J Math Model Algor (2009) 8:103–123
DOI 10.1007/s10852-009-9107-3

An Artificial Immune System for Clustering Amino Acids in the Context of Protein Function Classification

A. Secker · M. N. Davies · A. A. Freitas · J. Timmis · E. Clark · D. R. Flower

Received: 15 October 2008 / Accepted: 20 January 2009 / Published online: 25 February 2009

Abstract This paper addresses the classification task of data mining (a form of supervised learning) in the context of an important bioinformatics problem, namely the prediction of protein functions. This problem is cast as a hierarchical classification problem. The protein functions to be predicted correspond to classes that are arranged in a hierarchical structure (this takes the form of a class tree). The main contribution of this paper is to propose a new Artificial Immune System that creates a new representation for proteins, in order to maximize the predictive accuracy of a hierarchical classification algorithm applied to the corresponding protein function prediction problem.

Keywords Artificial immune systems · Optimisation · Bioinformatics · Classification · Clustering

Mathematics Subject Classifications (2000) 68T20 · 68W05

1 Introduction

This paper addresses classification within data mining in the context of bioinformatics, more precisely the prediction of protein function. In essence, a protein consists of a linear sequence of amino acids, and predicting the function of a

A. Secker · A. A. Freitas (✉)
Computing Laboratory and Centre for BioMedical Informatics,
University of Kent, Canterbury, CT2 7NF, UK
e-mail: a.a.freitas@kent.ac.uk

M. N. Davies · D. R. Flower
The Jenner Institute, University of Oxford, Compton, Newbury, Berkshire, RG20 7NN, UK

J. Timmis · E. Clark
Departments of Computer Science and Electronics, University of York, York, YO10 5DD, UK

protein, based on information derived from its sequence of amino acids, remains a subject of interest in bioinformatics. The problem is important because, although the structure and function of a protein can in theory be determined by experimental biological techniques, such techniques are in general very time consuming and expensive. For instance, crystallography can be used to determine a protein's three-dimensional structure, which greatly helps in determining its function, but this is a very slow technique (some proteins take months or even a year to be crystallized), and some proteins cannot be crystallized at all. This is particularly the case for transmembrane proteins. Computational prediction of protein function is in general less reliable than a biological experimentation, of course, but it is much faster and cheaper, and so it can provide useful clues for biologists to perform slow, expensive research in a more selective manner (investigating potential functions suggested by computational predictions, instead of investigating a much larger number of potential functions).

The main contribution of this paper is to propose a new Artificial Immune System (AIS) – a variant of opt-aiNet (a well-known AIS) – that creates a new representation for proteins, in order to maximize the predictive accuracy of a classification algorithm applied to the corresponding protein function prediction problem.

In this context, this paper proposes an AIS that evolves clusters of amino acids optimized for a given type of protein. The evolved clusters are then used to define the protein representation that will be used by the classification algorithm. In the words of machine learning and data mining, the AIS algorithm solves a clustering (unsupervised learning) problem, consisting of finding the optimal clustering of amino acids for the type of protein whose data is being mined, and the result of the AIS is then used to solve a classification (supervised learning problem).

The AIS proposed in this paper differs from the original opt-aiNet in three significant ways, namely: (a) it uses a novel immune cell representation suitable for the target problem; (b) it uses a different type of mutation, again suitable for the target problem; (c) and it uses a new fitness function, which is very different from and much more complex than the one used by opt-aiNet.

The proposed AIS is evaluated on a challenging real-world protein function prediction problem: the classification of GPCRs (G-protein-coupled receptors) into their functional classes. GPCRs constitute a large and diverse group of proteins that perform many important physiological functions [4–6]. The GPCR classification problem which is addressed is especially challenging because it involves a large number of classes organized in a hierarchy—being an instance of the so-called hierarchical classification problem—as will be explained later.

The remainder of this paper is organized as follows. Section 2 describes how the problem of predicting GPCR functions is cast into a classification problem. This section also provides some background on bioinformatics, in order to make the paper more understandable to readers without a biology background. Section 3 described the proposed AIS for clustering amino acids. Section 4 reports computational results, and Section 5 concludes the paper.

2 Casting Protein Function Prediction as a Classification Problem in Machine Learning/Data Mining

2.1 G-Protein Coupled Receptor (GPCR) Classification

Proteins are large molecules comprised of a long chain of amino acids that perform a wide range of vital functions in living organisms. A protein consists of a linear sequence of amino acids—each of which can be represented by a single letter. For instance, the sub-sequence "AVC..." corresponds to (A)lanine, (V)aline, (C)ysteine. There are 20 standard amino acids, each represented as a different letter and the order with which these amino acids are chained together is known as the protein's primary sequence. These long chains fold into complex structures allowing them to perform functions. In its folded shape the protein may present a certain conformation on its surface that may allow it to bind to another protein or catalyse a particular reaction.

GPCR proteins are transmembrane proteins, meaning that they fold themselves such that some parts of the protein are found inside a cell, while other parts are external. GPCRs are bound by a variety of different molecules (ligands) found outside the cell. The binding activates the GPCR, which in turn binds a G-protein inside the cell, often changing the behaviour of that cell. The GPCR superfamily is a large and diverse multigene superfamily of integral membrane proteins that perform many important physiological functions [4–6]. GPCRs control and/or affect physiological processes as diverse as neurotransmission, cellular metabolism, secretion, cellular differentiation and inflammatory responses [7]. They also affect taste, smell and vision. Malfunctions in GPCR expression can result in ulcers, allergies, carcinomas and migraine attacks. Mutations in GPCR-coding genes have been linked to over 30 human diseases including retinitis pigmentosa, hypo- and hyperthyroidism, nephrogenic diabetes insipidus as well as several fertility disorders [8]. The receptors binding endogenous peptides have an import role in mediating the effects of a wide variety of neurotransmitters, hormones and paracrine signals. The receptors that bind biogenic amines, e.g. norepinephrine, dopamine, and serotonin, are very commonly modulated by drugs. Pathological conditions, including Parkinson's disease, schizophrenia, drug addiction, and mood disorders are examples of where imbalances in the levels of biogenic amines cause altered brain functions. Indeed, GPCR proteins are a common target for therapeutic drugs and approximately 50% of all marketed drugs are targeted towards one of the receptors [9].

The most widely used classification for the superfamily was introduced in the first version of the GPCR database (GPCRdb) [10]. The classification system divides the GPCRs into six families, designated A–F. These were derived from the use of characteristic "fingerprints", unique compositions of similar regions when a primary sequence is compared with another [11, 12]. The classes can further be divided into sub-families and sub-sub families based upon the function of the GPCR protein and the specific ligand that it binds.

