion of a large amino acid, but a large proportion of a small one. We therefore normalize the SA of amino acids: the *relative* solvent accessibility (RSA) of an amino acid is the ratio of absolute solvent accessibility to the maximum observed accessibility for the amino acid type.

Different researchers have developed different methods of determining the maximum surface area, which give slightly different measurements (Richardson & Barlow 1999). Since we consistently adopt the values observed by Ahmad et al. 2003, all the RSA values we calculate are comparable. However, they are not comparable with RSA values obtained by different normalization methods. If one wants to compare RSA values between our data set and another which uses a different normalization method, RSA values for one or the other of the data sets would need to be recalculated.

⁴One Ångström equals 0.1 nanometres.

Structure	Label	Structure	Label
Alpha helix	A	Extended beta strand	E
3/10 helix	G	Turn	T
Pi helix	I	Bend	S
Beta bridge	B	Other/loop	L

Table 2: Structural categories used by DSSPcont.

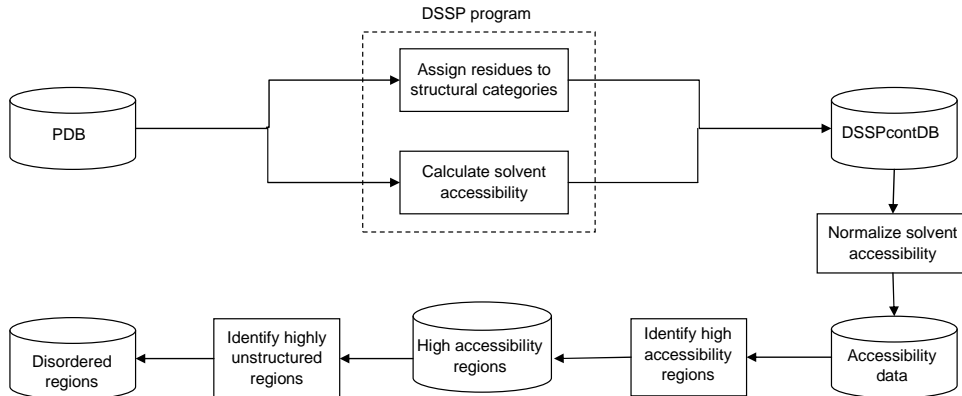


Figure 3: Processing of data

Although the files in the PDB are generally of high quality, some forms of “noise” and bad data do occur. One form of bad data we detect is residues that have been marked as “X” for “unknown” in the DSSPcont data files—is detected, and these residues are “sequestered”: they are not included in any statistical analyses of the data (but can be inspected separately). Other more subtle forms of noise can also occur, and represent a possibility for future work in this area.

Two conflicting requirements in relation to data noise are that although users typically want to remove or lessen the effects of noise, they often also want access to the original experimental data on which a database was based, and to have good records of the “provenance” (or history) of the data (Jagadish & Olken 2003). Our approach to resolving these conflicting requirements is to retain and make available the original DSSPcont data, but to allow “noise” to be screened out at the query stage.

3.2 Incremental query processing: creating a RSA repository

As discussed previously our queries to DSSPcont for disordered residues are defined by two criteria: (1) a high level of unstructuredness for a residue, and (2) a high level of RSA. The level of unstructuredness for a residue r is the probability that the residue is *not* part of an alpha helix or beta strand. By referring to Table 2, we can see that this is therefore the sum of the probabilities that the residue is part of a 3/10 helix (G), pi helix (I), turn (T), bend (S) and “other/loop” (L):

$$\text{unstructuredness} = P(G) + P(I) + P(T) + P(S) + P(L)$$

We wish to identify regions where the unstructuredness exceeds a threshold for unstructuredness α — $\text{unstructuredness} \geq \alpha$ —as well as having a high level of RSA, $\text{RSA}(r) \geq \beta$, where β is the threshold for RSA.

One simple way to identify disordered residues is to iterate over all the residues in the DSSPcont database, calculate the RSA and unstructuredness for each residue and identify which residues meet both criteria. A notable problem with the approach is the

large number of costly disk reads (recall that there are 20,216 files in the DSSPcont database). Thus, we only perform calculation of RSA once, creating an RSA repository with RSA values for the entire DSSPcont database.

