INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

NOTE TO USER

This reproduction is the best copy available.

UMI

Motif-Based Mining of Protein Sequences

Agatha H. Liu

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

University of Washington

2002

Program Authorized to Offer Degree: Computer Science and Engineering

UMI Number: 3053531

Copyright 2002 by
Liu, Agatha H.

All rights reserved.

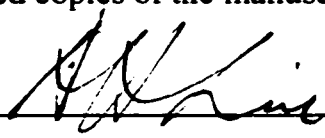
UMI Microform 3053531
Copyright 2002 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

©Copyright 2002

Agatha H. Liu

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to ProQuest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform.

Signature 

Date June 4, 2002

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Agatha H. Liu

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

Walter L. Ruzzo

Reading Committee:

Walter L. Ruzzo

Andrea Califano

Gustavo Stolovitzky

Date: June 4, 2002

The University of Washington

Abstract

Motif-Based Mining of Protein Sequences

Agatha H. Liu

Chair of the Supervisory Committee:

Professor: Walter L. Ruzzo

Computer Science of Engineering

We introduce CASTOR, an automatic, unsupervised system for protein motif discovery and classification. Given amino acid sequences for a group of proteins, CASTOR generates statistically significant motifs and constructs a classification of the proteins by performing motif discovery and refinement in a top-down and recursive manner. The members of each class are likely to share a function, and the motifs associated with the class are likely to account for the function.

We evaluate CASTOR's performance on the G protein-coupled receptor (GPCR) superfamily. The results show that the CASTOR-constructed classification is in better agreement with a manually curated classification than one constructed by another automatic, unsupervised system based on pairwise, global sequence similarity. Furthermore, while manually constructed classifications tend to be hierarchical, the CASTOR-constructed ones that are non-hierarchical suggest that complex functional relationships among classes may be more abundant than expected.

We also apply CASTOR to the mammalian olfactory receptor family, for which very little functional information is available. We infer the potential functional roles associated with the generated motifs and classes by integrating various complex data, such as mutation experiments and ligand binding assays. Among other functional insights gained, we obtain results that support previous hypotheses on structural integrity and post-translational modification. We also propose and provide evidence for a combinatorial molecular mechanism that supports and potentially explains the ligand binding behavior. We additionally define sub-sequences that capture structural features of these receptors and study the motifs present in the sub-sequences.

Finally, we introduce CASTOR+, an automatic, supervised system for protein classification. CASTOR+ adds new proteins to a pre-existing classification where each class is associated with specific motifs, such as that generated by CASTOR, by matching selected motifs in the given classification against each new protein. We evaluate the performance of CASTOR+ on the GPCR superfamily. We find that it performs

almost as well as an approach based on pairwise, global sequence similarity in terms of classifying proteins against the bottom level of the manually curated classification. Furthermore, it often succeeds even as the other approach fails when the new proteins have no close homologues in the pre-existing classification.

Table of Contents

Table of Contents i

List of Figures iv

List of Tables v

Introduction 1

CASTOR: Clustering Algorithm for Sequence Taxonomical Organization and Relationships 7
1 Introduction 7
2 Methods 9
2.1 Tree Approach 9
2.2 Non-Recursive Graph Approach 10
2.3 Recursive Graph Model 11
2.4 Pattern Masking, Class Space Pruning, and Structure reorganization 12
2.5 Discovery and Refinement of Statistically Significant Motifs 13
2.5.1 Motif Discovery 13
2.5.2 Motif Extension 14
2.5.3 Motif Refinement 16
2.5.4 Pattern Matching 17
2.5.5 Motif Masking 18
2.6 Class Space Pruning 18
3 Structure Reorganization 20
4 Recursive Graph Model and Structure reorganization 24
5 Binary Tree 27
6 Results 28
6.1 Sample Protein Superfamily 29
6.2 Set Mapping 29
6.2.1 Preparations and Considerations 30
6.2.2 Procedure 30
6.2.3 Results and Analysis 32
6.3 Motif Mapping 34
6.3.1 Preparations and Considerations 34
6.3.2 Procedure 35
6.3.3 Results and Analysis 35
6.4 Residue Mapping 36
6.4.1 Preparations and Considerations 36
6.4.2 Procedure 37
6.4.3 Results and Analysis 38
6.5 Graph Construction and Structure reorganization 38
7 Related Work 39
7.1 General Comparison with Other Systems 40
7.2 Performance Comparison with ProtoMap 41
7.2.1 Procedure 41
7.2.2 Analysis 42
8 Conclusion 43

Motif-Based Construction of a Functional Map for Mammalian Olfactory Receptors 63
1 Introduction 63

2 Results 66
2.1 General Properties of Olfactory Receptors 66
2.2 Overview of 1D Motifs 67
2.3 Study of 1D Motifs with Respect to Known OR Subgroups 67
2.3.1 Motifs Characterizing the Complete OR Database 68
2.3.2 Motifs Characterizing Class I and Class II Subgroups 68
2.3.3 Motifs Characterizing Human and Mouse Subgroups 69
2.3.4 Phylogenetic Clusters 69
2.4 Analysis of Individual 1D Motifs 70
2.4.1 Sites with Structural Roles 70
2.4.2 Post-Translational Modification Sites and Other Functional Sites 71
2.4.3 Sites Matching Mutation Experiments in Other GPCRs 72
2.5 Analysis of Multiple 1D Motifs 73
2.5.1 Sites with Similar Functions 73
2.5.2 Sites Involved in Ligand Binding 74
2.6 3D Sequences 75
2.7 Overview of 3D Motifs 75
2.8 Study of 3D Motifs with Respect to Known OR Subgroups 77
2.9 Analysis of Individual 3D Motifs 79
2.10 Analysis of Multiple 3D Motifs 80
3 Discussion 80
3.1 Combinatorial Molecular Mechanism for Ligand Binding 81
3.2 Conservation in 3D Space 81
3.3 Structures and Post-Translational Modifications of ORs 83
3.4 Ligand Binding of ORs 84
4 Methods 85
4.1 Construction of Complete OR Database 85
4.2 Application of CASTOR to Complete OR Database 86
4.3 Construction of Global Multiple Alignments and Global Consensus Sequences 86
4.4 Construction of Snake Diagrams and 2D EC Cross-Section 87
4.5 Computation of Feature Values for Sequences and Motifs 87
4.6 Construction of Motif Alignment 87
4.7 Construction of Phylogenetic Tree 87
4.8 Computation of Specificity to a Group of ORs with Respect to a Bigger Database 88
4.9 Comparison between Different Groups of ORs 88
4.10 Comparison between Discovered Motifs and Known Motifs for Known Protein Families and Domains 90
4.11 Comparison between Discovered Motifs and Known Mutation Residues in GPCRs 91
4.12 Identification of GVLC Positions 91
4.13 Construction of Complete 3D Database 91
4.14 Application of CASTOR to 3D Sub-Database 92
4.15 Computation of Feature Values for 3D Motifs 92
4.16 Comparison between 3D Motifs and 1D Motifs 93
CASTOR+ 113
1 Introduction 113
2 Methods 114
2.1 Full-Taxonomy Implementation 114
2.2 Partial-Taxonomy Implementation 116
3 Results 117
3.1 Full-Taxonomy Implementation 119

3.2 Partial-Taxonomy Implementation 121
3.3 Reconstructed Taxonomy vs. Extended Taxonomy 122
3.4 Refinements of Partial-taxonomy Implementation 124
4 Related Work 125
5 Conclusion 127
Bibliography 135

List of Figures

1. Motif Presence and Family Formation 45
2. Binary Tree 46
3. One-Level List 47
4. Recursive Graph Model 48
5. Motif Extension 49
6. Motif Masking and Residue-Level Composition 50
7. Overlap of *M*-Matrices 51
8. Graph Construction and Structure Reorganization 53
9. Motif Mapping 54
10. Overlap between One *C*-Motif and Several *P*-Motifs 55
11. *C*-Motifs and *P*-Motifs for the Same Family 56
12. Interesting Family and Sub-Families 58
13. Classification for One Family 59
14. Comparison between CASTOR and ProtoMap 60
15. Color-Coded Reference Taxonomy (Part I) 61
16. Color-Coded Reference Taxonomy (Part II) 62
17. One-Dimensional Variability Plot 94
18. Two-Dimensional Variability and Hydrophobicity Plots 95
19. Distribution of of Motif Occurrence 102
20. Organization of Classes 103
21. Interesting Sites 104
22. Similar Motifs 105
23. Globally Variable but Locally Conserved Positions 106
24. Three-Dimensional Sequence 107
25. Three-Dimensional Motif Information 108
26. Genuine Three-Dimensional Motifs 109
27. Other Interesting Three-Dimensional Motifs 110
28. Partial-Taxonomy Implementation 129
29. CASTOR-Generated Taxonomy and Partial-Taxonomy Implementation 130
30. Distribution of Number of Visited Classes 133
31. Distribution of Support 134

List of Tables

1. Exploration of Parameter Space 52
2. Residue Mapping 57
3. One-Dimensional Motif Information 96
4. Three-Dimensional Motifs with Large Supports 111
5. Three-Dimensional Motifs with Small Hydrophobicity Indices 112
6. Summary of Classification Performance 131
7. Instances of Sole CASTOR Success 132

Acknowledgement

I owe the completion of my Ph.D. study to many people, who have helped me in various ways during the past few years. I would like to express my deep and sincere gratitude to the extent that I can, which is exhaustive by no means.

I would like to thank Dr. A. Califano at First Genetic Trust. Professionally, I thank him for first taking me as a summer intern at IBM Research. I thank him for subsequently taking me as a long-term co-op and becoming my advisor. I further thank him for remaining to be my advisor after he left IBM Research to start his own company. I would not have made it without his extraordinary guidance and support. I appreciate his extensive consideration and remarkable patience, and I respect him for being highly reliable. Personally, I thank him for his genuine care and support. I also appreciate his delightfulness and complexity. I had not expected to meet someone with whom I shared this much both professionally and personally, and I cannot be more grateful.

I would like to thank Dr. G. Stolovitzky at IBM Research. I thank him for being my co-advisor and providing a rich and exciting work environment. I thank him for the detailed and insightful comments and discussions. I further thank him for the tremendous cooperation and assistance in many respects. I also appreciate his uncommon sincerity and kindness.

I would like to thank a few people at IBM Research. I thank Dr. A. Royyuru for providing prompt and appropriate assistance and being accessible and friendly. I thank Dr. J. M. Jasinski for his critical assistance and consideration. I thank R. W. Krull for providing prompt and appropriate support and for being accessible and friendly.

I would also like to thank many other people (once) at IBM Research for the various assistance and support. They include Dr. A. Apostolico (no longer at IBM Research), Dr. R. Hart (no longer at IBM Research), Dr. R. Mushlin, Dr. J. J. Rice, Dr. I. Rigoutsos, and Dr. Y. Tu.

I would like to thank Prof. W. L. Ruzzo at University of Washington. I thank him for being my co-advisor and overcoming the difficulty associated with remote communication. I thank him for the detailed and insightful comments and discussions. I further thank him for the tremendous cooperation and assistance in many respects. I also appreciate his remarkable patience and kindness.

I would like to thank a few people (once) at University of Washington. I thank Prof. R. M. Karp (no longer at University of Washington) for being my (previous) advisor and providing invaluable assistance and inspiration. I thank Prof. P. Tseng for providing substantial assistance and support and for being friendly and accessible. I thank Prof. D. Notkin for being my mentor and for being friendly and caring. I thank F. Jones and L. Michimoto for handling all the complications due to my on-leave status and for being friendly and accessible.

I would also like to thank many other people (once) at University of Washington for the various assistance and support. They include Prof. P. Green, Prof. A. Halevy, Prof. Tompa, all the instructors of the courses I took from Department of Computer Science and Engineering, Department of Mathematics, and Department of Molecular Biotechnology (now part of Department of Genome Sciences), and all the people at the computational biology group with whom I worked.

I would like to thank a few people at Columbia University. I thank Prof. S. J. Firestein for his substantial assistance and support. I also appreciate his dynamic and intriguing presence. I thank X. Zhang for the intellectually stimulating discussions. I further thank him for the solid and smooth cooperation. I would also like to thank the other people at Prof. Firestein's lab for the various assistance and support.

I would like to thank a few people at Novartis Pharmaceuticals. I thank N. R. Nirmala for her substantial assistance and her friendliness. I would also like to thank the other people at the Life Sciences Informatics Unit for sharing helpful discussions.

I would like to thank several people who provided me with assistance, encouragement, and inspiration before I began my Ph.D. study. They include Prof. C. Cardie, Prof. N. Greenleaf, Dr. V. Hatzivassiloglou, Prof. L. Jamieson, Prof. K. McKeown, and C. Tonas.

I would like to thank many people, who I met at school or work and who however offered invaluable friendship. They include C. Anderson, A. Castillo, S. Sobti, and my officemates at Department of Computer Science and Engineering of University of Washington. They also include those who I began to know during my M.Eng. study at Cornell University. I would also like to thank several people who remain my dearest friends. They include T.-H. Chao, Y.-W. Chiu, and those who I began to know during high school.

I would finally like to thank several people, one by one, who have played very special roles in my Ph.D. study and my life in general.

I would like to thank W. Chan. I thank him for being understanding and caring. I appreciate his deep sincerity and sensitivity. I treasure the connection between us, and I will forever remember him fondly.

I would like to thank K. F. Leong. I thank him for his tremendous assistance and support. I thank him for his precious care and encouragement. I also appreciate his courage and respect him for being highly responsible.

I would like to thank Y.-C. Chu. I thank her for her distinct dedication and inspiration. I appreciate her unparalleled perceptiveness and complexity. I cannot describe how much she means to me, and I will forever remember her with heartaches.

I would like to thank my father and sister. I thank them for providing me with an essential source of comfort and peace. I thank them for being understanding and caring. I further thank them for believing in me and depending on me. I would also like to thank my other relatives.

I would like to thank R. Y. Chen. I thank him for his unprecedented care and support. I thank him for his distinct encouragement and inspiration. I treasure his precious tenderness and thoughtfulness, and I appreciate his delightfulness and warmth. It is my privilege to have shared all the moments with him, and I cannot be more grateful.

Dedication

To my mother

Introduction

The Human Genome Project is nearing the end. It has generated tens of thousands of DNA sequences, which encode the secret ingredients of human life. Such a tremendous amount of data earnestly solicits the exertion of the grand computing power, and thus conferred upon the computer scientists is the privileged role of front-line warriors on the battlefield against the almighty creator of human life... The truth is that to accomplish the sacred mission of decoding the human genome as fast as possible, it only makes sense to apply intelligent and efficient algorithms that will shed some light on the underlying code. This mission thus has created a whole new interdisciplinary field of computational biology where research may flourish in the participating fields individually and collaboratively.

In this work, we describe our contributions in the field of computational biology. The ultimate goal is to understand what every human gene does, and it will be a long time before this goal is reached. Today, as human beings are confronted with all the fresh sequence data and struggle to find any clues to the meaning behind them, we study the subject of which parts of the genes may possess higher functional significance and which genes may be grouped together because they possess similar functional significance. In fact, we consider the genes in their "mature" form, the proteins, which may better reveal the hidden meaning. We hope that our study will help narrow down the focus in the search for functional clues and gain some insight into how functions are developed.

Given a group of related proteins with their amino acid sequences available but no or limited functional information available, two important tasks are the inference of subgroups of proteins that share a protein function and the identification of protein regions that possess functional significance. The two tasks are inextricably linked, and this can be captured by motifs as follows. Mutation events occur, and those that result in critical loss or change of protein functions lead to evolutionary dead-ends or the formation of new protein functions. As a result, protein regions that are causally related to a protein function tend to be highly conserved across the subgroup of proteins that share the protein function. Such unusual conservation manifests itself through the presence of motifs or patterns that occur repeatedly in the protein sequences that are highly unlikely to be the product of random processes and are therefore statistically significant based on a specific formulation of a null hypothesis [5]. Conversely, the presence of statistically significant motifs strongly hints at the existence of protein regions that are causally related to a

protein function shared by a subgroup of proteins. We will refer to such a protein region as a *functional region*. It in turn fingerprints a specific subgroup of proteins that share the corresponding protein function. We will refer to such a subgroup of proteins as a *functional subset*.