2.2 Representing Proteins by Local Descriptors of Amino Acid Sequences

Given a protein's sequence of amino acids, one can try to determine its function via either biological experiments or computational prediction methods. The former produce in general more precise results, but are much more time consuming and expensive. Hence, the latter is often used in practice, and it can provide valuable information for the more cost-effective use of biological experiments. This work addresses the computational prediction of protein function, by casting this problem as a classification (supervised learning) problem in machine learning/data mining, where protein functions are classes and attributes derived from the protein's sequence of amino acids are the predictor attributes.

The length of the primary sequence varies widely across different proteins. Since the vast majority of classification algorithms can cope only with datasets where all examples (records or data items) have the same length, it is necessary to convert all proteins (examples) to the same fixed number of attributes, using an attribute representation at higher level of abstraction than the full sequence of amino acids.

This paper is concerned with finding an optimum representation for a protein. In our experience with classifying GPCRs, a number of different representations have been used. For example, previously, we have published work which uses five "z-values" to represent a protein sequence [13, 14]. In this case, 26 separate physicochemical properties are reduced to five empirical 'z' values [15, 16]. One set of z-values has been determined for each of the 20 amino acids. Each amino acid in the sequence under scrutiny can then be transformed into its respective set of z-values. Once this has happened for every protein in the sequence, it was found that taking the mean of each z-value over the sequence resulted in a representation that was very competitive in terms of accuracy. As the mean is taken over each z-value individually, the entire protein sequence (usually many hundreds of amino acids) is reduced to just five attributes. More recent work, however, has shown that the use of local descriptors can produce superior results in the shape of higher classification accuracy.

2.3 Creating Attributes Using Local Descriptors

Local descriptors [1, 2] are a commonly used method to classify protein families. The high-level representation used here involves the attribute creation technique defined in [2], which is based on summarising the protein's entire sequence of amino acids by a fixed number of local descriptors (attributes). As there is greater structural than sequential homology with the GPCR superfamily, alignment-free approaches have been more effective at classification than techniques based upon sequence similarity [13, 17].

A method of creating a fixed-length representation for a protein using local descriptors is described below. Cui et al. [1] divided the 20 amino acids into three functional clusters: hydrophobic (amino acids C, V, L, I, M, F, W), neutral (amino acids G, A, S, T, P, H, Y), and polar (amino acids R, K, E, D, Q, N), as suggested by Chothia and Finkelstein [3]. (The term "hydrophobic" means "with fear of water", so a hydrophobic amino acid tends to be buried within a protein, rather than in its surface exposed to water; the term "polar" has the opposite meaning.) It is then possible to substitute the amino acids in the sequence for the cluster in which

that amino acid belongs. Assuming H = hydrophobic, N = neutral and P = polar, the protein sequence CVGRK would be converted to HHNPP. The position or variation of these identifiers within a sequence is the basis of three local descriptors: composition (C), transition (T), and distribution (D) as follows:

- C is the proportion of amino acids with a particular property (drawn from a particular cluster such as the hydrophobic one). As an example, given the cluster H, we can determine C(H) over the example sequence of HHNPP as 0.4 as two of five positions in the sequence are of value H.
- T is the frequency with which amino acids with one property are followed by amino acids with a different property. Thus to compute T(N) over the example sequence, we can see there is a transition between H and N from positions two to three, then a transition from N to P between positions three and four. In this case T(N) = 2/4 = 0.5 as there are four places where a transition may occur. Any transitions between H and P are ignored here as neither of these clusters are the subject.
- D measures the chain length within which the first, 25%, 50%, 75% and 100% occurrences of the particular property are located.

Given that the amino acids are divided into three clusters in this instance, the calculation of the C, T and D descriptors generates 21 attributes in total (three for C, three for T and 15 for D). While this technique is valid if applied over the whole amino acid sequence, Tong et al. [2] split the amino acid sequences into ten overlapping regions (Fig. 1), hypothesising that these regions will capture the different characteristics of the different places along the protein. For sequences A–D and E–F there may be cases where the sequence cannot be divided so that all elements have the same length. In this case each subsequence may be extended by one residue if required. Each descriptor—C, T, and D—is calculated over the ten subsequences.

The number of attributes created with this technique therefore generalises to $70n$, where n is the number of amino acid clusters. In the case of the three clusters of amino acids used by Cui et al., proteins are now represented by 210 numerical

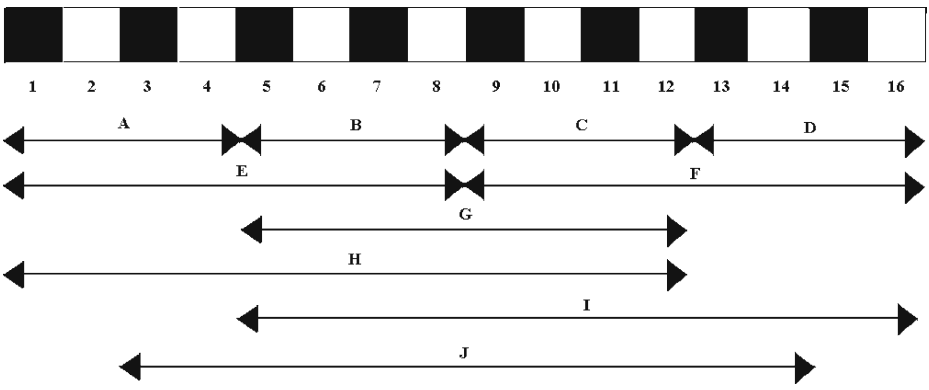

Fig. 1 The ten descriptor regions (*A*–*J*) for a hypothetical protein sequence of 16 amino acids. Adapted from Tong et al. [2]

descriptors, which can be offered to any of the plethora of well understood, well documented classification algorithms.

2.4 Motivation for Optimisation

In the case of [2], the three clusters as defined in [1] were used, no explicit explanation was included to justify the use of this particular clustering scheme. While there exists a truly enormous number of ways to partition the 20 amino acids, it seems clear that some will be more useful than others. The effectiveness of the techniques using local descriptors is dependent largely on the underlying amino acid grouping chosen as the basis of the descriptors. The accuracy of the technique can therefore be improved by using the "correct" grouping. However, in general it is not possible to determine, a priori, which amino acid clustering will result in the optimal performance for a given protein dataset. Sensibly, a biologically grounded grouping scheme would be used over one created at random. As an illustration, Table 1 shows a selection of candidate groupings identified in previous work [18]. The interested reader is directed towards this reference for a short literature review of papers that suggest groupings along with a great deal more information about the biological basis of such groupings. Each of the groupings in this table is as valid as any other as there is no way to tell *a priori* which group will yield the optimal accuracy when the prediction task with representation based on the generated attributes is attempted.