On the other hand, to compare the disordered regions obtained by our approach with those by other approach such as DisProt, we need to run many disorder-extraction queries with different parameter settings for unstructuredness and flexibility. In order to amortize space and time costs over multiple queries, we therefore process queries incrementally—caching the result of the first step for use in later queries. For instance, one might wish to hold the RSA threshold constant at, say, 25%, and use 5 different parameters for the threshold of unstructuredness to see what effect this would have. Firstly, regions are identified which meet the 25% RSA threshold and the results are cached. Then, when looking for regions which also meet the unstructuredness criteria, we need only consider the high-RSA regions, rather than iterating over the whole DSSPcont database. Of course, it is also possible to detect regions in the opposite order—unstructured regions first, and then high-RSA regions within them—but for brevity, we restrict ourselves here to discussing the case in which we first identify high-RSA regions, and then disordered regions within those high-RSA regions.

3.3 The complete procedure

Our complete procedure for extracting disordered segments from DSSPcont is shown in Figure 3. The process of creating the disordered database is as follows:

1. Firstly, normalize absolute SA for all residues in the DSSPcont database and store these in a repository of RSA data. This repository is used in the next stage for identifying high-accessibility regions.
2. Identify high-RSA regions, by iterating over the DSSPcont database and referring to the repository of RSA data. The algorithm for identifying high-RSA regions is shown in Figure 4.
3. Iterate over the set of high-RSA regions and identify sub-regions of those regions which addition-

```

Input:

- A database RSADB of amino acid records
- A minimum RSA threshold  $\beta$
- A minimum length threshold  $T_{length}$

Output:

- A set HighRSA of regions of high solvent accessibility

Method:

1. for each protein chain C in RSADB:
2.   for each residue r in C:
3.     if  $RSA_r \geq \beta$  and StartPos has not been set:
4.       StartPos  $\leftarrow r$
5.     if  $RSA_r < \beta$  and StartPos has been set:
6.       EndPos  $\leftarrow r$
7.       length  $\leftarrow EndPos - StartPos + 1$
8.       if length  $> T_{length}$ :
9.         add  $\langle StartPos, EndPos \rangle$  to HighRSA

```

Figure 4: Algorithm: Locate regions of high relative solvent accessibility

```

Input:

- A set HighRSA of regions of high solvent accessibility
- A set DSSPcont of mappings from ResidueID to structural assignments
- A minimum unstructuredness threshold  $\alpha$
- A minimum length threshold  $T_{length}$

Output:

- A set DisorderedRegions of disordered regions


1. for each region R in HighRSA:
2.   for each residue r in R:
3.     From DSSPcont, look up  $P(G)_r$ ,  $P(I)_r$ ,  $P(T)_r$ ,  $P(S)_r$  and  $P(L)_r$
4.     Calculate the degree of unstructuredness of r:
       
$$unstructuredness_r \leftarrow P(G)_r + P(I)_r + P(T)_r + P(S)_r + P(L)_r$$
5.     if  $unstructuredness_r \geq \alpha$  and StartPos has not been set:
6.       StartPos  $\leftarrow r$
7.     if  $unstructuredness_r < \alpha$  and StartPos has been set:
8.       EndPos  $\leftarrow r$
9.       length  $\leftarrow EndPos - StartPos + 1$
10.      if length  $> T_{length}$ :
11.        Add  $\langle StartPos, EndPos \rangle$  to DisorderedRegions

```

Figure 5: Algorithm: Locate disordered regions

ally are highly unstructured (that is, have a low level of secondary structure). The algorithm for identifying disordered regions is shown in Figure 5.

In step 1, since all residues in the DSSPcont data set must be iterated over, the time taken to create the RSA repository will be proportional to the number of residues in the DSSPcont database, that is, the algorithm is of $O(n)$ time complexity. The space required is likewise proportional to the number of residues. Once RSA values have been calculated for all residues, we then have enough information to detect disordered regions. We also allow a user to specify a minimum length of disordered region they wish to detect. This can help save on disk usage. It is less likely that a very short region will provide statistically useful information; if users are only interested in, say, regions over 20 residues length, we can save space by ignoring all regions less than that length.