Intuitively, we would want to approach these two tasks simultaneously simply by applying a pattern discovery algorithm to the entire group of proteins and expecting that it discovers all the motifs that are present in various subgroups. However, it is not practical to apply a complex, effective, powerful, but thus time-consuming pattern discovery algorithm in this case, as the number of possible motifs would increase exponentially with the size of the group of proteins. Even if we consider a less complex and thus less time-consuming pattern discovery algorithm, it still may not suffice to apply it to the entire group of proteins only, as it may easily miss those motifs that are present in such small subgroups that they are not considered statistically significant in the context of the entire group of proteins. We also need to deal with the additional problem of potential reduced accuracy in characterizing the motifs and sensitivity in identifying motif matches due to the lower complexity of the algorithm.

Consequently, the two tasks have generally been approached independently despite the fact that they are inextricably linked. The inference of functional subsets has typically been approached by bottom-up clustering based on pairwise sequence similarity and various linkage rules, where clusters comprising single proteins are first formed and then small clusters are merged to form larger clusters [74][77][86]. The identification of functional regions has typically been approached in a supervised manner based on multiple sequence alignment, where a classification of the proteins into subgroups is given and conserved or otherwise interesting regions are then identified for each subgroup [3][5][37].

In Chapter 2, we introduce CASTOR [48][49]. It is the first reported system to infer functional subsets and identify functional regions simultaneously. Given a group of related proteins in terms of their amino acid sequences, CASTOR generates potential functional regions and subsets by discovering, refining, and matching statistically significant motifs that are present in the amino acid sequences in a top-down and recursive manner. Specifically, it first deterministically and quickly discovers motifs represented by regular expressions using SPLASH [19] and then efficiently converts the regular expressions into the complex and rigorous profile HMMs using HMMER (http://hmmer.wustl.edu/). Furthermore, it repeats such manipulation of statistically

significant motifs, including the matching of motifs represented by profile HMMs, in a divide-and-conquer manner through a flexible and general model.

It has been shown that SPLASH [19][75] is very likely to generate functional motifs [35]. In particular, more than 75% of the discovered motifs may be associated with important functions in a direct comparison with PROSITE [5]. On the other hand, it has become common to use profile HMMs to represent the amino acid sequences of a protein family. The general structure of a profile HMM naturally captures the formation of amino acid sequences within a protein family in a formal statistical framework, which nonetheless provides rigor and sophistication to the modeling of the protein family [24]. As a result, CASTOR efficiently and systematically generates a list of statistically significant motifs, where each resulting motif corresponds to a potential functional region that is very likely to be a functional region, and its support set corresponds to a potential functional subset that is very likely to be a functional subset.

Furthermore, each resulting motif may occur in any part of a protein, and its support set may consist of any combination of proteins in the given group. In fact, the support sets of the resulting motifs cover all the proteins and constitute a complete classification or taxonomy on the given group of proteins. Since any of these support sets could contain, be contained by, overlap, or be disjoint from another one, the taxonomy could be either a hierarchical one, where only containments are present, or a non-hierarchical one, where partial overlaps are also present.

Once we introduce CASTOR, we explore exactly how it exploits the link between the two tasks and approach them simultaneously. We also investigate how it does so efficiently and yet systematically. In addition, we study the properties of the different types of output corresponding to different sets of values for its parameters. Furthermore, we explore how to incorporate additional complex data, such as secondary structures, when available in performing the two tasks. Finally, we examine how well CASTOR performs using a real protein superfamily as a test case.

To evaluate its performance, we apply CASTOR to the superfamily of G protein coupled-receptors (GPCRs) [81]. This is a significant, large, and diverse protein superfamily. It is responsible for key transmembrane signaling functions in virtually very cell of a complex organism and is one of the favorite targets of study to biologists. A great wealth of sequence data as well as functional information about the superfamily is available. Over 1,000 GPCR protein sequences from multiple species with substantial functional annotations are in SWISS-PROT [6],

and they correspond to over 200 distinct genes that may be classified into more than 70 families. A variety of derived information, such as sequence alignments, phylogenetic trees, motifs, and compilations of mutation experiments, is also accessible in many sources [3][10][39][80]. The nature of the superfamily and the abundance of the available data and information thus make the superfamily an ideal candidate for a test case in assessing the performance of CASTOR.

The main results of the performance assessment are as follows. The potential functional subsets inferred by CASTOR are in better agreement with a manually constructed and biologically meaningful taxonomy on the GPCR superfamily than those inferred by another system that performs bottom-up clustering based on pairwise sequence similarity. Furthermore, while manually constructed taxonomies tend to hierarchical ones, the non-hierarchical taxonomies constructed by CASTOR suggest that unusual relationships among functional subsets may be much more abundant than expected.

In Chapter 3, we further present how we use CASTOR to contribute to current biological research by applying it to the family of mammalian olfactory receptors (ORs) [25][50]. This is one of the largest protein families. There are currently over 1,000 potential human and mouse OR genes and translated protein sequences in GenBank [9], and they correspond to more than 600 distinct genes. The OR family also happens to have one of the largest ligand repertoires. The conceivably huge number of odorant-OR interactions [2][51] calls for many ligand binding sites and other functional regions in ORs and consequently many functional subsets of ORs interrelated in complex ways. However, there is virtually no functional information available about the OR family other than limited ligand-binding profiles for selected ORs [2][36][44] [51][61][79][88]. In particular, it is not even clear how to organize ORs into subfamilies. A few efforts have been made on the identification of global functional regions and specific ligand-binding sites [58][66][90], but the results are yet to be verified by lab experiments. The nature of the family, the abundance of sequence data, and the scarcity of available functional information nevertheless offer a great opportunity for the application of CASTOR to result in significant contributions to current biological research.

In applying CASTOR to the OR family, we not only consider the full-length amino acid sequences of ORs but also define pseudo-sequences that capture the conformational features of the transmembrane regions in the three-dimensional space. Therefore, we explore how to discover motifs in the amino acid sequences of ORs, taking into account not only their primary structures but also the secondary-structures. Upon the application of CASTOR, we continue to

elucidate the functional significance of the potential functional regions and subsets via systematic and powerful studies.

We obtain a substantial body of functional interpretations from such a comprehensive effort. The individual pieces alone are interesting in their own rights. As examples, we identify some protein regions that characterize subgroups of the OR family that are known to have similar ligand binding profiles. We also identify a potential post-translational modification site in the third extracellular region of relatively few ORs. When integrated together, the functional interpretations further amount to compelling evidence for certain findings about the functions of ORs in terms of their structures, post-translational activities, and ligand binding. In particular, we are able to propose a combinatorial molecular mechanism that supports and may explain the combinatorial behaviors of ligand binding. We therefore demonstrate how CASTOR could be applied to a given group of proteins with ease and efficiency as well as how it produces many valuable pointers that may lead to functional discoveries regarding the given group of proteins.

In Chapter 4, we introduce CASTOR+. It is an extension of CASTOR in that it approaches the same two tasks that CASTOR approaches but now at the presence of potential functional information about the given group of proteins. The potential functional information is available in the form of the output of CASTOR, namely a taxonomy on a training group of protein sequences, where each class or potential functional subset is associated with certain potential functional regions, which are described by statistically significant motifs. Therefore, the input to CASTOR+ is a test group of related protein sequences and a corresponding taxonomy constructed by CASTOR. CASTOR+ generates potential functional regions and subsets for the test group by matching some of the motifs in the given taxonomy against each protein sequence in the test group and classifying the protein sequence into the classes in the given taxonomy based on the matching results. Each motif in the given taxonomy directly corresponds to a potential functional region, thus saving the need for discovering and refining statistically significant motifs, and the support set of the motif with respect to the test group of proteins corresponds to a potential functional subset. In fact, this support set may be merged with the support set with respect to the training group of proteins to form an extended potential functional subset.

In this work, since we are interested in gaining some insight into the functions of those proteins with abundant sequence data available but little functional information available, we study the application of CASTOR+ in the following scenario. Given a large group of protein sequences to which we have applied CASTOR to generate a taxonomy and another small group of proteins, we

apply CASTOR+ to the small group of proteins. The result is a taxonomy on the union of the large group and the small group, which is extended from the taxonomy on the large group. In this way, CASTOR+ works solely with the output of CASTOR.

In fact, to obtain a taxonomy on the union of the large group and the small group, another option is to simply apply CASTOR to the union. Therefore, we attempt to get an idea of how these two options might produce different results. Furthermore, we investigate two complementary implementations of CASTOR+, which are expected to demonstrate the tradeoff between output quality and execution speed. We also examine how CASTOR+ performs compared to a more traditional approach based on pairwise sequence similarities.

In our performance evaluation, we use the GPCR superfamily as a test case. The results indicate that CASTOR+ performs almost as well as a BLAST-based approach in terms of classifying proteins against the classes at the bottom level of a manually constructed and biologically meaningful taxonomy on the GPCR superfamily. Furthermore, it often still succeeds even as the BLAST-based approach fails when no sequence is present in the training group that shares extremely high sequence similarity with a sequence in the test group and sequence similarity does not coincide with functional similarity. Furthermore, even if the taxonomy obtained by running CASTOR on the union of the training group and the test group and the one obtained by running CASTOR+ on the test group are of substantially different sizes, they lead to classification results at the lowest level of approximately the same quality. This suggests that while the former taxonomy may offer more detailed and complex information about the underlying functional subsets and regions, it may have the same capabilities for classification at the lowest level as the latter taxonomy.

A few efforts have been made on the classification of potential GPCRs with respect to somewhat different training groups of GPCRs along with somewhat different taxonomies on the training groups of GPCRs [42][68]. These efforts have different emphases, and it is very likely to be the case that one is more suitable for one type of protein families than the others are.

CASTOR: Clustering Algorithm for Sequence Taxonomical Organization and Relationships

1 Introduction

Given a group of related proteins, two important problems in biology are the inference of functional subsets and the identification of functional regions. CASTOR is a high-performance, unsupervised system that approaches both problems simultaneously. For simplicity and clarity, potential functional subsets inferred by experimental means will be referred to as *biological classes* and those inferred by computational means will be referred to simply as *classes*. These terms will be used to describe both subgroups of proteins and lists of their corresponding amino acid sequences.

CASTOR offers some critical advantages over other approaches previously reported. First, classes are directly inferred from and related to statistically significant motifs. The statistical significance may offer important clues to functional relevance. It also ensures that all members assigned to one class are related consistently, rather than indirectly through transitivity, as may happen with bottom-up clustering based on pairwise sequence similarity. The latter is the case with all other automatic systems that perform unsupervised, sequence-based, protein classification, including COG [77], DOMO [32][33], HHS/MSG [74], ProDom [72], ProtoMap [86], and the neighbor joining method [64].

Second, the inference of classes is generally dictated by those protein regions that are more likely to be functional only and is less likely to be affected by the other protein regions even if they share substantial sequence similarity. For example, protein sequences that are somewhat similar perhaps by chance but do not contain instances of the same functional region are generally not assigned to the same class, as different functional regions will be described by different motifs. On the other hand, evolutionarily close proteins might share high sequence similarity, but they would generally be classified based solely on the functional regions.

Third, the identification of potential functional regions is performed by first discovering statistically significance motifs quickly and then efficiently refining the motif representation into sophisticated and rigorous statistical models. Furthermore, while the inference of classes is performed in a divide-and-conquer manner, all the sub-classes that have been inferred from the same class can be processed independently and in parallel. These two features combine to make

it possible that CASTOR scales up and is applicable to large protein superfamilies or even full databases.

Finally, the classes may satisfy not only the containment relationships but also overlaps, thereby resulting in potential non-hierarchical taxonomies. Given an initial subgroup of related proteins, duplication events and mutation events lead to divergent evolution processes under normal circumstances. Over time, these processes tend to produce a taxonomy on a group of proteins that is characterized by a set of hierarchical functional relationships. However, exceptions, such as domain transfer events and convergent-evolution processes, are everywhere in nature, and they tend to result in a different type of functional relationship among otherwise remote functional subsets. **Figure 1** shows two hypothetical functional subsets characterized by individual motifs. Therefore, given a group of related proteins, both types of functional relationships are expected: the first one where more functionally specific functional subsets are contained in less functionally specific ones (e.g., growth hormone-releasing hormone as a subset of secretin-like proteins) and the second one where distant functional subsets share some members and thus overlap. While the former highlights the increasingly specific functional nature of certain functional subsets, the latter indicates subtle and interesting functional links among specific functional subsets.

To evaluate the performance of CASTOR, the G protein-coupled receptor (GPCR) superfamily has been studied. This is a large, significant, diverse protein superfamily. Proteins in this superfamily possess important transmembrane signaling functions and, as such, they have been long-time favorite targets of study to biologists. As a result, the functions of many members of the family are known *a priori*. SWISS-PROT [6] currently includes over 1,000 GPCR protein sequences from multiple species with substantial functional annotations, and they correspond to over 200 distinct genes. A great wealth of derived functional information about this superfamily is also available from a variety of sources, including GPCRDB [39], PRINTS [3], GPCRMD [80], tGrap [10], and SMART [66]. Such information is generated by computational means as well as biological means, which are based on binding assays or site-directed mutation experiments. It will be used as a basis in assessing the performance of CASTOR.

The chapter is organized as follows: Section 2 describes in detail the methodology, including how classes are inferred and how potential functional regions are identified. Section 3 discusses how to refine the inferred classes through structure reorganization when appropriate. Section 4 discusses the recursive graph model and structure reorganization in more detail. Section 5 discusses the output of the purely hierarchical approach reported in [48] in detail. Section 6

studies the effect of the choice of parameter values on the performance of CASTOR. Finally, Section 7 discusses related work.

2 Methods

Given a group of proteins in terms of their amino acid sequences, which will be called the *complete set of sequences S*, CASTOR infers classes from *S* based on the presence of the identified potential functional regions in a top-down, divide-and-conquer manner. It does so via a general and flexible model, which is illustrated by two special instances in Section 2.1 and Section 2.2, respectively, and formally introduced in Section 2.3

CASTOR identifies potential functional regions from *S* using a complex, pan-algorithmic approach that applies several established techniques in a novel way. In [35], it is shown that the application of SPLASH [19], which is a pattern discovery algorithm, coupled with a statistical framework to assess the statistical significance of the discovered motifs [75] leads to motifs that tend to correspond to functional regions. CASTOR enhances this approach and combines pattern discovery with pattern refinement, which yields high efficiency without sacrificing accuracy and sensitivity. Therefore, the resulting potential functional regions will be better characterized and in turn, the classes will be better constituted. This is discussed in detail in Section 2.5. It is important to note that CASTOR could have relied on any of a variety of pattern discovery algorithms. SPLASH is a particularly good choice from a performance standpoint, since it has been shown to outperform other representative pattern discovery algorithms in terms of raw performance, limitations on discovered motifs, and scalability [19].

2.1 Tree Approach

The simplest approach to infer classes from *S*, which leads to a tree, is as follows. Up to a predefined number of motifs are systematically discovered from *S* and ranked according to their statistical significance using the methods presented in [75]. The one highest in rank is selected and is used to divide *S* into two subsets, a *positive set* with proteins that match the motif and a *negative set* with proteins that do not match it. The positive set will also be called the *support set* of the motif. The set of occurrences of the motif or parts of the proteins where the motif occurs will be called the *occurrence set* of the motif. Each of the positive sets and the negative set is considered as a class or a union of classes. If the two subsets are known to be functionally divergent and the resulting partition indeed corresponds to a segregation of functions, it would be

reasonable to assume that the conserved region described by the motif was at some time the locus of a divergent-evolution process that resulted in the functional differentiation of the subsets.

This process can be repeated to subdivide both the positive set and the negative set recursively, until statistically significant motifs can no longer be discovered or the resulting subsets are too small. The result is a hierarchy of classes. It can be readily represented as a binary tree of "arbitrary" depth where for any node, which corresponds to a class, "one" motif is discovered and used to infer both of its children. An illustration of such a binary tree is given in **Figure 2**.

The binary tree can be simplified into a compact n-ary tree as follows. First, if an internal node corresponds to a negative set, both the node and the edge leading to it are removed. Second, if the parent of a node is removed, both the node and the edge leading to it are attached to the closest ancestor that is a positive set. Finally, consecutive nodes along a path that correspond to identical positive sets because the same protein sequences match multiple motifs are fused together, and the corresponding empty negative sets are removed. The resulting structure might better reflect the development of evolutionary inheritance. For instance, the graph-theoretic distance between any two nodes in this new tree might be a better indicator of the evolutionary distance between the corresponding classes.

The completely hierarchical structure described above can serve as a skeleton representation of normal evolutionary development, which is generally characterized by divergent-evolution processes and leads to natural functional relationships among functional subsets. However, it does not take into account exceptions, such as domain transfer events and convergent-evolution processes, which result in unusual functional relationships among otherwise remote functional subsets. This will be addressed in the following sections.