While a grounded grouping is likely to do well, its performance is unlikely to be optimal. There are numerous other variables that will have bearing on the final performance of the representation. For example, the best set of groupings may not be just hydrophobicity for example, it may be a combination of hydrophobicity and

Table 1 Candidate sets of amino acid groupings from Li *et al.* 2003 [19] and Cannata *et al.* 2002 [20] taken from [18]

Grouping	No. of groups	Reference
CMFILVWY AGTSNQDEHRKP	2	Li *et al.* [19]
CMFILVWY AGTSP NQDEHRK	3	Li *et al.* [19]
CMFWY ILV AGTS NQDEHRKP	4	Li *et al.* [19]
FWYH MILV CATSP G NQDERK	5	Li *et al.* [19]
FWYH MILV CATS P G NQDERK	6	Li *et al.* [19]
CFYWMLIV GPATSNHQEDRK	2	Li *et al.* [19]
CFYWMLIV GPATS NHQEDRK	3	Li *et al.* [19]
CFYW MLIV GPATS NHQEDRK	4	Li *et al.* [19]
CFYW MLIV G PATS NHQEDRK	5	Li *et al.* [19]
CFYW MLIV G P ATS NHQEDRK	6	Li *et al.* [19]
ARNDCQEGHKPST ILMFWVY	2	Cannata *et al.* [20]
ARNDQEGHKPST C ILMFWVY	3	Cannata *et al.* [20]
ARNDQEGHKPST C ILMFVY W	4	Cannata *et al.* [20]
AGPST RNDQEHK C ILMFVY	4	Cannata *et al.* [20]
AGPST RNDQEK H C ILMFVY W	6	Cannata *et al.* [20]
A RK N D C Q E G H IV LM F P ST W Y	16	Cannata *et al.* [20]
AS RK N D C Q E G H IV LM F P T W Y	16	Cannata *et al.* [20]
A R K N D C Q E G H IV LM FY P ST W	16	Cannata *et al.* [20]
AST R K N D C Q E G H IV LM F P W Y	16	Cannata *et al.* [20]
A R K N D C QE G H IV LM F P ST W Y	16	Cannata *et al.* [20]

size. Likewise there may be a combination of more than two physical chemical or electrical properties that could combine to create the best sets of groupings for a particular application. However, trying to determine accurate groupings of amino acids is extremely difficult due to the large number of possible groupings of 20 objects. The number of permutations may be determined by the Stirling number where S(n,k) represents the number of ways to partition a set on n object into k groups. If $k = 2$ or $k = 5$ then the number of possible permutations are 524,287 and 749,206,090,500 respectively [21]. It is therefore intractable to evaluate every single potential grouping.

In addition, the classifier used may have certain biases that can be exploited during the clustering procedure. For example some representations define 16 different groupings [20], which would result in 1,120 attributes being created. It is possible that a particular classifier does not cope well with a large number of attributes and as such this particular grouping may be incompatible with the, presumably otherwise good, classification algorithm. The keys to the success or failure of the technique described thus far are:

1. The number of clusters used, and
2. The specific amino acids that are included in each cluster.

Hence, in principle we can use a data-driven approach to evolve an amino acid clustering that approaches optimality with respect to both the data being mined and the classification algorithm applied to that data. The advantage of using such an optimiser is that a classifier may be used to gauge the quality of a solution or solutions at each stage in the optimisation process. Thus the optimisation of the representation (groupings) is guided by the output of the classification algorithm that will be used in the final testing stage. Therefore, the representation is able to exploit any bias in that classifier to improve prediction quality. The system described in this paper will, therefore, optimise amino acid groupings in a data driven manner such that classification accuracy is maximised when the groups are used to create predictor attributes and the resultant data representation is tested using a classifier.

2.5 Hierarchical Classification of G-Protein-Coupled Receptors (GPCRs)

The method of optimising clusters for a local descriptor-based attribute construction technique, as proposed in this paper, is generic to any protein dataset where it is sensible to represent the data using the local descriptors representation, but it should be pointed out that the GPCR dataset used in this study is hierarchical in nature. As described in Section 2.1. While the task of this paper is to consider the optimisation of a protein representation, this representation must be assessed using a classifier. As such it is worth briefly outlining the particularities of hierarchical data and hierarchical classification.

Some data can be naturally organised as a hierarchy of classes. The classification of data in such a hierarchy poses some unique challenges to data miners, such as the large number of classes to be predicted. Most classifiers deal with flat data sets, i.e., data for which a single level of classes may be assigned to an example. In a hierarchical dataset an example may be assigned to one class at a number of levels of specialisation. The most general level being near the root of the tree and becoming more specialised as the tree's branches are traversed. In the hierarchy of GPCR

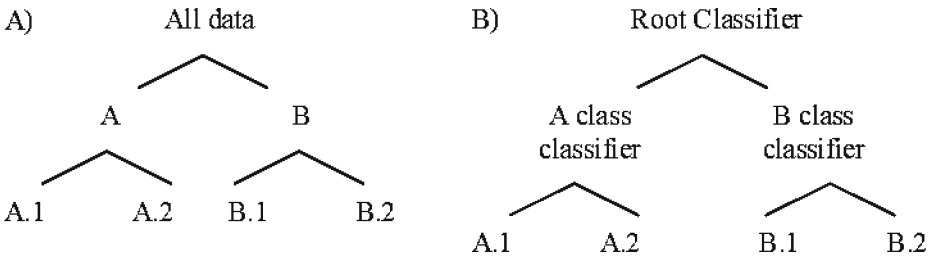

Fig. 2 Example of a hierarchical dataset (**A**) and how that hierarchy may be reflected in a tree of classifiers (**B**) ready for a top-down approach to classification

proteins, each class has exactly one parent—i.e. the data is structured like a tree. The class structure of a typical flat dataset will contain, for example, classes A, B and C which are all equally different from each other. However, in a hierarchy some classes may be more alike than others. Given the class tree in Fig. 2a, Classes A and B are equally dissimilar, but these classes may subdivide such that classes A.1 and A.2 are more alike than A.1 and B.1 as A.1 and A.2 share a common parent class.

There are a range of strategies available for predicting hierarchical classes [22]. The simplest is to flatten the dataset to one single level so that no superclasses or subclasses are present, then use one of the standard classification algorithms to predict the class. However, this strategy has two major drawbacks.

1. It does not take advantage of the information implicit in the class structure.
2. It will often require a single classifier to distinguish between a huge number of classes.

The top-down approach to classification on the other hand tackles both of these issues by constructing a tree of classifiers, thus converting the hierarchical classification into a number of flat classification problems that may be solved independently by running a flat classifier for each. The class structure is reflected by the structure of the classifiers while each classifier needs only to distinguish between a vastly reduced number of classes. Further explanation is reserved until Section 4.2.