In step 2, to detect high-RSA regions, we iterate over the DSSPcont database, this time using our database of RSA values to look up the RSA for each residue. When the RSA rises above the threshold, we record this as being the possible start of a high-RSA region. When the RSA falls below the threshold again, we have reached the end of a possible region. If the length of the region exceeds the user-specified length threshold, we add it to the result set of high-RSA regions. In step 3, the algorithm for identifying

disordered regions iterates over all residues in the previously detected high-RSA regions, and similarly to the previous algorithm, flags any sub-regions which exceed the unstructuredness threshold. Both algorithms are straight forward, and are of $O(n)$ complexity, where n is the number of residues under consideration.

3.4 Implementation

The Perl and Python languages are the most frequently used in the bioinformatics field (Cohen 2004) and are therefore a natural choice for developing data processing routines for our database. We have made use of Python for our data processing routines, and some use of Perl for data analysis.

Parsers exist for the DSSP format, but none for the newer DSSPcont format. We therefore created a parser for the DSSPcont format by adapting algorithms used in the BioPython (Chapman & Chang 2000) (see <http://www.biopython.org/>) and BioPerl (Stajich et al. 2002) (see <http://www.bioperl.org/>) open source bioinformatics toolkits.

The DSSPcont format contains a range of header information (see Figure 2) not used in the present research application, which we therefore ignore. This includes fields such as “AUTHOR” (identifying the authors of the research which solved the structure of the relevant protein) and statistics of various sorts

```

>sp|P02144|MYG_HUMAN Myoglobin - Homo sapiens (Human).
GLSDGEWQLVLNVWGKVEADIPGHGQEVLRIRLFKGGHPETLEKFDKFKHLKSEDEMKASED
LKKHGATVLTALGGILKKKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLQSKHP
GDFGADAQGMNKALELFRKDMASNYKELGFQG

```

Figure 6: Example of FASTA file format. Source: SWISS-PROT database, at <http://ca.expasy.org/sprot/>.

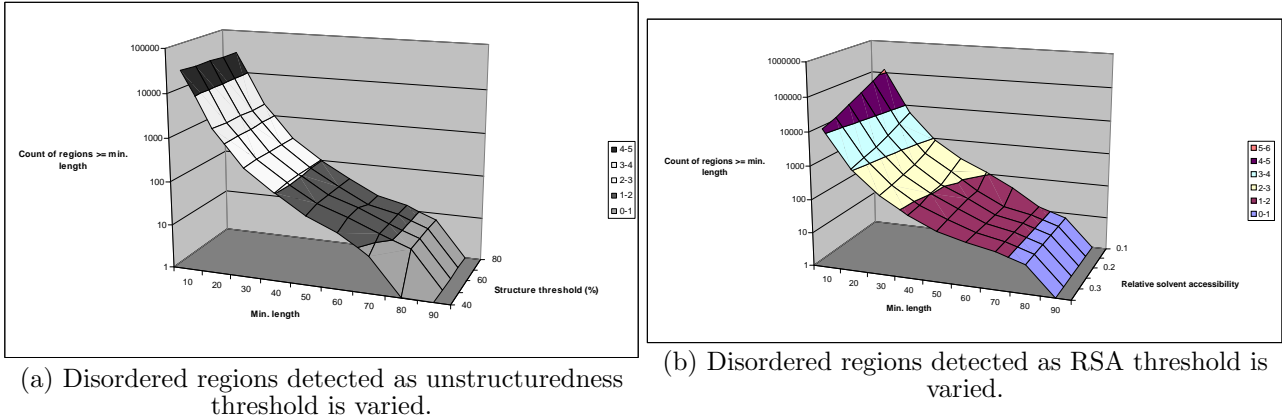


Figure 7: Disordered protein regions under different parameters

on the chemical bonds and structures within the protein (for instance, the total number of various types of hydrogen bonds). The data of interest to us is the residue records—one line of text per amino acid which list each residue in the protein, and give columns containing the structural data. Each line is 169 characters long, and data columns are identified by their position in the line for instance, the amino acid type of a residue is specified in the 13th character, and the SA in positions 34 to 37.⁵

There are some challenges in parsing DSSPcont data and integrating the results with other data sources. For instance, the ID numbers allocated to each residue in a protein do not correspond to those used by many other formats (although fortunately, DSSPcont also records PDB residue ID numbers). Additionally, the DSSPcont records contain discontinuities where the end of a protein chain is reached or where a region of missing structural data in the PDB is encountered, and these must be detected and handled by the parsing routine.