2.2 Non-Recursive Graph Approach

An alternative approach to infer classes from S, which leads to a graph, is as follows. Up to a predefined number of motifs are discovered from S and ranked, and the one first in rank is selected, as described in Section 2.1. This process is then repeated as many times as possible to obtain a group of distinct, statistically significant motifs. These motifs are then used to infer an equal number of positive sets and one additional negative set, which is the complement of the union of the positive sets. The result is a single list of possibly overlapping subsets. The list can be represented as a graph with nodes corresponding to all the positive and negative sets, directed edges that connect nodes corresponding to subsets and the subsets they contain, and undirected

edges that connect nodes corresponding to partially overlapping subsets. An illustration of this approach is shown in **Figure 3**.

This non-hierarchical structure can be considered as the byproduct of both "normal" and "exceptional" evolutionary events and processes, which cause the simultaneous presence of distinct functional regions that characterize different functional subsets in those members shared by these functional subsets. A drawback of such an approach, however, is that since the positive sets and the negative set are not processed recursively, small functional subsets may be missed because motifs that are present in small subsets may not be statistically significant in S.

Either of the structures discussed in this section and in the previous section has its advantages and disadvantages. Both, however, can be thought of as specific instances of a more general and flexible model proposed in the following section.

2.3 Recursive Graph Model

Suppose that for any class, pattern discovery is limited to no more than n_B distinct motifs; n_B will be called the *maximum breadth*. Suppose also that each of the positive sets and the negative set is recursively processed. Such processing can also be limited to no more than n_D recursion levels with respect to S; n_D will be called the *maximum depth*. Then, a general and flexible model is uniquely defined by the two parameters n_B and n_D. Their values determine the number of classes inferred by CASTOR as well as the organization of the classes. The tree discussed in Section 2.1 and the non-recursive graph discussed in Section 2.2 represent two instances of this model, ($n_B = 1, n_D = \infty$) and ($n_B = \infty, n_D = 1$), respectively. **Figure 4** illustrates this model.

For any instance of this model, the number of motifs discovered from each class may not reach the value n_B and the number of recursion levels along any thread of recursive processing also may not reach the value of n_D. This is due to the failure to discover additional statistically significant motifs or other rules that halt recursive processing. For instance, as shown in [35], as a class becomes smaller, so does the likelihood of discovering motifs from the class that correspond to functional regions. Therefore, another possible rule is to halt recursive processing when the class currently under consideration becomes too small.

Recursion imposes a hierarchical organization on the resulting structure. Thus, n_D corresponds to the maximum depth of the hierarchical part of the structure. Every class F is inferred directly from another one F_+ that contains it and will be referred to as its *parent*. Its parent is inferred

directly from another one F_{**}, and so on until S is reached. The unique path in the structure that traces this series of classes from S down to F will be referred to as F's *inference path* or $\Lambda(F)$. For instance, the inference path of $F_{1.3.2}$ in **Figure 4** is $\{S, F_1, F_{1.3}, F_{1.3.2}\}$. All the classes on $\Lambda(F)$ will be referred to as F's *ancestors*.

From each class F, at most n_B classes are inferred directly by pattern discovery. These will be referred to as its *children*. From each child class, in turn, at most n_B other classes can be inferred directly, and so on. The unique sub-structure that starting at F and covering all the classes F' such that F is on the inference path of F' will be referred to as the F's *inference structure* or $\Sigma(F)$. All the classes on $\Sigma(F)$ will be referred to as F's *descendants*.

Therefore, F can be denoted by $F_{j_1 \ldots j_k}$, where j_i identifies the $(i+1)$-th class along F's inference path by specifying its index as a child of the i-th class. S is the first class along every inference path, and the children of a class are ordered the way they are inferred serially, even if they are all independent. Then, if S corresponds to a depth of zero, k is the depth at which F is inferred. In addition, j_k may be considered as the breadth at which F is inferred. It is possible to compare any two classes breadth-wise by saying that F^a is inferred at a smaller breadth than F^b if F^a's ancestor that is a child of F^a's and F^b's youngest common ancestor is inferred at a smaller breadth than F^b's ancestor that is also a child of the youngest common ancestor. Formally, suppose that F^a is denoted by $F_{l_1^a \ldots l_w^a l_{w+1}^a \ldots}$ and F^b is denoted by $F_{l_1^b \ldots l_w^b l_{w+1}^b \ldots}$. That is, up to l_w, their inference paths coincide. If $l_{w+1}^a < l_{w+1}^b$, F^a is said to be inferred at a smaller breadth than F^b.

2.4 Pattern Masking, Class Space Pruning, and Structure reorganization

A few clarifications are in order. First, before the (i+1)-th motif can be discovered from a class, CASTOR "masks off" the i-th motif. This avoids having to consider the i-th motif in all future rounds of pattern discovery and thus re-discovering the same motif or its small variants repeatedly. Such motif masking will be discussed in detail in Section 2.5.5.

Second, when n_B is set to a value greater than one, CASTOR may infer two classes that overlap and thus later infer a class corresponding to the overlap twice. Any such duplicate should be identified and pruned to avoid an undesirable inflation of computational needs. Such class space pruning will be discussed in detail in Section 2.6.

Finally, especially when n_B is set to one, CASTOR may discover a motif twice from two separate classes, since it is actually present in members of both classes, without ever inferring a third class that is equivalent to the union of the two classes. The rectification of such a situation is part of structure reorganization, during which a given structure is cleaned and compacted into a non-redundant one that presents the desirable information in a concise and systematic way, will be discussed in Section 3.

2.5 Discovery and Refinement of Statistically Significant Motifs

CASTOR uses a variety of techniques to discover statistically significant motifs that tend to correspond to functional regions, refine the representations of these motifs, and locate the occurrences of these motifs.

2.5.1 Motif Discovery

CASTOR starts by discovering statistically significant motifs represented as regular expressions, using SPLASH. SPLASH is a deterministic pattern discovery algorithm that guarantees the discovery of every regular expression that occurs repeatedly in a set of sequences, subject to the following constraints:

- The characters of the regular expression are from an alphabet of amino acids, similarity classes, and don't-care. A *similarity class*, such as `[ILMV]`, can be any set of amino acids. For our purposes, it is defined as a set of amino acids that score above a given threshold with respect to the first amino acid in the similarity class, using a predefined scoring matrix such as PAM [67] or BLOSUM [37]. It may alternatively be defined using the method reported in [84] for general purposes. An individual amino acid in the regular expression matches exactly the same residue in a sequence. A similarity class matches any residue in a sequence that belongs to the similarity class. A don't-care matches any residue in a sequence. For instance, `C..[ILMV].[DE]` matches ACRKMVDQP starting at the second residue, with `[ILMV]` matching M and `[DE]` matching D.

- The motif must occur at least j_0 times in the set of sequences or in at least j_0 distinct sequences. In this chapter, the former definition is used.

- The motif must contain at least t_0 *tokens*, which are amino acids and similarity classes but not don't-cares.

- The motif must satisfy a *density constraint*. That is, any substring of l_0 characters in the motif that does not start with a don't-care must contain at least k_0 tokens. If the motif has fewer than l_0 characters, it must contain at least k_0 tokens.
- The motif must satisfy a *maximality constraint*. That is, no token can be "added" to it either at neighboring positions or at internal positions originally occupied by don't-cares without reducing its *support*, which is the number of occurrences of the motif in the set of sequences.

A motif is assigned a *z-score* that is computed from a null model based on a random database of a similar size and composition [75]. Motifs are considered *equivalent* if they have the same support j, length k (number of tokens), and span l (total number of characters including don't-cares). Equivalent motifs are assigned identical z-scores. The z-scores are approximately inversely related to the corresponding p-values. Furthermore, high z-scores tend to correspond to functional regions, as shown in [35].

Given a set of sequences, SPLASH looks for at least N statistically significant motifs before it reports that it has discovered one. The algorithm starts by looking for ubiquitous motifs by setting j_0 equal to the number of sequences N_S. Then, j_0 is gradually reduced by $N_S \times 0.05$. For each new value of j_0, two choices of the density-constraint are explored, by using the specified values for k_0 and l_0 and then by doubling the value of l_0. These steps are repeated until a sample of at least N statistically significant motifs is obtained or until a minimum support J_0 is reached (typically J_0 is set to 3). As soon as at least N motifs are discovered, the most statistically significant one is selected and returned and all the others are discarded. If J_0 is reached, then it is assumed that no more statistically significant motifs can be discovered from the set of sequences.

2.5.2 Motif Extension

The similarity classes described in the previous section are not context specific. That is, they are not likely to realize all possible amino acid substitutions that would preserve a protein function. One possible consequence is that they turn out to characterize only some members of a functional subset and thus do not generalize well. Therefore, regular expressions only provide a very rough representation for the underlying motifs. To address this problem, the regular expressions generated by SPLASH are used as a starting (anchor) point for a more refined representation of profile hidden markov models (HMMs) [24][45].

One major issue is that a regular expression may be significantly shorter than the underlying functional region. This is because those function-preserving amino acid substitutions that cannot be realized by the similarity classes and are located at the boundaries of a functional region would simply be dropped from the corresponding regular expression, while those that are located within the functional region would be kept and shown as don't-cares. It is conceivable to realize more amino acid substitutions that would preserve a protein function by lowering the threshold used in defining the similarity classes, as discussed in the previous section. However, lowering the threshold moderately is not expected to result in major differences in the resulting motifs, as shown in [35], and lowering the threshold too much would result in completely unspecific motifs. Therefore, the conserved region corresponding to a motif discovered by SPLASH is extended on both sides using the information theoretic approach described below, and a profile HMM will be constructed for the extended conserved region.

All the sequences matching a motif are aligned with respect to the motif. The left flanking region and the right flanking region of the conserved region where the motif occurs are then examined separately. The left flanking region will be used as an example, but the extension procedure is similar for the right flanking region. Residue statistics for individual positions in the flanking region are computed and analyzed to determine whether there is substantial conservation. Specifically, the amino acid entropy is computed over a small sliding window with its left side positioned against the left boundary of the set of aligned occurrences of the motif at the initial position. It is then slid to the left position by position until either the entropy E increases by more than a predefined threshold ΔE compared to the value E_0 when the window is at the initial position or when the beginning of a sequence is reached. At the end of this process, each occurrence of the motif is extended to the left all the way to the left side of the window at the last position before the increase of entropy exceeds ΔE. An illustration depicting this process is given in **Figure 5**. The entropy is computed as follows.

$$E = \sum_{i=1}^{w} \sum_{j=1}^{20} -p_{ij} \log p_{ij}, \tag{1}$$

where w is the size of the sliding window and p_{ij} is the frequency of the j-th amino acid at the i-th position incorporating a pseudo count of one. In this chapter, $w = 10$. The predefined threshold is computed as follows.

$$\Delta E = (E_{max} - E_0) / \lambda, \tag{2}$$

where E_{max} is the entropy of a random database of similar composition to S and a similar size to the current set of sequences, and λ is an empirically chosen constant that determines how much the conserved region is extended. In this chapter, $\lambda = 16$.

2.5.3 Motif Refinement

The extended occurrences of the motif are then used to generate a sophisticated statistical model, which is a profile HMM [24][45]. A profile HMM further addresses the problem of the lack of context specificity by fully characterizing the distribution of amino acids for every position in a motif. It is supported by a formal statistical framework [45], which provides a consistent theory for scoring insertions and deletions in the occurrences of the motif as well as for dealing with generalization issues by adding an element of randomization. The package HMMER (http://hmmer.wustl.edu/) is used in this chapter to perform all profile HMM-related operations. A profile HMM is obtained by running HMMBuild on the set of aligned, ungapped, and extended motif occurrences. HMMBuild constructs a profile HMM using the maximum *a posteriori* (MAP) [24] construction algorithm to determine the length of the main model and Dirichlet mixture priors with maximum likelihood estimates to train the parameters of the profile HMM [1][16].

Close estimates of the parameter values of the probability density function for the log-likelihood ratios to be computed by HMMSearch, as described in the next section, are then obtained by running HMMCalibrate on the profile HMM. This probability density function is approximated by an extreme value distribution (EVD) [22][43]. HMMCalibrate scores a large number of synthetic random sequences whose lengths are normally distributed around the average length of a protein with the given profile HMM and fits the resulting score histogram to an EVD.

A profile HMM can be further refined in an iterative fashion. For each profile HMM, H_i, a refined profile HMM, H_{i+1}, can be obtained by matching H_i against the current set of sequences, as described in the next section, and then running HMMBuild on the set of aligned and now possibly gapped matches of H_i. The calibration and matching procedures could then be repeated until $H_{i+1} = H_i$. In this chapter, the statistical model is refined only once, since a bigger difference is expected to lie between the regular expression and the first profile HMM than between any pair of successive profile HMMs.

The use of profile HMMs also takes care of two critical issues. Both address how general or representative the motifs generated from an incomplete training sample of a functional subset

would be with respect to the functional subset. The first issue arises when a motif becomes too specific as it allows "exceptional" residues for a given position that may occur only in a very small number (typically one) of representative sequences. For example, this is the case when a similarity class like [ACV] is used to represent a PROSITE motif, as discussed in [54], since the residues in the similarity class do not generally share chemical or physical properties. The second issue, which is basically the opposite of the previous one, is making a motif too specific because the training set is too small and hence does not contain all possibilities for a position in the motif. One example of addressing this issue in the overfitting context is given in [84].

The first issue is addressed in part by SPLASH, which defines similarity classes based on a substitution matrix so that they correspond only to groups of residues with similar properties. The first issue is further addressed by the use of profile HMMs supported by a formal statistical framework so that the effect of a single exceptional residue at a conserved position would be weighed appropriately. The second issue is also addressed by the use of profile HMMs. A profile HMM characterizes each position of a motif probabilistically, allowing the incorporation of prior information as well as pseudo counts. Prior information generally addresses mutations related to residues with similar properties. Pseudo counts generally reduce the bias resulting from a limited training set. The construction of a profile HMM in this chapter incorporates Dirichlet mixture priors, which provide both appropriate prior information and appropriate pseudo counts. The second issue is further addressed by the iterative refinement of profile HMMs. The first profile HMM constructed is refined into a second profile HMM in this chapter, and this generally further increases the size of the training set used to generate the motif. Overall, the two issues are adequately addressed by the use of profile HMMs. This is evidenced by the small average number of errors produced by the taxonomy constructed by CASTOR with respect to a reference taxonomy, as shown in Section 6.2.

2.5.4 Pattern Matching

The matches of a profile HMM in the set of sequences from which the motif is discovered by SPLASH are obtained by running HMMSearch on the profile HMM against the set of sequences. HMMSearch computes a score in the form of a log-likelihood ratio [24][43] for every sub-sequence matched with the profile HMM. This ratio indicates how statistically significant the match is with respect to a random database of a fixed size but similar composition to the set of sequences under consideration. It also computes an *E*-value from the log-likelihood ratio, which

estimates the number of sub-sequences in the random database that would have equal or greater log-likelihood ratios. A statistical criterion is formed by choosing a constant threshold e_0 such that any sub-sequence with an E-value less than or equal to e_0 is considered as an occurrence of the motif. In this chapter, the chosen value for the threshold is 1/1,000.

Any additional information that may be used to increase the confidence of the matching decisions can be integrated into this matching process. Structural information is one example of such additional information. Proteins can be assigned secondary- or tertiary-structures using a variety of prediction programs, such as PHD [63] and TMHMM [73]. For the GPCR superfamily, for instance, a part of a protein may be assigned with high confidence the cytoplasmic, transmembrane, or extracellular classification in terms of the seven-transmembrane topology [6][81]. To exploit the structural information, for a specific motif, all the occurrences whose structural assignments are in agreement with the structural assignment of the strongest match, as determined by the E-values, are first selected. Among these, those with E-values less than or equal to another rather generous threshold e_3 are further selected as the occurrences of the motif. In this chapter, the chosen value for this threshold is 20/1,000.

2.5.5 Motif Masking

Given a set of sequences of a moderate size, containing a single motif, a pattern discovery algorithm will typically discover a large number of statistically significant motifs. This number increases exponentially as j_0 decreases due to the combinatorial fashion in which slight variants of the motif with smaller supports can be arranged. Therefore, after a motif is discovered and before another one is to be found, CASTOR masks off the first motif by replacing every occurrence of the motif in a sequence by a series of predefined symbol that will be ignored by the pattern discovery algorithm. This effectively prevents the algorithm from rediscovering the same motif and all of its variants repeatedly.