3 The Proposed Artificial Immune System for Amino Acid Clustering

Pseudocode 1 shows the most general view of the process of attribute creation based on amino acid clustering (performed by an AIS) and subsequent use of a classification algorithm. Note that this process of attribute creation (or construction) based on clustering should not be confused with attribute selection. The goal of attribute selection is essentially to choose a subset of relevant attributes, out of all available attributes. This work rather involves attribute construction, where the goal is to create new attributes (new descriptors of amino acid sequences corresponding to higher-level information about proteins) based on the original sequence of amino acids (corresponding to lower-level information about proteins). The actual process of attribute creation is performed by using a clustering algorithm that groups together similar amino acids, and the result of this clustering is then used to produce a new set of predictor attributes for the classification algorithm.

Pseudocode 1 High level description of amino acid clustering-based attribute creation and subsequent use of classification algorithm

```
1   Split full dataset into training and testing sets
2   Split training set into sub-training and validation sets
3   Generate initial random candidate clustering solutions
4   Evolve clustering
    4a  Create attributes for sub-training and validation set from clusters
    4b  Train classifier on sub-training set
    4c  Evaluate classifier on validation set
    4d  Assign quality to this clustering
    4e  Update population depending on individual's quality
    4f  Repeat from 4 until stopping criterion is met
5   Return the best clustering from the population
6   Create attributes for training and testing datasets using this best clustering
7   Train classifier using newly transformed training set
8   Evaluate classifier using newly transformed test set.
```

In Pseudocode 1, points 1 and 2 are standard pre-processing tasks. Point 3 initialises the population for the AIS that performs amino acid clustering; while point 4 and sub-points thereof describe, at a high level of abstraction, the evolutionary process of amino acid clustering. Point 6 uses the output of the AIS (point 5) to create the data which will form the input to the classification algorithm, while points 7 and 8 are the standard training/testing steps used in a classification scenario.

3.1 The Optimiser: opt-aiNet

The optimiser used for generating the groupings is opt-aiNet, which is an optimiser based on principles and processes of the vertebrate immune system. opt-aiNet uses ideas of clonal selection and somatic hypermutation as found in natural immune systems and abstracts them for use as computer algorithms. Therefore, just as the immune system may be seen to maximise the efficiency of its reaction against a pathogenic stimulus, so too an optimiser based on the same principles may maximise a given criterion to solve a particular problem. opt-aiNet is, therefore, one of a class of computer algorithms called Artificial Immune Systems (AIS) [23]. AIS algorithms are typically population based algorithms that employ observed and theoretical immune functions and principles as inspiration. They tend to be (a) population based such that every individual in the population encodes a potential solution to a problem, and (b) evolutionary in nature [24] such that pressure is applied to the population as a whole, which has the effect, over many generations of individuals, of improving the population on average against some criteria. The proposed AIS for amino acid clustering is a new variant of opt-aiNet, which we call opt-aiNet-AA-Clust (opt-aiNet for Amino Acid Clustering).

Opt-aiNet was first proposed in [25], and it was originally proposed as a function optimisation tool. In this case, each immune cell would encode a single floating point value—the input to the function to be optimised.

Opt-aiNet encodes potential solutions to the optimisation problem as a network of immune cells. Each cell will start off in a random configuration, thus encoding a random point in the solution space. During successive generations, each individual will be assigned a measure of quality based on the output of a fitness function. This

function will take the point in the solution space encoded by an individual and return the quality of that point. The algorithm then uses the immune processes of somatic hypermutation and clonal expansion to impart dynamics on the population. All members of the population will, at this time, create clones, but the number of clones produced will vary in proportion to the measured fitness of the parent. This has the effect of, over time, increasing the average quality of the population compared with this objective fitness function. Finally, two other mechanisms are called upon at each iteration, these are network suppression and the introduction of randomly generated cells. The former will ensure a population that represents the solution efficiently as cells encoding the same or similar areas of space will be removed, while the latter will ensue the population cannot get stuck on a local optimum as it will maintain diversity in the population.

Several modifications were required to allow the opt-aiNet algorithm to work in our scenario of amino acid clustering. These included the changing of the individual representation from a real value to a string of symbols to represent clusters, the changing of the mutation procedure (to mutate a symbol rather than a real-valued number), the changing of the fitness evaluation from a straightforward mathematical function to a much more complex system for creating and evaluating the attributes produced by the clustering results and some minor procedural changes such as the termination function. In the case of the original opt-aiNet, the algorithm will terminate when there has been no improvement above a threshold in the population between successive iterations. In this case, it is possible that many iterations could pass before an improvement is found and thus the system terminates after a given number of iterations. These changes are explained in more detail below.

3.1.1 Individual (Immune Cell) Representation

The algorithm uses the clonal selection principle found in immunology to drive the optimisation process. The basis of the algorithm is a population of individuals which each encode a potential solution to the problem; in this case therefore each individual encodes a grouping scheme for the 20 amino acids. Each individual (immune cell) encodes a candidate solution to the problem of clustering the 20 amino acids. More precisely, each individual consists of a vector with 20 elements, $< c_1, \ldots, c_{20} >$, where the ith element, $c_i = 1, \ldots, 20$, indicates the id of the cluster to which the ith amino acid is assigned since there are 20 amino acids. To consider a simple hypothetical example, if the first five elements of a vector were 3, 1, 2, 1, 3, this would mean that the second and fourth amino acids would be assigned to the same cluster (arbitrarily denoted as cluster 1); the first and fifth amino acids would be assigned to another cluster (denoted as cluster 3); and the third amino acid would be assigned to yet another cluster (denoted as cluster 2); and so on, for all the 20 amino acids. The actual value of the group identifier is insignificant. It is used only to identify the group. Different individuals can produce different numbers of clusters, varying from one single cluster (all positions in the vector have the same value) to 20 different clusters (all vector positions have different values). The layout of an immune cell is shown in Fig. 3. In this case only five positions are shown corresponding with the example above, whereas in the algorithm all 20 amino acids will be represented.

Immune cell = <3,1,2,1,3>

Cell vector position	1	2	3	4	5
Corresponding Amino Acid	A	C	T	E	K

Fig. 3 Immune cell encoding. The *top* of the figure shows the actual information encoded by each immune cell while the *bottom* shows how that encoding may be interpreted to define groups for amino acids

3.1.2 The Algorithm's Pseudocode and Search Operators

The opt-aiNet-AA-Clust algorithm proceeds as shown in Pseudocode 2, which is a more detailed description of points 4a–4f from Pseudocode 1.

The algorithm is initialised by generating a population of immune cells such that the representation of each immune cell is in a random configuration. That is, amino acids are randomly assigned to clusters. Next, the quality of each immune cell (the accuracy of the attributes defined by interpreting the clusters defined by that individual) is assessed (see Section 3.1.3). Each immune cell is then cloned (copies of that cell are produced) mimicking the clonal expansion stage of an immune reaction.