The output of normalizing solvent accessibility for all residues in the DSSPcont database (step 1 of the complete procedure for extracting disordered regions) is an index which can be used to look up the RSA for any specified residue in the DSSPcont data set, where a residue is specified by a protein ID, a chain ID and a residue position number. These identifiers are based on the PDB system—the protein ID is the PDB accession number, the chain ID is the PDB chain ID, and so on. As with any index, our index can be modelled as a set of mappings from input (in this case, a residue ID—a tuple $\langle ProteinID, ChainID, ResiduePosition \rangle$) to output (the RSA value). The function was implemented in Python; internally, the index is stored as a set of serialized Python data structures, since the code for reading and writing these to and from disk has been optimized to be very fast. The RSA data for each protein is serialized as a single file, within which the RSA for a specific amino acid can be looked up in constant time.

Both the algorithm for identifying high-RSA regions (Figure 4) and the algorithm for identifying disordered regions (Figure 5) output sets of regions.

Internally, our implementations of these algorithms store the sets as serialized Python data structures. For each protein, a serialized file contains a list of the detected regions for that protein. Fully specifying the location of a region requires specifying its start and end—thus, a region consists of a pair of residue IDs.

4 Managing the Disordered Protein Database

As discussed in Section 3, our disordered protein region extraction procedure are run with different unstructuredness and flexibility parameter settings so as to examine how these parameters affect the regions extracted and to verify the feasibility of our definition of disorder. The results are shown in Figure 7.

Figure 7(a) shows the effect on the number of regions found as we vary the level of unstructuredness required. It would appear that varying this threshold causes some variation, but has not nearly so pronounced an effect as does length. This seems to be because although DSSPcont uses a “continuous” method of assigning structure categories to residues, many residues show only either a very high chance of belonging to a category, or a very low one. That is, given a particular category (say, “alpha helix”), it appears that most residues are either given a high (often 90% to 100%) chance of belonging to that category, or a very low one (say, 0% to 10%), with few residues falling into intermediate probabilities.

Figure 7(b) shows the effect on the number of regions found as we vary the RSA threshold. For long regions (greater than about 65 residues in length), varying the adsAcRSA threshold does not seem to have a great impact on the number of regions detected, whereas for shorter regions, varying the adsAcRSA threshold from 0.4 to 0.1 can result in more than 10 times as many regions being detected. This suggests that where very long (> 65 residue length) regions occur at all, they have a very high adsAcRSA (> 0.4), whereas shorter regions have a lower and more variable adsAcRSA.

A question that is left to answer is how to store and manage the extracted disordered protein regions. This question is complicated by the complexity of our data. Considering the difficulty of applying generic Database Management Systems (DBMSs) and the

⁵<http://cubic.bioc.columbia.edu/services/DSSPcont/DSSPcont.html>.

characteristics that the data is used, we propose to manage the database with our custom data management tools. Specifically, data is stored internally as flat text files and output is in XML and text-based FASTA format for compatibility and interoperability reasons.

Generic DBMSs are in general not suitable for managing disordered segments. This is partly due to the fact that the data they contain is complex in structure. With relational DBMSs, representing the position and structural information of disordered segments is difficult and awkward. Alternatively, simply storing the data as unstructured binary or text-based fields (that is, BLOBs, or Binary Large Objects, and CLOBs, or Character Large Objects) means additional routines must be written for interpreting and using these fields.

Additionally, as with many biological databases, the patterns of access in our database differ significantly from most transactional databases. New data is required to be added at a reasonably slow rate (when compared with transactional systems), and once added, is only retrieved by users rather than edited. Only a small number of users require write access to the system. As a result, features of generic DBMSs such as concurrency control, transaction-based processing, and so on may not be needed (Han & Kamber 2001).