2.6 Class Space Pruning

Suppose that n motifs have been discovered and masked in a set of sequences F at some depth and that $(n+1)$ corresponding classes $\{F_i\}_{i=0}^{n}$ (n positive sets and one negative set) need to be further considered. To avoid inferring the same class repeatedly, pruning of the class space is necessary. The main reason is that if two classes inferred at different breadths overlap, a third child class contained within their intersection might be inferred repeatedly in the inference

structures of the aforementioned two classes. If such repeated instances were to be kept, the sub-inference structures of the repeated instances could be duplicates of each other, leading to duplication of effort and results.

Therefore, each class F_i is examined to determine if it is a subset or a duplicate of another class F_j that has also been inferred from F. Since F_0 (the negative set) by definition consists of the sequences that do not match any of the motifs discovered from F, it cannot overlap any other class inferred from F and thus does not need to be examined for this purpose. Each class F_i should also be examined to determine whether it is a subset or a duplicate of another class $F^=$ that has been inferred but is not an ancestor of F_i. $F^=$ may have been inferred at a larger breadth, since the inference structures of the children of the same class grow independently and thus unpredictably.

Specifically, pruning of class space is performed as follows.

1. If $F \subset F_j$, F_i is discarded and the corresponding motif is discarded and unmasked in all other classes inferred from F.

2. If $F_i = F_j$, F_i is discarded but the corresponding motif is kept.

3. If $F_i \subseteq F^=$, F_i is discarded and the corresponding motif is discarded.

In the first case, it is desirable to eventually have F_i and F_j as well as their corresponding motifs. By pruning as described, the desired result may be achieved by re-discovering the motif defining F_i and thus re-inferring F_i from F_j. In the second case, it is desirable to eventually have the motifs corresponding to F_i and F_j, which will be different, but only necessary to keep one of F_i and F_j, which are identical. In the third case, if $F_i \subset F_k$, it is similar to the first case. However, since F_i and $F^=$ are not inferred from the same class, no unmasking needs to be done and the motif defining F_i is expected to be re-discovered and thus F_i is expected to be re-inferred from $F^=$. If $F_i = F_k$, it is similar to the second case, and the motif defining F_i is also expected to be re-discovered from $F^=$.

If both F_i and the other class were further processed in any of the cases, $\Sigma(F_i)$ would very likely be a subset of either $\Sigma(F_j)$ or $\Sigma(F_k)$. The small uncertainty, which is rooted in the re-discovery of a motif from a different class, stems from the fact that the statistical significance of a motif

depends on the size as well as composition of the set of sequences from which it is discovered. Therefore, which motif is discovered and chosen as the most statistically significant one depends on which sequences and specifically which residues are present in the current set of sequences. It is theoretically possible, then, that the motifs discovered independently from two classes with different paths that contain common sequences may be different. Such an example is shown in **Figure 6**. In practice, however, this is inconsequential, as it tends to happen only in extreme cases. Specifically, a class in $\Sigma(F_i)$ might turn out to be absent in $\Sigma(F_j)$ or $\Sigma(F_k)$ only if the motif defining the class had "borderline" statistical significance, which is usually not the case. If a more stringent approach is desired at the expense of computational efficiency, pruning could be limited only to classes such that satisfy the subset or duplicate relationship while containing exactly the same residues in the common sequences. By pruning as described, however, CASTOR avoids redundant processing without sacrificing the completeness of the results except in truly pathological cases.

3 Structure Reorganization

In any structure described in Section 2, a positive set F is the support set of a corresponding motif M with respect to F_+ from which M is discovered. Since F_+ is a subset of S, F may not be the support set of M with respect to S, which will be referred to as the *full support set* of M with the corresponding set of occurrences referred to as the *full occurrence set of* M. In other words, there may be sequences in S that do not belong to F but nonetheless match M.

This situation, for instance, arises when two motifs are present simultaneously in a set of sequences and are present separately in two other sets of sequences. Suppose that there are a set A where only motif a is present, a set B where only motif b is present, and a set AB where both motifs are present. Suppose further that a is the most statistically significant one in the context of $A+B+AB$. Then it would be discovered first and lead to a split of $A+AB+B$ into two subsets, $A+AB$ and B. The motif discovered next from $A+AB$ would be b. However, b is also present in B, which is not part of $A+AB$, and would likely be discovered from B as b', which is a slight variant of that discovered from $A+AB$. This situation arises frequently in a tree, due to its binary-splitting nature, and much less frequently in other structures. In fact, for the structure $n_g = \infty; n_p = 1$, this situation never arises by definition.

Determining the full support sets of all the discovered motifs may correct some errors in classifying the sequences in S and identifies additional non-hierarchical relationships among distant functional subsets. Identical full support sets could be further merged, since the associated motifs may correspond to functional regions that contribute to protein functions shared by the members in the identical full support sets. In summary, given a structure as described in Section 2, it is desirable to reorganize it into a list of distinct classes, each associated with a list of distinct motifs, such that the classes are the full support sets of the corresponding motifs. In the example discussed earlier in this section, for instance, two classes should be generated, one for a and the other for the merged version of b and b'.

A given structure will be referred to as the *original structure*, and the reorganized structure will be referred to as the *final structure*. The original structure may be reorganized as follows. The full support set F of each discovered motif M, represented as a profile HMM, is obtained by first running HMMSearch on the profile HMM against S, as described in Section 2.5.4. The profile HMM is then refined once, as described in Section 2.5.3, since it is built from a limited set of occurrences and its matches in S should be much more representative of the full occurrence set of M_i. The set of sequences that match the refined profile HMM is taken as the full support set of M_i. In the remainder of this section, the term support (occurrence) set of a motif will be used to refer to its full support (occurrence) set.

All the discovered motifs are then considered one by one so that those motifs that are slight variants of one another are merged into a single *motif group*. This is accomplished by performing single-link clustering where initially, each discovered motif is in a cluster of its own, and two clusters are merged into one as long as two motifs, one from each cluster, are almost identical, which is defined in the next paragraph. Equivalently, it is accomplished by finding connected components in a graph where the nodes represent the discovered motifs and the edges connect two nodes whose corresponding motifs are almost identical. At this point, it is desirable to accommodate several discovered motifs in one motif group such that all of them occur in similar locations with similar composition and overlap one another, but not necessarily every pair of them are almost identical. Furthermore, it is desirable to have a deterministic partition of all the discovered motifs. Therefore, such a procedure is used to assign the discovered motifs into motif groups.

M_i is compared with M_j in terms of their occurrence sets as follows. Each occurrence set can be represented as a matrix, called the *m-matrix*, which is a multiple alignment of all the members in the occurrence set. Therefore, there is one row for every occurrence of the motif and one column for every position in the motif. Deletions in the occurrences are shown as gaps in the m-matrix and insertions are discarded. Conceptually, two motifs can then be compared in terms of their *m*-matrices. They are considered almost identical if their corresponding *m*-matrices substantially overlap. Suppose that s_i is the support of M_i, s_j is the support of M_j, l_i is the length of M_i, and l_j is the length of M_j. Suppose further that s is the number of sequences shared by F_i and F_j, and l is the length of overlap between a pair of occurrences in one shared sequence, which could vary from one shared sequence to the next. This can be visualized in **Figure 7**. Suppose now the following.

$$n_i = s_i - s, n_j = s_j - s, n_{i,1,\max} = \max\left(round\left(t_1 \cdot s_i\right),1\right), n_{j,1,\max} = \max(round(t_1 \cdot s_j),1)$$
$$r_i = l_i - l, r_j = l_j - l, r_{i,\max} = \max\left(round\left(u_1 \cdot l_i\right),1\right), r_{j,\max} = \max(round(u_1 \cdot l_j),1)$$
$$y = u_2 \cdot \min(s_i, s_j)$$

Two *m*-matrices substantially overlap if the following *motif identity constraint* holds.

$$(n_i \le n_{i,1,\max}) \text{ and } (n_j \le n_{j,1,\max}) \tag{3}$$

$$(r_i \le r_{i,\max}) \text{ and } (r_j \le r_{j,\max}) \text{ for at least } y \text{ instances}. \tag{4}$$

where t_1 is a percent coefficient that specifies the *support identity constraint* in (5) in terms of how the *m*-matrices need to overlap row-wise, which indicates how many sequences need to be shared by F_i and F_j; u_1 is a percent coefficients that specifies the *composition identity constraint* in (6) in terms of how the *m*-matrices need to overlap column-wise for a specific pair of rows, which indicates how many residues need to be shared by respective occurrences of the two motifs in one shared sequence; u_2 is a percent coefficient that further specifies the composition identity constraint in terms of how the *m*-matrices need to overlap column-wise over all pairs of rows, which indicates how many pairs of respective occurrences of the two motifs in one shared sequence need to satisfy the composition identity constraint. The max operation allows some tolerance for very small classes or very short occurrences where even a single error would otherwise invalidate the comparison. Here, $t_1 = 0.3$, $u_1 = 0.3$, and $u_2 = 0.5$.

As can be seen in **Figure 7**, there are situations where even if the motifs are not almost identical, they share much of the support as in (b) or composition as in (d). There are also situations where

they may be similar at a borderline as in (c) and the decision of whether the two motifs are almost identical depends on the values chosen for the percent coefficients. In this case, even if the support of one motif is a subset of that of the other, the first motif may contain additional residues that make it much more specific, which makes the two motifs effectively different. In general, each individual case should be analyzed in detail and the use of percent coefficients for the comparison of motifs is aimed at understanding some general properties of the comparison. A more complex comparison algorithm, such as LAMA [57], could also be used, especially if the comparison is not merely to tell whether two motifs are literally identical in that they occur in the same regions of the same sequences, which is the case here.

After all the discovered motifs are merged into appropriate motif groups, a *consensus motif* represented as a profile HMM is constructed for each motif group. The profile HMM is obtained by running HMMBuild on the set of occurrences constructed as follows. For each sequence in the intersection of the support sets of the motifs in the motif group, the subsequence delimited by the smallest starting position of an occurrence in the sequence and the largest ending position of an occurrence is taken as an occurrence of the consensus motif. The profile HMM is then calibrated by running HMMCalibrate on it. The matches of the profile HMM are then identified by running HMMSearch on the profile HMM against S. The profile HMM is then refined once. An *intermediate structure* is then created such that every class in the structure corresponds to the support set of a consensus motif. The intermediate structure will also include a single class that contains all the sequences that do not match any consensus motif.

All the classes in the intermediate structure are then considered one by one so that those that are almost identical are merged into a single *subset group*. This is accomplished by comparing each class F_i that has not been assigned to a subset group with every other class F_j with $j > i$ that has not been assigned to a subset group. If F_i and F_j are almost identical, the latter is assigned to the subset group defined by F_i. F_i and F_j are almost identical if they satisfy the constraint specified in Equation 3. Here, $t_1 = 0.1$. At this point, it is desirable that each subset group consists of highly similar classes only. Therefore, a smaller value for t_1 and a one-pass pairwise comparison rather than an iterative one, as used in the construction of motif groups, are used.

After all the classes are merged into appropriate subset groups, the first class in each motif group is designated as the *consensus class* for each subset group, which is generally appropriate when t_1 is relatively small and thus all the classes in a motif group are all very similar. A final structure is

then created such that every class in the structure corresponds to a consensus class. If a larger value is used for t_1, the number of distinct classes in the final structure becomes smaller.

Even if the original structure may be in the form of a tree, the final structure is most likely in the form of a graph. To provide more information about the final structure, various relationships among the classes may be determined as follows. Suppose further the following.

$$n_{i,2,\max} = \max\left(round\left(t_2 \cdot s_i\right),1\right), n_{j,2,\max} = \max(round(t_2 \cdot s_j),1)$$

$F_i \supseteq F_j$ if the following *support containment constraint* holds.

$$(n_i > n_{i,1,\max}) \text{and} (n_j \le n_{j,1,\max}) \tag{7}$$

F_i is disjoint from F_j if they satisfy the following condition holds.

$$(n_i > n_{i,2,\max}) \text{and} (n_j > n_{j,2,\max}), \tag{8}$$

where t_2 is another percent coefficient. In this chapter, $t_2 = 0.9$. Finally, F_i overlaps F_j otherwise.

In conclusion, structure starts with an initial structure that is constructed by setting n_B to some value and has an initial number of motifs and their support sets as classes. It constructs the full support set for each of the motifs using HMMSearch, merges the motifs into motif groups by performing single-link clustering or constructing connected components, construct a consensus motif for each motif group using HMMBuild and HMMCalibrate, and merges the support sets of the consensus motifs into subset groups by one-pass pairwise comparisons. The time requirement is thus limited to performing profile HMM-related operations for the initial number of motifs and otherwise manipulating them in polynomial time. Therefore, it is generally efficient, for example, compared to constructing another structure with a larger value for n_B.

4 Recursive Graph Model and Structure reorganization

As the value of n_B increases, the output of CASTOR goes from hierarchical to non-hierarchical and thus from simple and concise to complex and complete. Structure reorganization as a post-processing step may be applied to any structure to improve its quality. How the output of CASTOR might change as the value of n_B increases and by structure reorganization is studied specifically via two examples, as shown in **Figure 8**. Each lower-case letter represents a motif that is present in the entire database, where the corresponding upper-case letter again represents the full support set of the motif in **Figure 8** and in the following discussion.

Given a set of classes, which are the support sets of the discovered motifs, a subset of any of them that could be expressed in terms of set operations (intersection, union, difference) could also be considered as a class. For example, in **Figure 8**(a), given *A*, *B*, *C*, and *D*, *A*, *B*, *C*, *D*, *AB*, *AD*, *BD*, *ABD*, and *BC* could be considered as classes. Any combination of *A-B-D*, *AB-D*, *AD-B*, and *ABD* as subsets of *A* could also be considered as classes, and similarly with *B* and *D*. *C-B* and maybe the complement of *A+B+C+D*. These subsets might be the results of complex interactions among functional regions. For example, when ligand binding takes place, two functional regions, which could be described by two separate motifs, might both be involved. In that case, the subgroup of proteins where both functional regions are present, which corresponds to the intersection of the support sets of the two motifs, might be a distinct functional subset. How many and which of the subsets that might reflect the complex interactions among functional regions would depend on the nature of the given group of related proteins. The point is that once the bigger classes are available, the potentially exponentially many smaller classes, which are subsets of the bigger classes, could always be inferred by the appropriate set operations. However, when the smaller classes are available, it is not straightforward to merge them into the bigger classes. In fact, that is exactly what structure reorganization does. Therefore, the number of bigger classes or full support sets that are inferred may be used as the measure of the complexity of output as well as the amount of information conveyed by the output. This is done with the examples shown in the figure, where the full support sets refer to *A*, *B*, *C*, or *D* rather than a subset of any of them.

In each of the two examples shown in the figure, each motif is eventually discovered, even if possibly in terms of a partial support set. Furthermore, no class is inferred repeatedly, and thus no pruning is required. As can be seen, the number of full support sets that are inferred does not necessarily strictly increase as the value of n_B increases, but it never decreases. It depends on whether the most statistically significant motif to be discovered next is present in the support set of a motif that has been discovered. For example, in **Figure 8(a)**, *a*, *b*, *c*, and *d* are supposed to be discovered in that order, from the entire database or whatever class that results from a split. However, the support set of a later one always overlaps that of an earlier one, and so the number of full support sets that are inferred strictly increases. In **Figure 8**(b), however, *B* does not overlap *A*, and so it is inferred even when $n_B = 1$.

As mentioned previously, in each of the two examples, each motif is eventually discovered, even if possibly in terms of a partial support set. When this is the case, structure reorganization is

expected to recover the full support set of every motif that is present in the entire database, as long one of its partial support sets is sufficiently representative of its full support set. Therefore, setting n_B to one can be sufficient when it is followed by structure reorganization.

However, it is possible that the classes that are subsets of the full support set of a motif turn out so small that the motif will not be discovered in any of the parents of these classes. For example, in **Figure 8**(a), when $n_B = 1$, if D is very small and thus its subsets are even smaller, d may not be considered statistically significant in any of A-B (a masked off), AB (a and b masked off), B-A-C (b masked off), and the complement of A+B+C (a, b, and c masked off). When this is the case, structure reorganization cannot recover D. On the other hand, when $n_B = 2$, d may now be considered statistically significant in any of A (a and b masked off) and B (a and b masked off). In fact, the situation could be the other way around in that d is considered statistically significant in some of the appropriate classes when $n_B = 1$ but is not in any of the appropriate classes when $n_B = 2$. Therefore, there is not a clear correlation between the value of n_B and the feasibility of recovering all the full support sets by structure reorganization. It certainly depends on how the full support sets of the motifs overlap.