These clones are mutated with a rate inversely proportional to their parent's (and therefore their) quality. More precisely, point 2c of pseudocode 2 is implemented by the following formula (applied to each clone i):

$$Num\_mut\_AA\,(i) = k * (1 \; - \; parent\,(i)\,.\,fitness) * length\,(i)$$

Pseudocode 2 opt-aiNet (adapted from [25])

Input
nt = network affinity threshold
nc = number of clones
s = *population size*
Output:
N = set of immune cells in the last iteration (cell with highest fitness is the solution returned to user)
Begin
1. Create an initial random set of immune cells, N, such that $|N| = s$
2. **repeat{**
2a. Determine fitness of each immune cell member of N
2b. Generate nc clones for all elements of N
2c. Mutate attributes of each clone based on the fitness of its parent (P)
2d. Determine the fitness of all new clones
2e. For each P, determine the offspring with the highest fitness (C_h).
2f. If fitness(C_h) > fitness(P) then replace P with offspring C_h in N
2g. Determine the affinity between each pair of cells in N and, for each pair of cells with an affinity less than a pre-specified threshold (nt), remove the lowest affinity cell in that pair from N // network interactions
2h. Introduce new, randomly generated, immune cells into Nsuch that $|N| = s$.
}Until (a stopping condition has been met)
End

where $Num\_mut\_AA(i)$ is the number of amino acids whose cluster indices will be mutated in the clone i, $parent(i).fitness$ is the fitness value of the parent immune cell from which the clone i has been generated, and $length(i)$ is the number of amino acids in clone i (20 in our case). The constant k was empirically set to 4, based on preliminary experiments. Once the number of positions (amino acids' cluster indices) to be mutated has been decided by the above formula, the choices of the actual positions to be mutated are random. The actual value (cluster index) that a position is mutated into is, again, randomly generated—but it is guaranteed to be in the valid range of the cluster indices, of course.

The mutation scheme used in this algorithm is somewhat different to the original opt-aiNet. In the latter, the single value encoded by each immune cell will be incremented or decremented with a magnitude based on its fitness. However, as the representation has changed, so must the mutation scheme and a point mutation in this context is a change in the value of one position in the immune cell's vector (Fig. 4). This has the effect of:

1. Switching an amino acid from one cluster to another; or
2. Taking the amino acid out of a cluster with others and placing it in a cluster on its own, creating a new group; or
3. Taking an amino acid out of a group in which it was the sole occupant, and placing it in an existing group, thus removing that group; or
4. Taking an amino acid out of a group in which it is the sole occupant and placing it in a new group on its own. In this case, the mutation has had no effect on the observed groupings.

The better the solution encoded by an immune cell the fewer positions are mutated. This has the effect of drastically changing poorly performing clustering schemes in the hope that a better solution may be found, while at the same time not destroying solutions that are already good. These newly mutated clones are then assessed for quality once again and, for each parent, its best (highest fitness) clone is determined. If the fitness of that clone is higher than the fitness of its parent, the former replaces the latter in the population.

Next, for all cells in the population, cells who are badly performing (with low fitness) and have low "affinity" with another cell are discarded. Affinity is actually a measure of distance, but we keep the term affinity here to be consistent with the majority of the AIS literature. Affinity is calculated by the well-known Euclidean distance [23]. The discarded cells are replaced in the population with an equal number of randomly configured immune cells, in order to keep the population size constant across all the iterations of the algorithm. This injection of randomness into the population discourages the population converging prematurely on a single local optimum.

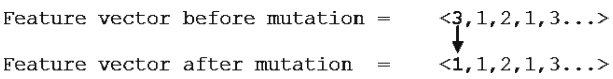

Fig. 4 A point mutation at the first element removes the corresponding amino acid from group 3 and places it in group 1

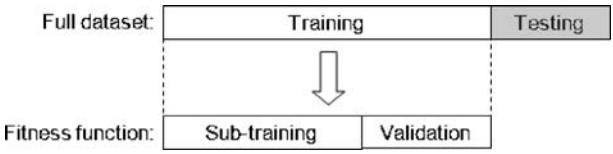

Fig. 5 Division of dataset for training and testing

3.1.3 Fitness Function

The original opt-aiNet used a single mathematical function as a measure of quality whereas the assessment of quality for each immune cell in this scenario is not as straightforward. Several stages must be gone through to assess the quality of the representation as encoded by a single immune cell. As each immune cell encodes a different set of clusters, it is important to note here that the above-described entire process of creating the new training set from the encoded clustering and then training/evaluating the classifier must be repeated every time a fitness evaluation is requested. For each immune cell, the defined clusters must firstly be translated from the immune cell feature vector, as shown in Fig. 3. All protein sequences of a training set are transformed using the defined clusters and the method set out in Section 2.3. This produces an entire dataset consisting of elements with $70n$ predictor attributes (where n is the number of clusters as defined by the particular immune cell). This dataset must then be split into two further sets—sub-training and validation, as shown in Fig. 5. In these tests, the split of the training examples between the sub-training and validation datasets is fixed at 80%/20%. The chosen classification algorithm is now trained on the sub-training set and evaluated using the validation set. The fitness of the cell is the percentage predictive accuracy output from the classifier on that validation set. As all fitness evaluations take place using the same classifier, the changing fitness values are reflective of the representation defined by the cell.

Note how this predictive accuracy is measured on the validation set, rather than the test set. The optimiser cannot have access to the test set until the test stage of the algorithm. If the fitness was determined using this test set then the reported accuracy would be artificially inflated as the optimisation would have taken place using the test data for guidance.

Pseudocode for the fitness evaluation is shown in Pseudocode 3. This entire process must be executed each time an immune cell is required to assess its fitness. That is, lines 2a and 2d of Pseudocode 2.

3.1.4 Parallel Processing

Each iteration of opt-aiNet-AA-Clust may require many hundreds of fitness evaluations to occur. The fitness evaluation in this AIS is therefore extremely processor-

Pseudocode 3 Pseudocode for fitness evaluation

1. Generate groupings by interpreting immune cell features
2. Generate new dataset using these groupings.
3. Split this dataset into sub-training and validation
4. Train classifier on sub-training set
5. Test classifier on validation set
6. Return classification accuracy on validation set as fitness of this cell.

intensive and as such the assessment of immune cell fitness was distributed over a cluster of 30 computers. As the proportion of the time taken to perform the fitness evaluation was the overwhelming majority of the time taken to complete a run of the algorithm, the fitness procedure was the obvious point for parallelisation. It proved fairly straightforward to parallelise the fitness function in this case. Each node in the cluster acted as a server with the main algorithm as a client. Although it will change from run to run, during a single run of the algorithm the training set will not change. Therefore, as each node in the cluster has its own copy of the training partition of the data set, each fitness evaluation becomes entirely atomic in nature. In this case, the algorithm can proceed until it reaches line 4a in Pseudocode 1. At this point lines 4a–4d would usually be executed once for each member of the population, one after another. Instead multiple fitness evaluations can occur simultaneously. At line 4a, the main thread of control will distribute the feature vector of each member of the population to a node in the cluster of servers and then pause. As every node has its own copy of the training and validation set it can create the groups from the feature vector and process the training set to create the dataset unique to that individual cell. From then on each server can apply the converted data to a pre-selected classification algorithm (each node will use the same algorithm of course). Once the server has trained the algorithm on the converted dataset and tested on the same, the percentage accuracy may be returned. This accuracy is then associated with the original immune cell as that cell's current fitness. The main algorithm will remain paused until all evaluations are complete. The main algorithm can then resume and continue as if the fitness evaluations had taken place in the normal, serial manner.