On the other hand, historically, many biological databases adopted the approach of storing their data as text-based flat files, and writing customized routines to parse and manage this data (Nelson et al. 2003). Because flat files are so widely used in the bioinformatics world, supporting them, at least as a format which can be exported, is important. Many bioinformatics analysis tools which operate on sequence data only accept data in the “FASTA” format. Figure 6 shows an example of the format, which is very simple. FASTA files begin with a “comment” line which starts with the “>” character, and are followed by 60-character-width lines containing the actual sequence. Thus, rather than using a generic DBMS, we use the operating system file system directly, and define our own routines for storing and managing data in FASTA format—effectively creating a custom DBMS. Due to the effort involved in implementing all the features (such as transactions, for instance) typically found in generic DBMSs, our system is far simpler.

XML (Bray et al. 2000) is becoming a widely used format for the storage and interchange of biological data (Srdanovic et al. 2005). Considering the interoperability of our data in future applications, we also support managing data in the XML format.

5 Conclusions

We have presented a prototype system for building a disordered protein database from PDB data using the derived secondary structure database DSSPcont. We have discussed the data management issues that arise in building the system, and in particular, we have proposed techniques for data cleaning, incremental query processing and data storage and analysis. We have also discussed implementation considerations on the management of protein sequence data. Work is underway to develop tools for analyzing the protein disorder databases and to make our prototype system publicly available.

Our system has demonstrated that our approach for building biological databases is feasible. Our system has also reflected on that managing biological data should follow the “business rules” in biology. The experimental nature of the biological rules leads

to the uncertainty in data and calls for specialized tools for managing different types of biological data. Indeed our work is an initial part of a bioinformatics project focusing on the study of disordered proteins. Our future work will focus on developing a suite of tools for querying and mining the disordered database.

Acknowledgements

The authors would like to thank Dr. Zhi-Ping Feng from The Walter and Eliza Hall Institute of Medical Research for helpful discussions.

References

- Ahmad, S., Gromiha, M. A. & Sarai, A. (2003), ‘Real value prediction of solvent accessibility from amino acid sequence’, *Proteins* **50**(4), 629–635.
- Baxevanis, A. D. & Ouellette, B. F. F. (2005), *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*, John Wiley & Sons, Hoboken, NJ.
- Berg, J. M., Tymoczko, J. L. & Stryer, L. (2002), *Biochemistry, Fifth Edition: International Version*, W. H. Freeman.
- Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shinalov, I. N. & Bourne, P. E. (2000), ‘The Protein Data Bank’, *Nucleic Acids Research* **28**, 235–242.
- Branden, C. & Tooze, J. (1991), *Introduction to Protein Structure*, Garland Publishing, NY.
- Bray, T., Paoli, J., Sperberg-McQueen, C. M. & Maler, E. (2000), *Extensible Markup Language (XML) 1.0 Second Edition W3C Recommendation*, Technical Report REC-xml-2001006, World Wide Web Consortium.
- Brusic, V., Wilkins, J. S., Stanyon, C. A. & Zeleznikow, J. (1998), Data learning: Understanding biological data, in G. Merrill & D. K. Pathak, eds, ‘Knowledge Sharing Across Biological and Medical Knowledge Based Systems: Papers from the 1998 AAAI Workshop’, AAAI Press, pp. 12–19.
- Bry, F. & Kroger, P. (2003), ‘A computational biology database digest: Data, data analysis, and data management’, *Distributed and Parallel Databases* **13**(1), 7–42.
- Carter, P., Andersen, C. A. F. & Rost, B. (1983), ‘DSSPcont: continuous secondary structure assignments for proteins’, *Nucleic Acids Research* **31**(13), 3293–3295.
- Chapman, B. & Chang, J. (2000), ‘Biopython: Python tools for computational biology’, *SIGBIO Newsletter* **20**(2), 15–19.
- Cheng, J., Sweredoski, M. & Baldi, P. (2005), ‘Accurate prediction of protein disordered regions by mining protein structure data’, *Data Mining and Knowledge Discovery* **11**(3), 213–222.
- Cohen, J. (2004), ‘Bioinformatics—An introduction for computer scientists’, *ACM Computing Surveys* **36**(2).
- Dunker, A. K., Brown, C. J., Lawson, J. D., Iakoucheva, L. M. & Obradovic, Z. (2002), ‘Intrinsic disorder and protein function’, *Biochemistry* **41**(21), 6573–6582.