Therefore, when most of the motifs that are present in the entire database could be discovered, setting n_B to one followed by structure reorganization is expected to result in great performance; otherwise, setting n_B to larger values may be favorable sometimes. Without structure reorganization, which is a complicated process, setting n_B is certainly expected to result in the most accurate and thorough account of evolutionary development and specifically the formation of functional relationships among functional subsets. A significant drawback of the approach, however, is that program execution and output analysis for this structure would be highly inefficient.

It would be reasonable to expect that the structure obtained by setting both n_B and n_D to infinity would provide the most accurate and thorough account of evolutionary development and specifically the formation of functional relationships among functional subsets. A significant drawback of the approach, however, is that data and output analysis for this structure may be computationally prohibitive. The size of a structure would grow polynomially as the value of n_B increases. However, when the exponent is as large as possible, the impact of the resulting variation in the size of the structure could be substantial in practice. The time it takes to construct a structure would be highly related to the size of the structure, with much more time needed for inferring the classes closer to S and thus the bigger ones in the structure. It is also lengthened

more by higher need to prune the subset space. Finally, the analysis especially the visualization of output becomes more cumbersome as the size of the structure becomes larger.

In general, setting n_B to two already is expected to generate significant information about unusual functional relationships among functional subsets. Unless S is relatively small, in which case it could be reasonable to set n_B to a relatively large value, it is advisable to use a small value. However, different instances of the recursive graph model can be constructed by varying the values of the two parameters of the model, and an optimal one can be chosen with the desired tradeoff between output quality and execution efficiency.

5 Binary Tree

The ($n_y = 1, n_n = \infty$) is an important special case. At most two classes (one positive set and one negative set) are inferred from any class, which do not overlap by definition. As a result, the structure is entirely hierarchical and no pruning is required. This has several implications. First, since partially overlapping classes are not possible, only "normal" evolutionary events and processes may be modeled. Therefore, the structure may be thought of as a skeleton for evolutionary development, indicating natural functional relationships among functional subsets. Second, the fact that no pruning is required contributes to high efficiency in program execution. Finally, the set of all the classes corresponding to the leaves of the binary tree is a partition of S. That is, each sequence in S belongs to one and only one class corresponding to a leaf. Since each such class can be uniquely characterized by the set of motifs associated with the classes on its inference path, this structure can be readily used as a decision tree to classify novel proteins.

In an attempt to circumvent the somewhat arbitrary and drastic nature of the decision about a sequence matching or not matching a motif based on a single statistical criterion, some degree of statistical flexibility can be introduced. This can be accomplished by replacing the single criterion with two independent ones, based on two constants e_1 and e_2 with $e_1 \geq e_2$, such that any region with an E-value less than or equal to e_1 is considered as a match of the corresponding profile HMM and any region with an E-value greater than e_2 is considered as a mismatch. This may result in somewhat overlapping classes, as some sequences will be considered both as matches and as mismatches and assigned to both the positive set and the negative set.

This approach will be called *fuzzy clustering* as opposed to the original *exact clustering*. The resulting structure can be re-organized *a posteriori* to form a tree once again, which leads to a partition of S, by considering, for each sequence in S, all the paths leading to the leaves to which

it belongs and by removing all but the optimal one according to some appropriate criterion. The re-organized structure is then also readily used as a decision tree to classify novel proteins.

The binary tree serves as an optimal choice for studying and assessing the performance of CASTOR for the following reasons. First, the binary tree is constructed with high efficiency in program execution because increasingly smaller classes are quickly inferred with no pruning required. As such, it offers an ideal platform for parameter-value space exploration and choice optimization. Second, as discussed in Section 6.1, the reference structure with which the output of CASTOR will be compared for evaluation purposes is a hierarchical structure. Comparing a non-hierarchical structure with a hierarchical one would be tricky and might result in some novel and meaningful non-hierarchical relationships artificially increasing the number of matching errors. Third, as discussed in Section 7, existing automatic systems that perform unsupervised protein classification based on primary structures, including ProtoMap, generate hierarchical structures. Specifically, the output of CASTOR will be compared with that of ProtoMap. Once again, comparing a non-hierarchical structure with a hierarchical one would be inappropriate. For these reasons, although CASTOR is capable of performing different types of clustering by setting n_B and n_D to arbitrary values, most of the parameter space exploration is performed with the binary tree. However, some important results from non-hierarchical clustering on the protein family of interest will be discussed in Section 6.5

6 Results

To assess the performance of CASTOR, the GPCR superfamily is chosen as a test case, as discussed in the next section. Section 6.2 evaluates the system's ability to infer functional subsets. This is accomplished by mapping the classes inferred by CASTOR to a reference set of biological classes. Section 6.3 and Section 6.4 evaluate the system's ability to identify functional regions and residues that play specific functional roles. This is accomplished by mapping the motifs discovered by CASTOR to a reference set of motifs discovered in a supervised manner and by mapping residues covered by those motifs to a reference set of residues for which mutation experiments have been performed. Finally, Section 6.5 studies the effect of structure reorganization and the advantages of inferring unusual functional relationships among functional subsets.

Most experiments in the following sections are performed using the binary tree ($n_B = 1$, $n_D = \infty$). The structure ($n_B = \infty$, $n_D = 1$) and the structure ($n_B = 2$, $n_D = \infty$) are also analyzed in a few cases.

6.1 Sample Protein Superfamily

To evaluate the performance of CASTOR, it is most appropriate to analyze a large, significant, and diverse protein superfamily, where the functions of the majority of member proteins are known *a priori*. The GPCR superfamily is an ideal choice with respect to these requisites. It is one of the largest protein families found in nature. More than 200 distinct receptors in this family have been cloned and more than 1,000 full-length sequences and fragments are available in SWISS-PROT. The superfamily possesses substantial sequence homology among member proteins, which share a well-understood structural topology, consisting of seven transmembrane helices connected by loops. The member proteins play a fundamental role in regulating some key activities in virtually every cell of a complex organism. They act by mediating cellular responses to external stimuli across the cell membrane to an enormous number of signaling molecules. Specially, upon binding with high-specificity with extracellular ligands, these proteins interact with heterotrimeric G-proteins that can then, in their activated forms, inhibit or activate various effector enzymes and/or ion channels. The high selectivity in ligand binding is expected to result in a number of highly selective motifs for the recognition and binding of ligands. Some weak signals may also exist for the coupling with G proteins [40].

The *reference sequence set*, which is used as the complete set of sequences in evaluating the performance of CASTOR, is constructed by first taking all the sequences available from GPCRDB as of early year 2,000. Sequences that are fragments or correspond to orphan proteins are excluded. Those sequences that are not included in PRINTS as of early year 2,000 (PRINTS 25.0) are excluded as well. The resulting reduced database is taken as the reference sequence set. It contains 846 full-length sequences.

Structural information is obtained from the annotations of the corresponding entries in SWISS-PROT.

6.2 Set Mapping

This section describes the steps taken to compare the structure constructed by CASTOR for the GPCR superfamily against a biologically motivated and manually assembled taxonomy.

6.2.1 Preparations and Considerations

To evaluate the performance of CASTOR in terms of the inference of classes, a *reference class set* is required, which defines a set of biological classes and the relationships among them. Unfortunately, there is little agreement on a global taxonomy for the GPCR superfamily, with different sources constructing and reporting different, albeit somewhat overlapping structures. Two of the richest sources are GPCRDB and PRINTS. GPCRDB is a general-purpose database on the GPCR superfamily. It reports a hierarchical taxonomy based mainly on known pharmacological properties of member proteins. PRINTS is a database of fingerprints for various protein families, where a fingerprint is a set of motifs represented as ungapped multiple sequence alignments and it characterizes the corresponding protein family in that the motifs collectively match the member proteins almost exclusively. For the GPCR superfamily, PRINTS reports a hierarchical taxonomy based on biological literature [81] and assembles one fingerprint for every biological class in the taxonomy. Therefore, PRINTS offers both a taxonomy and a comprehensive, hierarchical set of fingerprints for the GPCR superfamily. This will be discussed in detail in Section 6.3.

Since GPCRDB reports several interesting biological classes that are not reported by PRINTS and vice versa, the reference class set is constructed by fusing the information from these two sources as follows, with the aim of maximizing the total number of distinct biological classes and relationships. The two hierarchical lists of biological classes are merged by adopting the finer classification whenever a discrepancy exists between the two, while maintaining consistency with either one whenever possible. The result is a set of non-overlapping biological classes, which cover the entire reference sequence set and will be called the *base biological classes*, as well as a hierarchical set of combinations of the base biological classes, which will be called the *composite biological classes*. The combination of the base and composite biological classes is taken as the reference class set. The biological classes in the reference class set will all be referred to as the *b-classes* for convenience. Only *b*-classes with at least three members are considered in this procedure, and there are 212 such *b*-classes.

6.2.2 Procedure

The classes inferred by CASTOR will be called the *c-classes* for convenience. The set-mapping procedure is as follows. Each *c*-class C is compared against each *b*-class B, and the result is expressed in terms of the number of false positives (members of the *c*-class that do not belong to

the b-class) and that of false negatives (members of the b-class that do not belong to the c-class). These are the standard type I and type II errors. Suppose the following.

$$n_c = |C - B|, n_b = |B - C|, n_{\max} = \max(round(\alpha \cdot |B|), 1)$$

A success of set mapping is declared if the following condition holds.

$$(n_c \le n_{\max}) \text{ and } n_b \le n_{\max}, \tag{9}$$

where α is a coefficient that specifies how strict the set mapping is.

The set-mapping statistic N_m is defined as the number of distinct b-classes for which successes of set mapping are declared. N_m is an intuitive and easy measure of the quality of a structure generated by CASTOR with respect to the reference class set. The variability of N_m as the value of α is varied also offers a clue to the stability of the c-class/b-class mapping with different degrees of tolerance for errors. It is possible that for a b-class, multiple successes of set mapping are declared with different c-classes, or for multiple b-classes, successes of set mapping are declared with the same c-class. However, as shown in Section 7.2.2, where a one-to-one correspondence between the set of b-classes and the set of c-classes is obtained for one experiment, most of the b-classes for which successes of set mapping are declared turn out to be exclusive matches of single c-classes based on appropriate set-mapping criteria.

CASTOR has a number of parameters, the most relevant ones being the following.

- t_0, minimum number of tokens for SPLASH
- k_0 and l_0, minimum number of tokens in a window and window length (density constraint parameters) for SPLASH
- N_s, minimum number of motifs searched for before reporting one
- e_1 and e_2, E-value thresholds on output of HMMSearch

A priori, it is not clear which set of parameter values is likely to produce the best output and, more importantly, how the output quality changes from one set of parameter values to another. It is therefore desirable to explore the parameter-value space and expect to observe the robustness in performance. Other features of CASTOR, such as iterative pattern refinement (IPR), as discussed in Section 2.5.3, use of structural information (SI), as discussed in Section 2.5.4, and fuzzy

clustering, as discussed in Section 0, are also evaluated. A total of 28 experiments are performed for three relatively small values of α.

6.2.3 Results and Analysis

The results of set mapping are summarized in **Table 1**. Only c-classes with at least three members are considered in this procedure, and there are from 322 to 431 such c-classes in the binary trees from the first 23 experiments. In the following discussion, the experiments will be referred to by their indices enclosed in parentheses. Some reasonable trends emerge as follows.

- As shown in (1 – 5), results tend to improve with larger values of t_0. This is reasonable because SPLASH motifs that are more specific tend to reduce the number of false positives with respect to a biological class that are used to build and calibrate the corresponding profile HMMs. However, the results deteriorate marginally when the value of t_0 goes from 10 to 12. It is possible that there is a value for t_0 that best describes the motifs present in GPCRs, in which case larges values would only increase the number of false negatives with respect to a biological class.

- As shown in (6 – 8), results first improve rapidly with a looser density constraint and then become rather stable with even looser density constraints. This is reasonable because motifs that correspond to functional regions are rarely very sparse. For instance PROSITE reports only one instance of a regular expression with more than 17 consecutive don't cares. Meanwhile, using a very loose density constraint increases the chances of detecting random motifs. Therefore, as long as the density constraint is reasonably tight, the system performs well, especially since both l_0 and $2l_0$ are used in each run as discussed in Section 2.5.1.

- As shown in (9 – 13), results first improve with larger values of N_s and then deteriorate with even larges values of N_s. This is intuitively correct because the statistical significance of a motif is based on its support as well as its composition. For instance, a motif that has a smaller support may be longer and have a higher statistical significance. Therefore, using a relatively large value for N_s ensures that a sufficiently large pool of motifs, including those with moderately large supports, is discovered before the most statistically significant one is chosen. However, since pattern discovery is performed from the top down, a motif that is more general or present in more sequences is generally preferred to one that is more specific. It is therefore possible that excessively large values of N_s result in the discovery of those motifs that are not general enough to lead to the accurate top-down classification of proteins.

- Results also improve with smaller values of e_1 and e_2, as shown in (3, 14, 15) as well as in (6, 16, 17). Larger E-value thresholds tend to dramatically increase the number of false positives. Furthermore, motifs for the GPCR superfamily seem highly specific in general, leading to better matching scores.

- Results improve significantly with iterative pattern refinement, as shown in (3, 22), and improve even further with the incorporation of structural information, as shown in (22, 23).

- Results improve only moderately with fuzzy clustering as compared to exact clustering, as shown in (3, 24) as well as in (18, 25).

The structure ($n_B = \infty$, $n_D = 1$) contains 207 c-classes with at least three members, which is a much smaller number than that in an average binary tree. This is expected since no recursion is performed. It thus results in much fewer successes of set mapping, as shown in (26).

The structure ($n_B = 2$, $n_D = \infty$) contains 427 c-classes with at least three members, which is slightly more than that in an average binary tree. The set-mapping results for this structure appear satisfactory although they compare slightly less favorably with those for the best binary tree, as shown in (27). This is possible because many classes that could be inferred by this approach are only shown implicitly through the relationships among those that are inferred. This situation may be observed, for instance, when two motifs are present simultaneously in a set of sequences and are present separately in two other sets of sequences. Suppose again that there are a set A where only motif a is present, a set B where only motif b is present, and a set AB where both motifs are present. Suppose further that a is the most statistically significant one in the context of A+B+AB. Then the tree approach would infer four c-classes ($A \cup AB$, B, A) while the graph approach would infer only two ($A \cup AB$ and $B \cup AB$). The third class where both a and b are present would therefore be indicated only implicitly through the overlap of the two classes and the corresponding undirected edge in the graph.

The structure obtained by structure reorganization contains 450 c-classes that are positive sets and have at least three members. The set-mapping results for the set of positive sets are poor, as shown in (28). This is also possible because many classes that could be inferred by this approach are only shown implicitly through the relationships among those that are inferred. However, six of the 62 c-classes that are considered identical to 66 b-classes are not inferred in (23), which corresponds to the tree from which this graph structure is derived. On the other hand, 374 or about 82% of these c-classes are considered to neither identical nor containing (and being

contained by) *b*-classes. Furthermore, about 95% of the non-disjoint relationships between pairs of *c*-classes are overlaps. This strongly suggests that hierarchical structures are significantly different from non-hierarchical ones, and a purely hierarchical taxonomy may not be sufficiently informative of the relationships formed throughout evolution within a complex protein superfamily.

Overall, the results from (23) seem to be superior to all the others, leading to successes of set mapping for between 124 and 151 out of 212 biological classes, or for between 58% to 71% of the biological classes, depending on the value of α. The results from (29) will be further studied and a detailed comparison between the results from (23) and those from (29) will be performed in Section 7.2.

6.3 Motif Mapping

This section describes the steps taken to compare the motifs discovered by CASTOR from the GPCR superfamily with a biologically motivated and computationally assembled set of motifs.

6.3.1 Preparations and Considerations

To analyze the performance of CASTOR in terms of the identification of functional regions, a *reference motif set* is required. The reference motif set is constructed by taking the set of all the fingerprints from PRINTS reported for the GPCR superfamily and subgroups of the superfamily as of early year 2,000 (PRINTS 25.0). Those motifs in the fingerprints generated from sequences that are not in the reference sequence set are excluded. The resulting motif set is used as the reference motif set. The motifs in the reference motif set will be referred to as the *p-motifs* for convenience. Only *p*-motifs that are supported by at least three sequences are considered in this procedure, and there are 1,088 such *p*-motifs.