It can be seen therefore that if the number of members of the population is kept lower than the number of nodes (n) in the computing cluster, all fitness evaluations will complete in the time of the slowest fitness evaluation. If the number of members of the population (p) is $n + 1 < p < 2n$ the fitness evaluation must take place in two rounds with n evaluations in the first and $p - n$ taking place in the second. The total time will be the sums of the slowest evaluation in round 1 and the slowest evaluation in round 2. As such, it was decided to keep the number of cells in the population lower than the number of nodes in the cluster to keep the evaluation time to the minimum possible. Preliminary tests showed that a small number of cells was quite sufficient to maintain an acceptable level of optimisation performance. As such, only 20 cells were used as this resulted in up to ten redundant nodes in the cluster at any one time. This was done as a precaution as, if at any time p exceeded n the evaluation time for every fitness evaluation doubles and the run would not complete in an acceptable amount of time. It was found that executing these fitness evaluations in parallel was the only way to ensure the algorithm completed a reasonable number of iterations in a reasonable amount of time.

4 Computational Results

4.1 Test Protocol

The new variant of opt-aiNet proposed in Section 3—opt-aiNet-AA-Clust—was implemented by modifying the original opt-aiNet's code kindly obtained from Andrews [26], which formed part of [27]. The WEKA data mining toolkit [28] was

used to provide the classification algorithm used in the fitness function, many of the algorithms used in the selective top-down classifier and a number of auxiliary functions regarding data manipulation. Some algorithms from [29] were also used in the selective top down classifier. The dataset used for training and testing was our own comprehensive dataset of GPCR sequences. This dataset, called the GDS dataset, originally contained 8,354 protein sequences (examples) in 108 classes, but classes with fewer than ten examples were discarded—since: (a) in general such rare classes cannot be reliably predicted and (b) when diving the dataset into ten folds for a cross-validation procedure, it ensures we can include at least one example of each class in each fold. Once examples of these classes had been removed, this left 8,222 protein sequences in the dataset. The dataset contains five classes (A–E) at the family level (the first level), 40 classes at the sub-family level and 108 classes at the sub-sub-family level (the third level). This dataset is described in more detail in [13].

This full dataset was divided up into ten folds and the data was sampled such that each fold contained at least one example of every class. All tests were therefore, run ten times, once for each fold. Each training set was split into sub-training and validation folds for the optimiser in the ratio 80%/20% (as shown in Fig. 5) and for each of the training sets used by the optimiser, the number of training examples was reduced to half size by random sampling. It was found that this reduced training set size is significantly quicker to process by the rather processor intensive fitness function whilst having no noticeable impact on the quality of the final result. To clarify, for each fold, the optimiser created the groupings using a reduced training set, these groupings were used to create the protein representations for the full training and testing sets. The full training set was used to train the classifier and the full test set was used to test the classifier and therefore determine the quality of the representations.

For each opt-aiNet-AA-Clust run, the algorithm runs for 40 generations, using a population size of 20 individuals. While the algorithm was allowed to form clusters using any combination of amino acids, a limit of five clusters per individual was enforced. Because of the way the clustering is used to produce the predictor attributes, a large number of clusters per individual results in a very large number of predictor attributes, and so the classifier becomes too slow to train and test in a reasonable amount of time. Thus, it was decided that five clusters struck a reasonable balance between the algorithm's flexibility and constraining the time taken during evaluation of the representation.

Ideally, the opt-aiNet-AA-Clust's fitness function would use a classification algorithm to predict classes in all three hierarchical levels of GPCR function. However, this is prohibitively slow with each individual evaluation likely to take many hours. Clearly a faster solution must be found. It was decided that just one classifier should be used in the fitness function. As 1-Nearest Neighbour (1-NN) has appeared to be the more accurate than other classifiers on this type of data in preliminary tests, it was chosen here. As only one classifier is to be used, it was decided that for the purpose of fitness computation the fitness function classifier will distinguish between classes only at the top level of the hierarchy (GPCR families A–E). Table 2 shows the parameters used for each run of opt-aiNet-AA-Clust.

To assess the effectiveness of the proposed algorithm, an experiment was undertaken to compare the accuracy of a classifier when attributes are evolved by the algorithm against a baseline. As stated above, the dataset used was our GDS

Table 2 opt-aiNet-AA-Clust parameters

Population size (s)	20
Number of clones for each immune cell during clonal selection (nc)	20
Suppression threshold for network cell affinities (nt)	0.5
Number of algorithm iterations	40
Maximum number of clusters that can be produced by each immune cell	5

dataset. In the case of the baseline, attributes were generated from raw protein sequences by the approach of Tong et al. [2], as described earlier. In other words, the experiments compare the performance of a given hierarchical classification method in two different scenarios, using two different types of predictor attributes: the attributes created by using our proposed opt-aiNet-AA-Clust and the baseline attributes proposed by Tong et al.. Hence, what is ultimately being compared is the effectiveness of two different protein representations: one of them automatically evolved by opt-aiNet-AA-Clust and the other proposed in [2] using the author's domain knowledge about proteins and amino acid properties.

4.2 Test Algorithm

Both protein representations were tested using the same classification algorithm. This was our selective top down classifier, previously described in [13] and [14]. The selective top down technique is an improvement on the standard top down technique which works as follows. Given, for example, the class tree in Fig. 2a, a tree of classifiers is built to reflect the structure of the classes, as shown in Fig. 2b. Thus a tree of classifiers is generated such that the output of one classifier constitutes the input for another. The number of layers of classifiers will be equal to the number of levels represented by the class attribute. As practically any standard, well known classifier can be used at each node the process of building a hierarchal classifier is greatly simplified. No special classifier must be written to perform the classification (other than the scaffolding required to support a classifier tree). Rather, common well understood classifiers can be used and as such, informed choices can be made about which to use.

The manner in which the top-down approach works takes advantage of the hypothesis that some characteristics of the data may be important to discern between two classes at one node of the class tree while being irrelevant at another. For example, certain attribute values may be equal in classes A.1 and A.2 (making this attribute useless to discern between the two) but yield 100% accuracy when used to choose between B.1 and B.2.