- Dyson, H. J. & Wright, P. E. (2005), 'Intrinsically unstructured proteins and their functions', *Nature Reviews* **6**, 197–208.
- Gupta, A. (2004), 'Life science research and data management', *SIGMOD Record* **33**(2), 12–14.
- Han, J. & Kamber, M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, CA.
- Herbert, K. G., Gehani, N. H. & Piel, W. H. (2004), 'BIO-AJAX: an extensible framework for biological data cleaning', **33**(2), 51–57.
- Jagadish, H. V. & Olken, F. (2003), 'Database management for life science research: Summary report of the workshop on data management for molecular and cell biology at the National Library of Medicine, Bethesda, Maryland, February 2–3, 2003', *OMICS* **7**(1), 131–137.
- Lesk, A. M. (2002), *Bioinformatics*, Oxford University Press, Oxford, UK.
- Linding, R., Jensen, L. J., Diella, F., Bork, P., Gibson, T. J. & Russell, R. R. (2003a), 'Protein disorder prediction: implications for structural proteomics', *Structure* **11**(11), 1453–1459.
- Linding, R., Russell, R. B., Neduva, V. & Gibson, T. J. (2003b), 'GlobPlot: Exploring protein sequences for globularity and disorder', *Nucleic Acids Research* **31**(13), 3701–3708.
- Luscombe, N. M., Greenbaum, D. & Gerstein, M. (2001), 'What is bioinformatics? A proposed definition and overview of the field', *Methods of Information in Medicine* **40**, 346–358.
- Nelson, M. R., Reisinger, S. J. & Henry, S. G. (2003), 'Designing databases to store biological information', *BIOSILICO* **1**(4), 134–142.
- Raven, P. H. & Johnson, G. B. (1989), *Biology*, Times Mirror/Mosby College Publishing, St Louis, Missouri.
- Richardson, C. J. & Barlow, D. J. (1999), 'The bottom line for prediction of residue solvent accessibility', *Protein Engineering* **12**(12), 1051–1054.
- Shui, W. M., Wong, R. K., Graham, S. C., Lee, L. K. & Church, W. B. (2003), A new approach to protein structure and function analysis using semi-structured databases, in Y.-P. P. Chen, ed., 'First Asia-Pacific Bioinformatics Conference (APBC2003)', Vol. 19 of *Conferences in Research and Practice in Information Technology*, Australian Computer Society, Inc., Adelaide, Australia, p. ???
- Singh, A. K. (2003), 'Querying and mining biological databases', *OMICS: A Journal of Integrative Biology* **7**(1), 7–8.
- Smail-Tabbone, M., Osman, S., Messai, N., Napoli, A. & Devignes, M.-D. (2005), Bioregistry: A structured metadata repository for bioinformatic databases, in 'CompLife 2005 (First International Symposium on Computational Life Science, Konstanz, Germany, September 25–27, 2005)', Vol. 3695, Springer, Berlin, pp. 46–56.
- Srdanovic, M., Schenk, U., Schwieger, M. & Campagne, F. (2005), 'Critical evaluation of the JDO API for the persistence and portability requirements of complex biological databases', *BMC Bioinformatics* **6**(1), 5.
- Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C. J., Osborne, B. I., Pocock, M. R., Schattner, P., Senger, M., Stein, L. D., Stupka, E., Wilkinson, M. D. & Birney, E. (2002), 'The Bioperl toolkit: Perl modules for the life sciences', *Genome Research* **12**(10), 1611–1618.
- Uversky, V. N. (2002), 'What does it mean to be natively unfolded?', *European Journal of Biochemistry* **269**, 2–12.
- Vucetic, S., Obradovic, Z., Vacic, V., Radivojac, P., Peng, K., Iakoucheva, L. M., Cortese, M. S., Lawson, J. D., Brown, C. J., Sikes, J. G., Newton, C. D. & Dunker, A. K. (2005), 'DisProt: A database of protein disorder', *Bioinformatics* **21**(1), 137–140.
- Wooley, J. & Lin, H., eds (2005), *Catalyzing Inquiry at the Interface of Computing and Biology*, The National Academies Press, Washington.