There are several key differences in the generation of motifs between PRINTS and CASTOR: (a) PRINTS generates each fingerprint for a predefined biological class, where every motif in the fingerprint is present in all the member proteins. Furthermore, it does so for all the biological classes in no particular order. Meanwhile, CASTOR discovers motifs from S that are present in various subsets of S in an unsupervised manner. Furthermore, the set of motifs defining a class would not be discovered until those defining its ancestor have been discovered. (b) As an implication of (a), PRINTS tends to generate motifs that substantially overlap, since it may generate for a biological class a motif that borrows part of a motif that is meant to be part of the

fingerprint for one of its ancestors. In contrast, CASTOR discovers a motif and masks it off, thereby eliminating the possibility of discovering overlapping motifs. (c) PRINTS focuses on ungapped regions in generating a fingerprint, while CASTOR discovers motifs that account for insertions and deletions and have higher tolerance of unconserved internal positions. As a result of (b) and (c), PRINTS tends to generate shorter and more motifs than CASTOR.

6.3.2 Procedure

The motifs discovered by CASTOR will be called the *c-motifs* for convenience. The motif mapping procedure is as follows. Each *c*-motif is compared with each *p*-motif as described in Section 3. Specifically, a success of motif mapping is declared if their corresponding *m*-matrices substantially overlap, which means that the conditions specified in Equation 3 and Equation 4 hold. Here, a range of values is used for t_l (0.1 through 0.3) as well as for u_l (0.1 through 0.7) so that the variation in the results with these two parameters may be observed. A larger range is used for u_l to account for the differences in length between the *p*-motifs and the *c*-motifs.

The motif-mapping statistic N_d gives the number of motifs generated by one system for which successes of motif mapping are declared. The results from experiment 23 are analyzed in detail in the next section.

6.3.3 Results and Analysis

The results of motif mapping are summarized in **Figure 9**. Only *c*-motifs supported by at least three sequences are considered in this procedure, and there are 378 such *c*-motifs for this experiment. As can be seen in **Figure 9**(a), as the value of t_l increases, the value of N_d increases at a very slow rate. As the value of u_l increases, the value of N_d increases steadily. Notice that the value of N_d for CASTOR either is equal to or falls short of that for PRINTS. In most of the cases when the latter happens, two *p*-motifs are mapped to the same *c*-motif. This reflects the fact that PRINTS tends to have more and shorter motifs, which results in multiple, adjacent *p*-motifs corresponding to a single *c*-motif. Furthermore, it is possible that a biological class corresponds to multiple classes defined by some *c*-motifs, since the output of experiment 23 is a tree. It is then possible that these *c*-motifs correspond to a *p*-motif that characterizes the biological class, but the support identity constraint cannot be satisfied and thus no success of motif mapping can be declared.

As can be seen in **Figure 9**(b), a much higher percentage of the *c*-motifs are mapped to the *p*-motifs, since there are many fewer *c*-motifs. The tendency for multiple adjacent *p*-motifs to correspond to a single *c*-motif also makes it possible that in such a case, each of the *p*-motifs is so short that the composition identity constraint cannot be satisfied and thus no success of motif mapping can be declared. This is illustrated in **Figure 10** and **Figure 11**. Meanwhile, it is also likely that PRINTS generates some *p*-motifs artificially for subsets of a functional subset such that the corresponding regions are generally conserved across the entire functional subset. This is also illustrated in **Figure 10**.

For a biological class in the taxonomy used by PRINTS that is also a class inferred by CASTOR, it generally happens that most of the *p*-motifs in its fingerprint have corresponding *c*-motifs, but not vice versa. In other words, CASTOR may identify more previously unknown functional regions from a given protein family. This is illustrated in **Figure 11**.

6.4 Residue Mapping

An important area of research is the identification of binding sites via site-directed mutagenesis assays. These tend to be lengthy, complex, and expensive experiments. Given that the motifs discovered by CASTOR lead to a taxonomy that highly agrees with a biologically motivated and manually assembled taxonomy, it is reasonable to hypothesize that the conserved regions where these motifs occur are where the residues that play functional roles tend to cluster. Such residues will be referred to as *functional residues*. This section describes the steps taken to test this hypothesis by showing how to map functional residues onto the conserved regions. It is shown that surprisingly, those residues for which mutation experiments have been reported with visible functional effects tend to distribute with almost identical probabilities in the conserved regions as well as in the rest of the proteins.

6.4.1 Preparations and Considerations

To analyze the performance of CASTOR in terms of the identification of functional residues, a *reference residue set* is required. GPCRMD is a compilation of published mutation experiments conducted on members of the rhodopsin family and published up to year 1997. TGrap is another compilation of published mutation experiments conducted on members of all the five major families and published up to April, 2001. For each experiment, either compilation records the sequence, the mutated residue(s), the observed functional effect(s) of the mutation, and the publication that reports the full results. The reference residue set is constructed by first taking all

the entries corresponding to single-substitutions and deletions in both compilations. Those entries involving residues from sequences that are not in the reference sequence set and those corresponding to mutation experiments with no observable functional effects are excluded. The remaining entries are then checked for accuracy against the reference sequence set, such as whether the specified residue resides in the specified location in the sequence of the specified species. Entries that fail the check are either corrected or removed depending on the availability of additional information. The resulting database of entries with one residue per entry is taken as the reference residue set. The residues in the reference residue set will be referred to as the *g-residues* for convenience, and there are 336 of them from 99 sequences.

6.4.2 Procedure

The residue mapping procedure is as follows. For each *c*-motif represented as a profile HMM, its support set with respect to the set of 99 sequences is obtained by running HMMSearch on the profile HMM against the set of 99 sequences, as described in Section 2.5.4. For each of the matches, it is determined whether it contains a *g*-residue. If it does, a success of residue mapping is declared. Even if all the 99 sequences are in the reference sequence set, it is possible that a g-residue is not already in the support set of the c-motif and yet a success of residue mapping could be declared for the g-residue, since the support set may not be the full support set of the c-motif.

The first residue-mapping statistic N_{r1} gives the number of distinct *g*-residues for which successes of residue mapping are declared. Under the assumption that the mutation experiments with visible effects involve functional residues (while the others involve non-functional residues) and they represent a random sample of all the functional residues, this statistic is used as a measure of the distribution of the functional residues with respect to the conserved regions, where the *c*-motifs occur. To investigate whether this exhibits any bias towards the conserved regions, the appropriate sequence coverage of the *c*-motifs also needs to be computed. In this case, this is the ratio of the total length of protein regions covered by the *c*-motifs to the total length of protein sequences in the reference sequence set.

The second residue-mapping static N_{r2} is computed in the same way subject to the following constraint. Each *g*-residue is from a sequence that belongs to exactly one class C that corresponds to a leaf L in the binary tree. For each *g*-residue, only the successes of residue mapping resulting from the matching of the *c*-motifs that are associated with the nodes along the inference path of L count. This statistic restricts the hunt for functional residues for each

sequence r to only those motifs such that r belongs to each of their support sets in the binary tree. In this case, the appropriate sequence coverage is the average ratio of the total length of protein regions covered by the appropriate c-motifs in all the sequences that belong to C to the total length of protein sequences that belong to C, over all the g-residues.

The results from experiment 23 are analyzed in detail in the next section.

6.4.3 Results and Analysis

The value of N_{s1} turns out to be 232, about 69% of the number of all the g-residues. This is a significant majority. However, the c-motifs cover about 71% of all the protein regions. Therefore, 31% of the g-residues are contained in about 29% of the protein regions where no c-motifs occur. The value of N_{s2} further supports this surprising result. The value of N_{s2} is computed and reported in an accumulated fashion, starting with the consideration of c-motifs associated with the nodes discovered up to the first level. The results are shown in **Table 2**.

The results show quite clearly that there is no significant association of the functional residues with the conserved regions, some of which may correspond to active sites while some others may play other functional roles. This can arise from three possibilities. (a) Mutations at critical protein regions, even outside conserved regions, can lead to measurable functional changes. For examples, cysteins often play structural roles without residing in a conserved region. It is also possible that some range of substitutions may be tolerated, wide enough that it shows up as basically a don't care, but an extreme substitution or deletion might disrupt folding or the like. (b) Reported levels of functional changes span a relatively large interval and some of them may be too small to be considered truly significant. It is possible that if the same analysis were to be performed only for those residues whose mutations lead to catastrophic functional effects, there would be clear bias to conserved regions. This analysis requires considerable manual efforts and will be the subject of a follow-up investigation.

6.5 Graph Construction and Structure reorganization

As discussed in Section 2.1, a tree may fail to represent some complex, non-hierarchical relationships among functional subsets. It may also lead to classification problems because it does not allow the recovery from a misclassification. Graph construction may alleviate these problems, and structure reorganization may additionally improve these situations. These are illustrated with some real examples in **Figure 12** and **Figure 13**.

Consider the set of proteins *T* that consists of the majority of Beta Adrenoceptors (except for B3AR and B4AR) and New Composite Group 2, which are two rather different biological classes. In **Figure 12**, members of *T* are shown in red. In the graph generated from experiment 27, a very specific motif (`RiAqkQirrIdslErrfe`) is supported unambiguously and exactly by *T*, which thus links the two biological classes that would not be otherwise related. In the tree generated from experiment 23, this subtle relationship does not reveal itself. This is because *T* is split early in the tree, and the resulting classes can no longer be related to one another. An example of such a split is due to another motif (`srkklslarEkKat`), which separates the B1AR sequences from all others in *T*.

As can be seen in **Figure 13**, in the tree generated from experiment 23, the biological class GLYCHORMONER is split into two at an early stage. The reason is that the motif (`ipPlfGWsryvpeglltsCgidyytd`) is deemed more statistically significant than any other motifs, including any present exclusively in GLYCHORMONER, but matches only some members of GLYCHORMONER. However, deeper in the hierarchy, as highlighted in the figure, the two subsets of GLYCHORMONER end up being two classes of their own. Each of these two classes is associated with a list of motifs. As could be expected, the two lists of motifs have strong similarities. For instance, the two motifs, (`FAisAAFKvPLitvtnsKvLLvL`) from one of the lists and (`vpLitvtnskiLLvL`) from the other, are very similar. Therefore, as structure reorganization is performed as described in Section 3, when either motif is matched against members of the other class, strong matches emerge. These motifs are thus merged into one motif group and associated with one class that is identical to GLYCHORMONER. Two other motifs, (vsnYmKvsiClPmDvEstLsqvYiLsi) from one of the lists and (vssymkvSiclpmDietpLsqaYil) from the other, are also quite similar. These are then merged into another motif group and associated with another class that is identical to GLYCHORMONER. Finally, the two identical classes are merged into one subset group associated with all the corresponding consensus motifs.

7 Related Work

In this section, CASTOR is compared with other systems that accomplish similar tasks in various aspects.

7.1 General Comparison with Other Systems

There exist a number of other sequence-based, automatic protein classification systems of varying scope and emphasis. CASTOR uses an unsupervised learning approach. COG [77], DOMO [24] [33], HHS/MST [74], ProDom [72], and ProtoMap [86] are among those that also use an unsupervised learning approach. CASTOR uses a top-down or divisive approach via pattern discovery, refinement, and matching such that all members in a class are related in a consistent way by matching a common motif. All of these other systems, however, use a bottom-up or agglomerate approach via pairwise local alignments using various linkage rules, where members in the same class may be related indirectly through transitivity. Meanwhile, while our system naturally generates a complete classification for a given group of related proteins, all of these other systems except for ProtoMap focus on subgroups of highly related proteins and only generate classes at the bottom level of a classification. In other words, they automatically stop constructing larger classes by merging existing smaller ones when the resulting classes no longer contain only proteins that are highly related to one another. The system described in [62] also uses an unsupervised learning approach via pattern discovery and matching in building a dictionary of motifs. However, it again focuses on subgroups of highly related proteins and does not attempt to build identify the complex relationships that may exist among the various subgroups of proteins.

CASTOR identifies local and conserved regions in protein sequences such that all members in one class share the same set of conserved regions in principle. This is done in the hope that such conserved regions correspond to functional regions or particular protein domains. However, CASTOR does not attempt to specifically delimit the potential protein domains and thus recognize their exact boundaries and locations. In this sense, CASTOR as well as PRINTS and BLOCKS attempt to identify "partial protein domains" such that all the proteins in one class share identical potential domains. On the other hand, DOMO, SBASE [59], and ProDom are among those that attempt to identify complete protein domains, where not necessarily all the proteins in one class share identical potential domains. These other systems tend to assign proteins into one class such that they are related globally, where the global relationship may be formed by transitivity of sharing individual protein domains.

CASTOR uses profile HMMs to represent local and conserved protein regions, which are first efficiently identified by discovering motifs represented by regular expressions from full-length

protein sequences. This allows for a combination of statistical soundness and sensitivity of the profile HMM techniques and the efficiency of deterministic discovery of rigid patterns. Pfam (http://pfam.wustl.edu/) is the only other system that uses profile HMMs to represent conserved protein regions. Unlike other model-based systems, however, it deals with full-length protein sequences rather than merely local protein regions. This results in a larger computational load in the training or model-building phase.

7.2 Performance Comparison with ProtoMap

As discussed above, ProtoMap appears to be the only sequence-based, automatic, unsupervised protein classification system that is capable of generating a taxonomy for a group of proteins as complex as the GPCR superfamily. Thus, it serves as a logical choice for a comparative performance analysis. It is out of the scope of this chapter to perform an exhaustive performance comparison considering all sorts of existing systems. The performance comparison with ProtoMap, however, yields a variety of interesting and generalizable observations.

7.2.1 Procedure

The biological classes in the reference class set are called the *b*-classes and the classes inferred by CASTOR are called the *c*-classes, as mentioned in Section 6.2. The classes inferred by ProtoMap will be referred to as the *p*-classes for convenience as well. The performance comparison procedure is as follows. Take CASTOR for example. Each *b*-class is compared with each of the *c*-classes. For each *b*-class, the *c*-classes corresponding to the smallest total number of errors, the sum of the number of false positives and that of false negatives, are identified. These are called *best-matching classes* for the *b*-class. The same process is repeated with the roles of *b*-classes and those of *c*-classes reversed, and the best-matching classes for each *c-class* are obtained. A list of best matches is then established such that there is a best match involving a *b*-class and a *c*-class if and only if they are best-matching classes for each other. A list of best matches can be established in this way for ProtoMap as well.

The best-matching relationship is one-to-one for the *c(p)*-classes and the *b*-classes that are involved in the best matches. That is, for CASTOR, no two *b*-classes are best-matching classes for a *c*-class, which is also a best-matching class for these two *b*-classes, and vice versa. The same holds for ProtoMap. In general, a best-matching relationship is not necessarily one-to-one. If it is not, an algorithm that determines the maximum bipartite matching [20] may be used to generate a final list of best matches such that the best-matching relationship one-to-one. For each

entry in a list of best matches, the ratio of the corresponding smallest total number of errors to the size of the corresponding b-class is computed. This is called the *fitness score* of the entry.

The first comparison-performance statistic N_b gives the ratio of entries in a list of best matches with a given fitness score to the total number of b-classes. The second statistic N_c gives the ratio of entries in a list of best matches with a given fitness score to the total number of c-classes (or p-classes). The results from experiment 23 are used in this procedure. There are 212 b-classes with at least three members, as mentioned in Section 6.2.1. There are 379 c-classes and 189 p-classes with at least three members.

7.2.2 Analysis

The results of the performance comparison are shown in **Figure 14**, **Figure 15**, and **Figure 16**. For CASTOR, the list of best matches contains 164 entries, which corresponds to about 77% of all the b-classes and 42% of all the c-classes inferred by CASTOR. For ProtoMap, the list contains 114 entries, which corresponds to about 54% of all the b-classes and 60% of all the p-classes inferred by ProtoMap. As can be seen in **Figure 14**, the smallest total number of errors is generally relatively small. For both CASTOR and ProtoMap, over 80% of the entries have fitness scores that are less than 0.3. Furthermore, for CASTOR, 137 of the matches satisfy Equation 7 in Section 6.2.2. This number is more than 90% of the value of N_m when $\alpha = 0.3$ for experiment 23 as shown in **Table 1**. It therefore confirms the claim that most of the b-classes for which successes of set mapping are declared turn out to be exclusive matches of single c-classes based on appropriate set-mapping criteria, as discussed in Section 6.2.2.