The top-down approach to classification exploits this as all classifiers are trained using only examples (data instances) of the class they are required to classify between. Despite this variation of the training data at each node in the class tree, in the standard top-down approach the same classification algorithm is used in each node of the class tree. Intuitively, this is unlikely to lead to a maximization of classification accuracy. It is natural to hypothesise that different classifiers may be more suited to different nodes in the class tree. Each type of classifier has its own bias and we hypothesise it is possible to maximise the classification accuracy of the top-down approach by using different classification algorithms in the classifier hierarchy. These classifiers are selected in a data-driven manner using the training

set. We call this the selective top-down approach and it has been successfully applied to the classification GPCR classification problem previously [13, 14] – using a very different protein representation and no optimisation of amino acid groupings.

The selective top-down approach proceeds as follows. A tree of classifiers is produced with a structure identical to the standard top-down approach. However, at each node the training data for that node is split into a sub-training and validation set. A number of different classifiers are then trained using this sub-training set and tested using the validation set. The classifier which yields the highest classification accuracy in the validation set is selected as the classifier for this node in the class tree. The sub-training and validation sets are then merged to produce the original training set again, and the selected classifier is then re-trained. The following classifiers were used in the selective top down algorithm, where all were the standard implementations from Weka [28] or the Weka Class Algorithm package [29]:

1. Naïve Bayes
2. Bayesian network
3. SMO (a support vector machine [30])
4. One nearest neighbours (using Euclidean distance)
5. PART (a decision list [31])
6. J48 (an implementation of C4.5)
7. Naïve Bayes tree (a decision tree with a naïve Bayes classifier at each node)
8. AIRS2 (a classifier based on the Artificial Immune System paradigm [32])
9. Conjunctive rule learner

Note that this selective approach effectively produces a hybrid hierarchical classification system, since different nodes in the class tree each use potentially different types of classifiers. It should be emphasised that the actual classifier used for testing the protein representation is not under scrutiny here, it is the optimised representation output by opt-aiNet-AA-Clust we are assessing and we use the selective top down classifier to aid us in doing this.

4.3 Results

opt-aiNet-AA-Clust was evaluated by running a ten-fold cross-validation procedure—a standard procedure for evaluating predictive accuracy in data mining [28]. Each iteration (fold) of the cross-validation procedure took about 6 days, so that the entire cross-validation procedure took about 2 months. The results are shown in Table 3 where the predictive accuracy at all three levels of the class hierarchy is shown for each of the ten folds. This table also shows the mean predictive accuracy over these ten folds, the mean accuracies for the baseline and finally the statistical significance of the difference between the accuracies of the evolved representation and the baseline (attributes created using the grouping scheme of Tong et al. [2], and predictive accuracy ascertained using the same top-down classifier). This has been computed using Student's t-test with two-tails. This test was used for two reasons. Firstly the number of runs is fairly small and this test is designed to be used for small numbers of observations, while secondly it can be used to compare two sets of results where there are different numbers of observations for each. In this case, ten observations for the evolved attributes and 100 for the baseline.

Table 3 Output from classifier on test set

		1st level (%)	2nd level (%)	3rd level (%)
Classifier using evolved attributes	Fold 1	95.87	82.54	69.54
	Fold 2	96.96	82.95	75.49
	Fold 3	97.69	82.80	72.63
	Fold 4	96.23	82.07	73.26
	Fold 5	96.59	83.78	74.91
	Fold 6	97.57	84.60	74.11
	Fold 7	97.08	84.39	73.58
	Fold 8	96.23	84.49	70.51
	Fold 9	98.18	81.73	72.47
	Fold 10	96.72	82.03	71.01
	Mean	96.91	83.14	72.75
	Std. dev.	0.732	1.095	1.927
Classifier using baseline attributes		96.97	82.72	70.46
P value result of Student's *t*-test		0.775	0.280	0.003

It can be seen from the table that the difference in the predictive accuracy of the two approaches on the first (most general) and second class levels are statistically negligible—the *t*-tests produced high *p* values. On the other hand, at the third class level the attributes evolved by opt-aiNet-AA-Clust led to a very significant improvement in predictive accuracy over the baseline attributes, statistically significant at the 1% level. This is especially interesting as the fitness function only dealt with the top level of the hierarchy. It would, therefore, be expected that accuracy would be maximised on this level. Instead it is at the third (most specific) level that appears to have benefited most from the evolved representation. It should be noted that the third class level represents the most challenging classification scenario, since it involves many classes and typically a smaller number of examples per class (making generalization more difficult), as compared with the first two levels. In addition, classes at the third level are often more informative to biologist users, since they specify a protein's function more precisely.

The reasons for this unexpected result are unclear. It is possible that the optimiser is overfitting to the training data. Overfitting refers to the construction of a classification model that is too tailored to the training set, resulting in a relatively low generalization ability and reduced predictive accuracy in the test set (unseen during training) [28].

It should be stressed that, although the automatically evolved clusters of amino acids have led to an improvement for the particular dataset of GPCR proteins used in our experiments, there is no guarantee that the same evolved amino acid clusters will be optimal for predicting other types of protein functions. However, the proposed algorithm is generic enough to be easily applicable to other types of proteins, offering us an automated approach for trying to find a near-optimal cluster of amino acids tailored to the type of protein whose functions have to be predicted.

During the optimisation stage of the process, the fitness of the population was recorded during every iteration. Figures for the number of groups represented by the individuals in the population were also recorded. Upon initialisation, the vast majority of the individuals in the populations contained the maximum of five groups. It was found that the tiny number of individuals representing four groups

quickly gained another group. Almost without exception this occurred within the first few iterations so that all members of the population represented five groups. This result mirrored those previously observed using this technique. In [18] it was found that the optimum number of groups on this dataset was around 12 and when any individuals were deliberately initialised with fewer than this, the number of groups that individual represented would quickly increase. It is thought, therefore, that allowing an unlimited number of groups would result in better accuracy on the test set. However, this is impractical as, firstly, the optimiser will have a hugely increased solution space to search which would require an increase in time taken to optimise. Secondly, increase in the number of groups defined by the fittest individual would result in a huge number of attributes being created for the data, which can be impractical when using the selective top-down technique for classification.

5 Conclusions

Previous experience has shown that the protein representation generated by the local descriptors method results in highly competitive predictive accuracies when attempting to classify GPCR proteins. The local descriptors technique, as currently published in the literature, divides amino acids into three clusters, leading to a specific set of predictor attributes. When evaluating this published representation, we found no clear reason why these three clusters were used. It was therefore hypothesised that predictive accuracy could be improved over this "one size fits all" set of clusters by assigning amino acids to clusters in a data driven manner. In this spirit, this paper proposed a new variant of opt-aiNet, called opt-aiNet-AA-Clust, that optimizes the clustering of amino acids for the type of protein being mined and for the type of classification algorithm being used.