The results of the performance comparison may be clearly visualized in **Figure 15** and **Figure 16**. As a summary, there are 91 yellow nodes, 25 blue nodes, 72 red nodes, 23 green nodes, and 81 gray nodes. The number of red nodes substantially exceeds the number of green nodes. It is interesting to note that in a few cases, such as for Opsins and for Amine, ProtoMap recovers rather large b-classes, while CASTOR reconstructs their fine-grain subdivisions with high accuracy. The underlying reason may be that the GPCR superfamily possesses high sequence homology overall, which causes the segregation of function to become clear only when the extent of sequence homology becomes relatively low. CASTOR does not stick to the notion of sequence similarity but lends it self particularly to discovering motifs that describe functional regions. Therefore, it may split a large b-class early in the classification process while it identifies potential functional regions that may not necessarily lead to simple, hierarchical

relationships among *b*-classes, thereby compromising the chance to recover it in a single *c*-class. Meanwhile, ProtoMap groups proteins based on pairwise sequence similarity. As a result, it is better at recovering larger, looser *b*-classes with members that are "closer" from an evolutionary perspective. However, it appears much worse at recovering smaller biological classes, where excessive sequence conservation may not allow the system to detect the functional regions in a bottom-up way. This suggests that a combination of the two approaches may be useful.

Another interesting observation is that most of the blue nodes, where both CASTOR and ProtoMap fail to reconstruct a *b*-class, are large. This hints at the fact that some of the parameters used by biologists to group proteins into some of the larger biological classes cannot be interpreted in light of sequence conservation. However, with almost no exception, those nodes with more than two sequences that are descendants of the blue nodes were reconstructed by CASTOR and to a significantly lesser degree by ProtoMap.

8 Conclusion

It has been shown that CASTOR, based on a combination of regular expression-based pattern discovery, pattern discovery statistics, profile HMM-based pattern refinement, and pattern matching, can efficiently and accurately identify functional regions and infer functional subsets in a top-down and recursive manner. It has been further shown by running CASTOR on the GPCR superfamily as an example that the system is well behaved with respect to the choice of parameters and that results are in remarkable agreement with manually constructed taxonomies.

An important feature of the system is that it enables the identification of non-hierarchical functional relationships. It has been shown that such functional relationships are incredibly rich, suggesting that the complexity of the evolutionary development leading to the segregation of function may not be easily captured in traditional hierarchical taxonomies.

Furthermore, it has been shown that conserved regions in protein sequences do not necessarily contain a high concentration of functional residues. This surprising result will naturally lead to further investigation.

Finally, it has been shown that the tree approach with an additional step, dubbed structure reorganization, can construct a significant number of non-hierarchical relationships in an efficient manner. The efficiency of this extra step lends itself to the analysis of very large protein sets, including full databases such as SWISS-PROT.

Some sample output of CASTOR is available at http://www.research.ibm.com/splash/gpcr/. It includes the list of discovered motifs, the exact 1D coordinates of and the secondary-structure information about each occurrence for each discovered motif, and the list of inferred classes. The system will be accessible on-line in the near future.

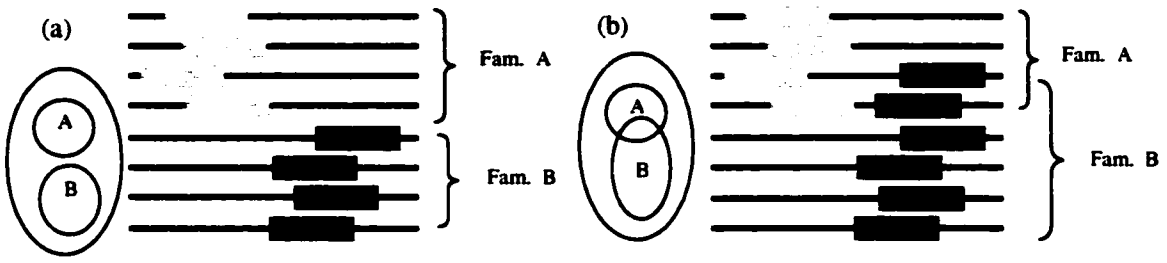


Figure 1: Motif Presence and Family Formation. Each line represents a protein sequence and each block of a given color represents an occurrence of a specific motif. The location of a block corresponds to the location of the occurrence within the sequence. In (a), the first four sequences belong to the family A and the last four to the family B. In this case, the motifs perfectly co-segregate with function, resulting in a hierarchical structure. In (b), members of distinct families may match some of the motifs simultaneously. The two additional occurrences in sequences 3 and 4 may be the result of a gene transfer event. In this case, a hierarchical structure cannot fully describe the underlying complexity. Venn diagrams are useful to describe the two cases and are shown on the left of the lines.

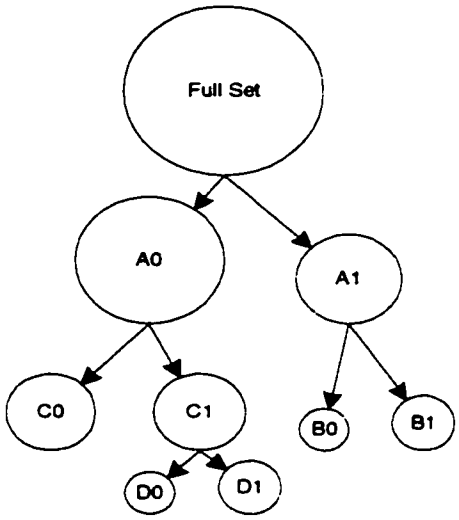


Figure 2: Binary Tree. Each circle (node) represents a protein subset, with its diameter proportional to the size of the subset. A pattern X is discovered from each protein subset, and it is used to divide the subset into two subsets, one being its support set X1 and the other being the complement of its support set X0. Both subsets are further processed, and such recursive processing is continued as long as still possible and desirable.

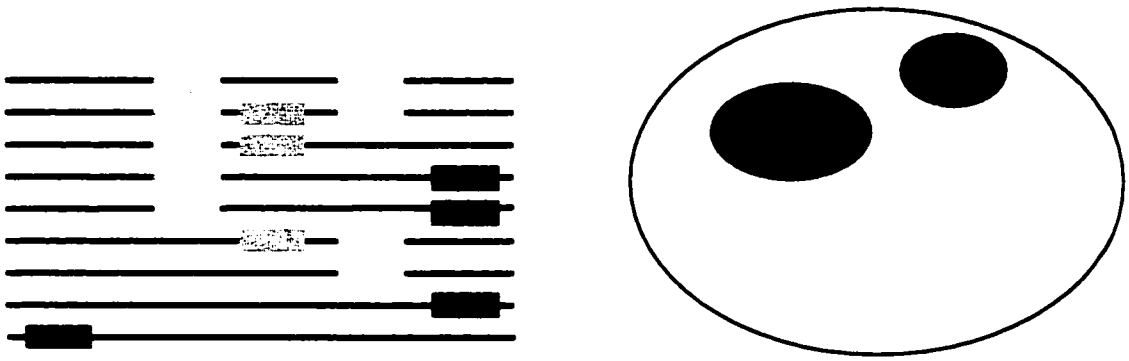


Figure 3: One-Level List. Given a group of related protein sequences, all statistically significant motifs are discovered from the set, each one defining a corresponding subset. An additional subset that is the complement of the union of these subsets can also be defined. A Venn diagram is again useful to depict subset relationships and is shown on the right.

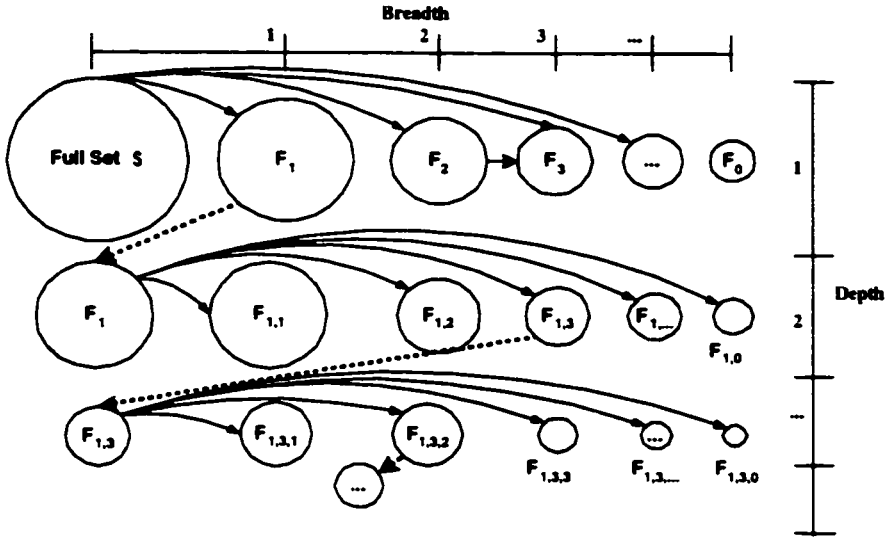


Figure 4: Recursive Graph Model. Each circle represents a class. Each row corresponds to a specific depth and each column corresponds to a specific breadth. Solid arrows represent the inference of children, and dotted arrows simply indicate the advancement to the next recursive level. Only one of the possible recursive discovery paths at each breadth is shown. In reality, each class is further explored, including each of the negative sets (ending in 0).

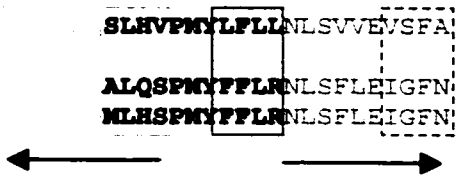


Figure 5: Motif Extension. Bold characters represent the aligned occurrences of a pattern. Orange and brown characters show the left- and right-flanking region, respectively. Rectangles indicate different positions of the sliding window (from solid to dotted).

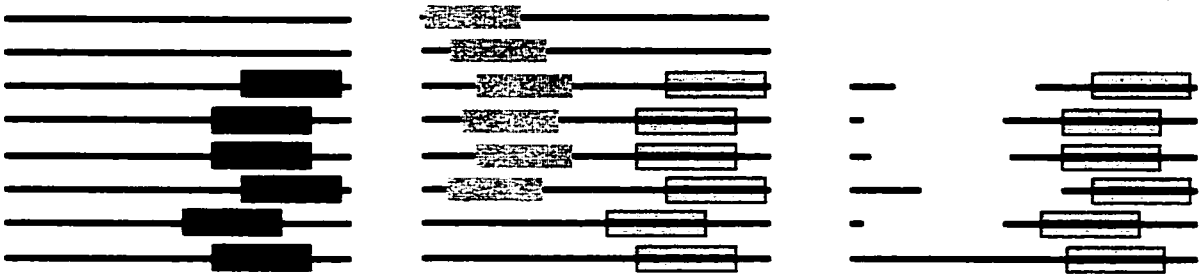


Figure 6: Motif Masking and Residue-Level Composition. Horizontally aligned lines represent the same protein sequence in three sets. In (a), there is originally a sequence set with all the residues available. The statistical significance of a pattern is computed with respect to all the residues available. The red pattern is then discovered. In (b), the red pattern is masked off and the statistical significance of a pattern is computed without considering residues covered by the red pattern. It could therefore be different from the statistical significance of an equivalent pattern discovered in (a). The orange pattern is then discovered. This prevents the discovery of the yellow one. However, it is possible that the sequence set in (c) is inferred along a different path, with the red pattern masked. The yellow pattern would then be discovered next, thereby preventing the discovery of the orange one. If this sequence set is eliminated, however, because it is contained in the sequence set in (b), the yellow pattern would not be discovered. Such cases, however, occur rarely.

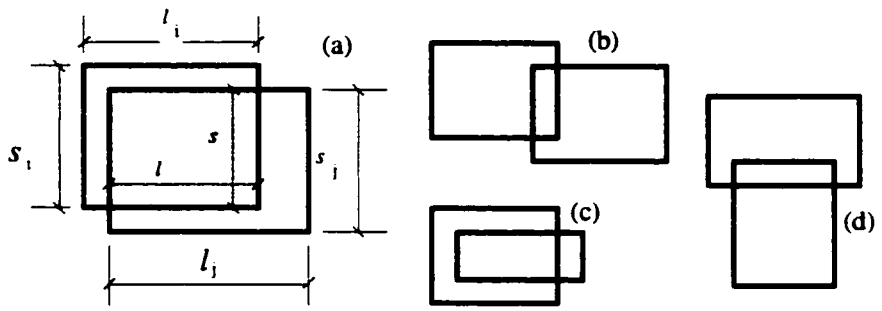


Figure 7: Overlap of *M*-Matrices. Each rectangle represents an m-matrix, where a row corresponds to an occurrence of the corresponding motif, and a column corresponds to a position in the motif. In (a) – (d), different ways of how m-matrices overlap are shown. Not all of them correspond to almost identical motifs.

Table 1: Exploration of Parameter Space. Row shading is for grouping purposes. Parameter value space is explored in Exp. 1 through Exp. 21. Iterative pattern refinement (IPR) is studied in Exp. 3 and Exp. 22, use of structural information (SI) is studied in Exp. 22 and Exp. 23, and fuzzy clustering is studied in Exp. 3 and Exp. 24 as well as Exp. 18 and Exp. 25. Structures other than the binary tree are studied in Exp. 26 ($n_B = \infty$, $n_D = 1$) and Exp. 27 ($n_B = 2$, $n_D = \infty$). Structure reorganization is studied in Exp. 23 and Exp. 28. Finally, the results obtained by running ProtoMap on the reference sequence set are studied in Exp. 29. See text for discussion.

Index	k_o, l_o	t_o	N_s	e_1, e_2	N_m, α = 0.1, 0.2, 0.3
1	4, 8	4	3	1, 1	45, 50, 65
2	4, 8	6	3	1, 1	93, 102, 113
3	4, 8	8	3	1, 1	106, 117, 131
4	4, 8	10	3	1, 1	110, 117, 135
5	4, 8	12	3	1, 1	109, 115, 132
6	4, 12	8	3	1, 1	112, 118, 132
7	6, 12	8	3	1, 1	114, 119, 136
8	8, 12	8	3	1, 1	98, 105, 125
9	4, 8	8	1	1, 1	102, 112, 125
10	4, 8	8	10	1, 1	111, 124, 140
11	4, 8	8	20	1, 1	120, 127, 148
12	4, 8	8	50	1, 1	91, 99, 113
13	4, 8	8	100	1, 1	86, 95, 109
14	4, 8	8	3	10, 10	100, 108, 126
15	4, 8	8	3	0.1, 0.1	116, 123, 141
16	4, 12	8	3	10, 10	100, 108, 130
17	4, 12	8	3	0.1, 0.1	112, 122, 135
18	4, 8	8	20	0.1, 0.1	121, 130, 147
19	4, 12	6	3	10, 10	92, 101, 130
20	4, 12	8	10	10, 10	101, 106, 119
21	4, 12	8	10	0.1, 0.1	109, 121, 139
With IPR					
22	4, 8	8	3	1, 1	115, 128, 143
With IPR and SI					
23	4, 8	8	3	N/A	124, 134, 151
Fuzzy Clustering					
24	4, 8	8	3	0.1, 10	110, 116, 134
25	4, 8	8	20	0.05, 5	124, 131, 149
$n_B = \infty$, $n_D = 1$ with IPR					
26	4, 8	8	3	1, 1	34, 43, 52
$n_B = 2$, $n_D = \infty$ with IPR and SI					
27	4, 8	8	3	N/A	108, 115, 136
With IPR and SI followed by Structure reorganization					
28	4, 8	8	3	N/A	51, 55, 66
ProtoMap					
29	N/A				98, 101, 108