When compared against the original local descriptors-based representation, which was not optimized for the data nor for the classification algorithm, it was found that a significant increase in predictive accuracy was observed at the third level of the class hierarchy, which is the most informative (most specialized) type of protein function for the user. However, it was interesting to note that the accuracy at the top level of the hierarchy did not differ significantly between the evolved representation and the baseline. This was unexpected, as the classification accuracy at the top level of the class hierarchy is used in the optimiser's fitness function to guide the optimisation process.

One future direction would be to let the AIS algorithm have free reign to decide the number of clusters. It is thought that allowing an unlimited number of clusters could result in better predictive accuracy. In these tests we manually enforced a maximum of five groups on the optimised groupings. However, other experiments [18] have shown that the optimum number is probably nearer 12. However, in the experiments reported here this was impractical as, firstly, the AIS would have a hugely increased solution space to search, which would require an increase in time taken to solve the clustering problem. Secondly, an increase in the number of clusters defined by the solution returned by the AIS would result in a huge number of attributes being created for the data, which can be impractical when using a hierarchical classification algorithm. More computing power will have to be sourced before this can begin, but such an investigation has the prospect of yielding some very

interesting results and, if the final representation was combined with an attribute selection algorithm, for example, the predictive accuracy could be enhanced with little impact in terms of time when making classifications.

Acknowledgements The authors should like to gratefully acknowledge funding under the Engineering and Physical Sciences Research Council grant EP/D501377/1—A Synergistic Integration of Natural and Artificial Immunology for the Prediction of Hierarchical Protein Functions. The authors are also grateful to the systems research group at the University of Kent for allowing the use of the pi-cluster of computers, Engineering and Physical Sciences Research Council grant EP/C516966/1—TUNA: Theory Underpinning Nanotech Assemblers (Feasibility Study).

References

1. Cui, J., et al.: Computer prediction of allergen proteins from sequence-derived protein structural and physicochemical properties. Mol. Immunol. **44**, 514–20 (2007)
2. Tong, J.C., Tammi, M.T.: Prediction of protein allergenicity using local descriptions of amino acid sequence. Front. Biosci. **13**, 6072–6078 (2008)
3. Chothia, C., Finkelstein, A.V.: The classification and origins of protein folding patterns. Ann. Rev. Biochem. **59**, 1007–1035 (1990)
4. Christopoulos, A., Kenakin, T.: G protein-coupled receptor allosterism and complexing. Pharmacol. Rev. **54**, 323–374 (2002)
5. Gether, U., et al.: Structural basis for activation of G-protein-coupled receptors. Pharm. Toxicol. **91**, 304–312 (2002)
6. Bissantz, C.: Conformational changes of G protein-coupled receptors during their activation by agonist binding. J. Recept. Signal Transduct. Res. **23**, 123–153 (2003)
7. Hebert, T.E., Bouvier, M.: Structural and functional aspects of G protein-coupled receptor oligomerization. Biochemical Cell Biology **76**, 1–11 (1998)
8. Schoneberg, T., et al.: Mutant G-protein-coupled receptors as a cause of human diseases. Pharmacol. Ther. **104**, 173–206 (2004)
9. Klabunde, T., Hessler, G.: Drug design strategies for targeting G-protein coupled receptors. ChemBioChem **3**, 928–944 (2002)
10. Kolakowski Jr., L.F.: Gcrdb: A G-protein-coupled receptor database. Recept. Channels **2**, 1–7 (1994)
11. Attwood, T.K., Findlay, J.B.: Design of a discriminating fingerprint for G-protein-coupled receptors. Protein Eng. **6**, 167–176 (1993)
12. Attwood, T.K., Findlay, J.B.: Fingerprinting G-protein-coupled receptors. Protein Eng. **7**, 195–203 (1994)
13. Davies, M.N., et al.: On the hierarchical classification of G protein-coupled receptors. Bioinformatics **23**(23), 3113–3118 (2007)
14. Secker, A., et al.: An experimental comparison of classification algorithms for the hierarchical prediction of protein function. Expert Update (Magazine of the British Computer Society's Specialist Group on AI), Special Issue on the 3rd UK KDD (Knowledge Discovery and Data Mining) Symposium **9**(3), 17–22 (2007)
15. Sandberg, M., et al.: New chemical descriptors relevant for the design of biologically active peptides. A Multivariate Characterization of 87 Amino Acids. J. Med. Chem. **41**(14), 2481–2491 (1998)
16. Guan, P., et al.: Analysis of peptide-protein binding using amino acid descriptors: Prediction and experimental verification for human histocompatibility complex Hla-A0201. J. Med. Chem. **48**(23), 7418–7425 (2005)
17. Davies, M.N., et al.: Proteomic applications of automated GPCR classification. Proteomics **7**(16), 2800–2814 (2007)
18. Davies, M.N., et al.: Optimizing amino acid groupings for GPCR classification. Bioinformatics **24**(18), 1980–1986 (2008)
19. Li, T., et al.: Reduction of protein sequence complexity by residue grouping. Protein Eng. **16**(5), 323–330 (2003)
20. Cannata, N., et al.: Simplifying amino acid alphabets by means of a branch and bound algorithm and substitution matrices. Bioinformatics **18**(8), 1102–1108 (2002)

21. Luthra, A., et al.: A method for computing the inter-residue interaction potentials for reduced amino acid alphabet. Biosciences **32**, 883–889 (2007)
22. Freitas, A.A., de Carvalho, A.C.P.L.F.: A tutorial on hierarchical classification with applications in bioinformatics. In: Taniar, D. (ed.) Research and Trends in Data Mining Technologies and Applications, pp. 175–208. Idea Group (2007)
23. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, New York (2002)
24. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer, New York (2002)
25. de Castro, L.N., Timmis, J.: An artificial immune network for multimodal optimisation. In: Proceedings of: 2002 congress on evolutionary computation (CEC 2002). Part of the 2002 IEEE world congress on computational intelligence, pp. 699–704 (2002)
26. Andrews, P.: Opt-Ainet source code in Java. Accessed October 2007 (2005)
27. Andrews, P.S., Timmis, J.: On diversity and artificial immune systems: Incorporating a diversity operator into Ainet. In: Proceedings of: International workshop on natural and artificial immune systems (NAIS), pp. 293–306 (2005)
28. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2005)
29. Brownlee, J.: Weka classification algorithms. Version 1.6. Accessed February 2007 (2006)
30. Keerthi, S.S., et al.: Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput. **13**(3), 637–649 (2001)
31. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: Proceedings of: Fifteenth international conference on machine learning (1998)
32. Watkins, A., Timmis, J.: Artificial immune recognition system (AIRS): Revisions and refinements. In: Proceedings of: 1st International conference on artificial immune systems (ICARIS 2002), pp. 173–181 (2002)