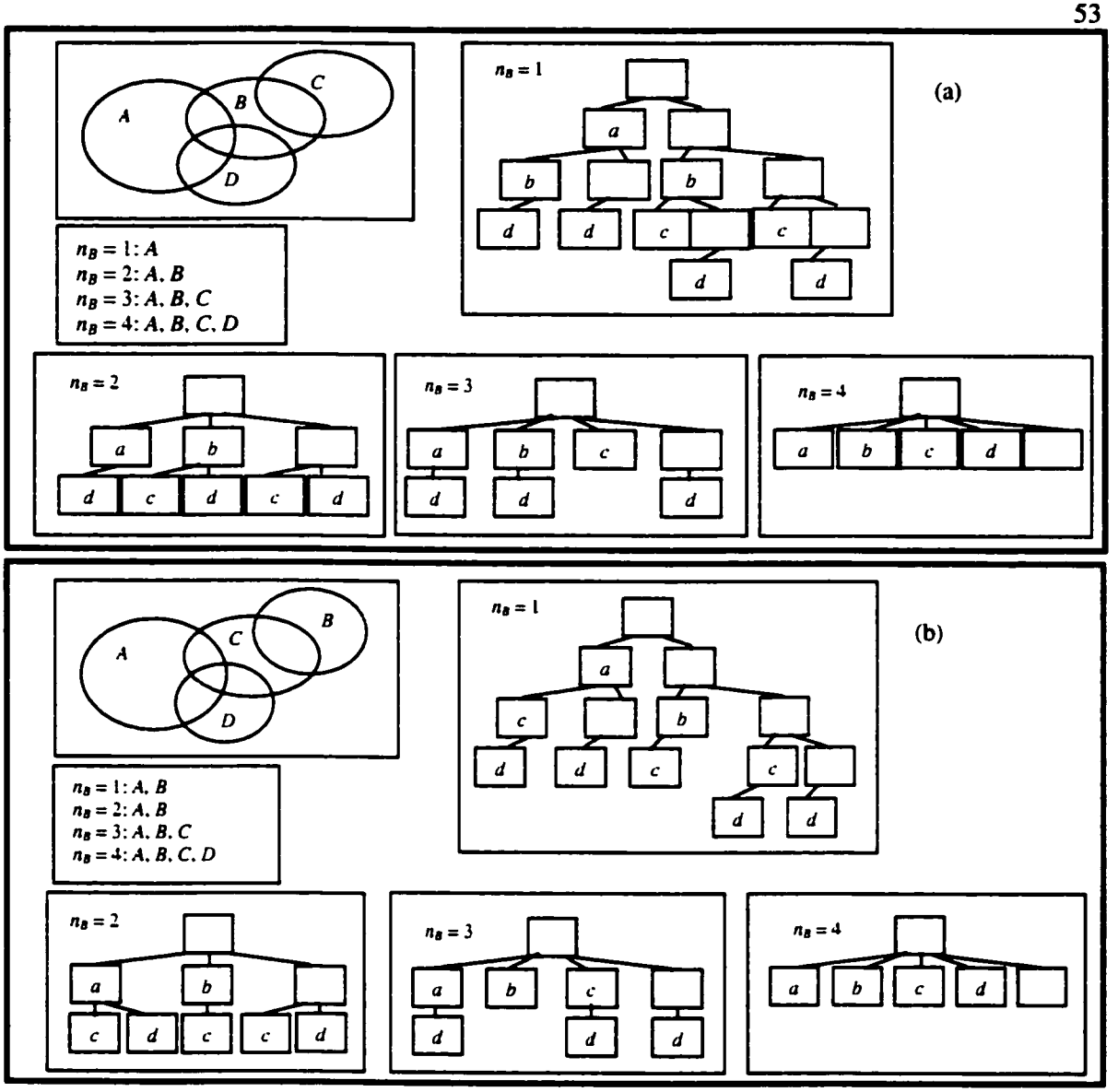


Figure 8: Graph Construction and Structure Reorganization. Two examples of the clustering process to illustrate how the output of CASTOR changes as the value of n_B increases as well as by structure reorganization. In both (a), and (b), the box on the upper left shows the Venn diagram for the support sets of the motifs that are present in the given database. The box right below shows the number of full support sets inferred at each value of n_B. The other boxes show the clustering process for each value of n_B. Each low-case letter represents a motif, and the corresponding upper-case letter represents the full support set of the motif. The difference between the two examples above lies in the order in which the motifs are discovered.

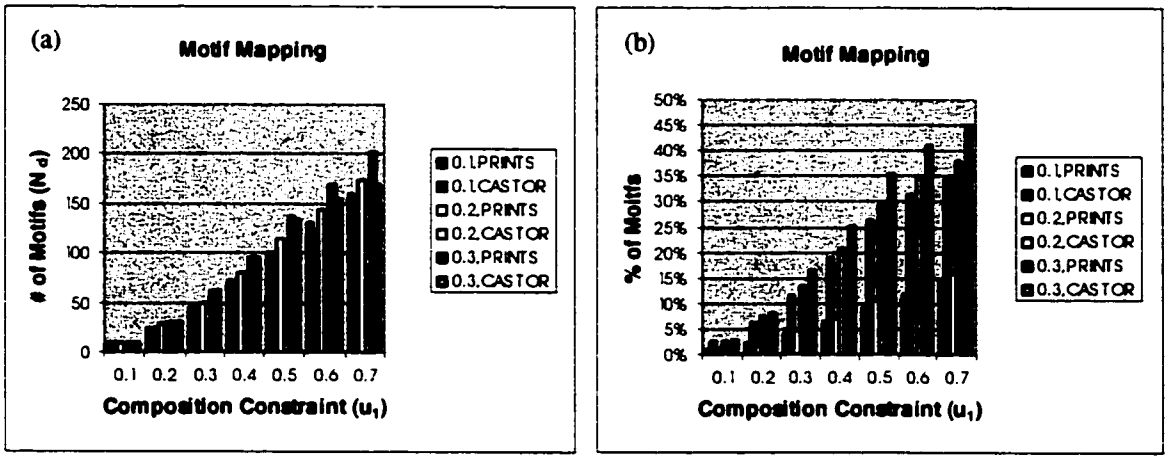


Figure 9: Motif Mapping. Each color corresponds to the specified value of the percent coefficient for the support identity constraint (t_1) for the specified system. A data value shown in (b) is computed as the ratio of the corresponding data value shown in (a) to the total number of motifs discovered by the specified system.

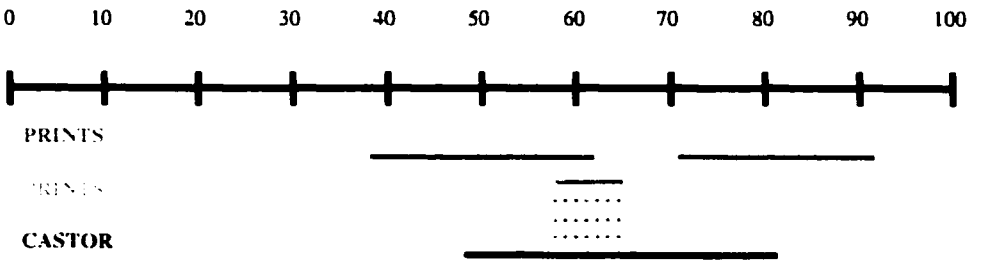
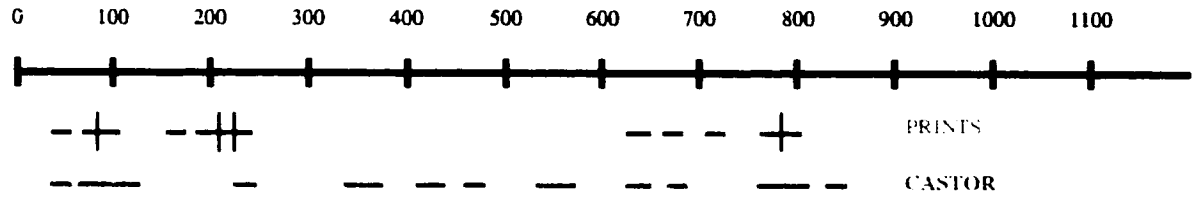


Figure 10: Overlap between One *C*-Motif and Several *P*-Motifs. A *c*-motif overlapping several *p*-motifs is shown. The top scale represents the first 100 residues of 5H1A_HUMAN, a human serotonin receptor, for illustration purposes. The two pink lines represent the occurrences of *p*-motifs identified for GPCRRHODOPSN. The first orange line represents a *p*-motif for 5HTRECEPTOR1, which is a subset of GPCRRHODOPSN. The dotted orange lines represent the occurrences of three other *p*-motifs that occur in the same region in other subsets of GPCRRHODOPSN. The cyan line represents the occurrence of the single *c*-motif that is present this region. This single motif captures six *p*-motifs. See text for discussion.



PRINTS	1	2	3	4	5	6	7	8	9	10	11	12
	42-54	75-90	90-109	159-185	192-211	211-227	227-244	620-642	665-686	706-729	759-782	782-803

CASTOR	1	2	3	4	5	6	7	8	9	10	11
	39-58	73-126	222-245	330-381	403-434	465-484	537-572	612-639	670-688	757-809	817-847

Figure 11: *C*-Motifs and *P*-Motifs for the Same Family. Motifs discovered by either PRINTS or CASTOR for the *b*-class GPCRMGR is shown. This group contains 1 CASR and 8 MGR* genes, corresponding to 22 protein sequences. The top scale represents the 1,194 residues of MGR1_HUMAN for illustration purposes. Pink lines represent the occurrences of the 12 *p*-motifs generated for this *b*-class. The first table gives the exact locations of these occurrences. The brown lines represent the occurrences of the 11 *c*-motifs discovered for this *b*-class. The second table gives the exact locations of these occurrences.

Table 2: Residue Mapping. The first row gives the ratio of the value of N_{s2} to the total number of g-residues. The second row gives the appropriate sequence coverage. Each row corresponds to a level such that only those c-motifs associated with nodes along the inference path discovered at up to the specified level are considered. The last column corresponds to the maximum depth and thus the inclusion of all the levels.

	1	2	3	4	5	6	7	8	All
g-residue %	6.5	13.4	21.4	25.0	32.1	43.5	48.2	57.4	66.7
Sequence coverage %	4.5	10.1	19.0	23.8	29.0	38.7	44.5	57.5	68.4

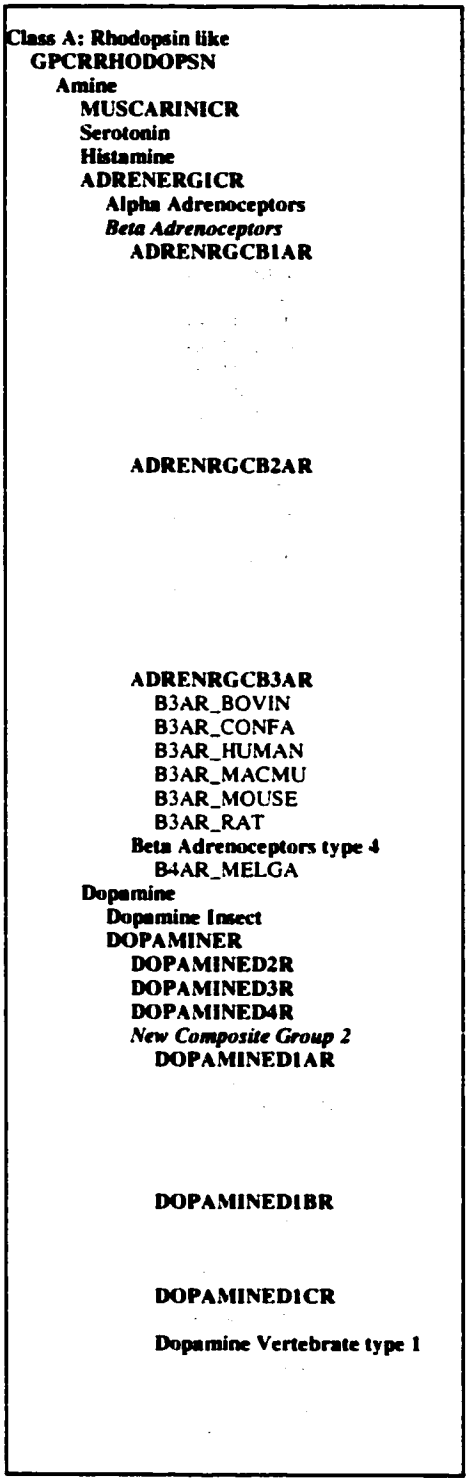


Figure 12: Interesting Family and Sub-Families. The biological class Amine is shown, which contains Beta Adrenoceptors and New Composite Group 2 as two rather distant subsets.

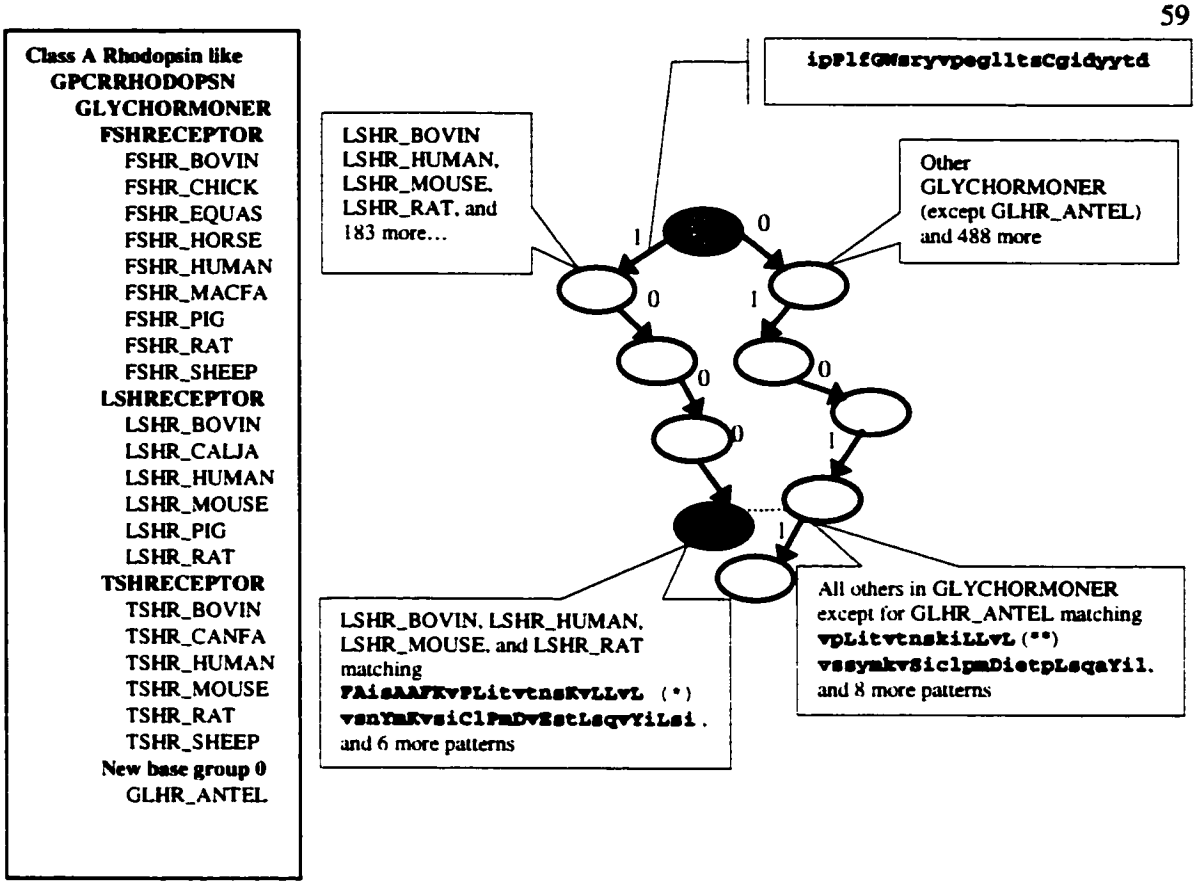


Figure 13: Classification for One Family. The biological class GLYCHORMONER is shown. The box on the left lists its subsets and member proteins. The diagram on the right shows how its members are classified via the tree approach.

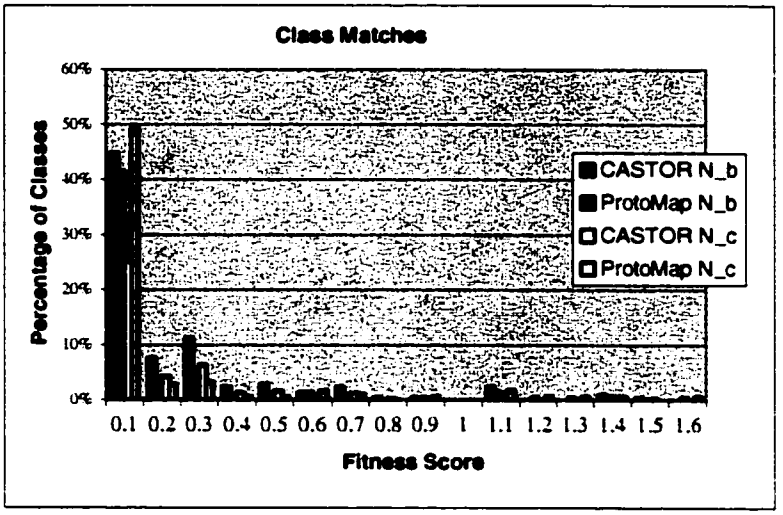


Figure 14: Comparison between CASTOR and ProtoMap. Bars of the same color correspond to the N_b or N_c performance-comparison statistic for CASTOR or ProtoMap. Each category on the x-axis means that the value of the statistic is equal to or greater than the label of the previous category, starting with zero, and less than that of the label of the current category.

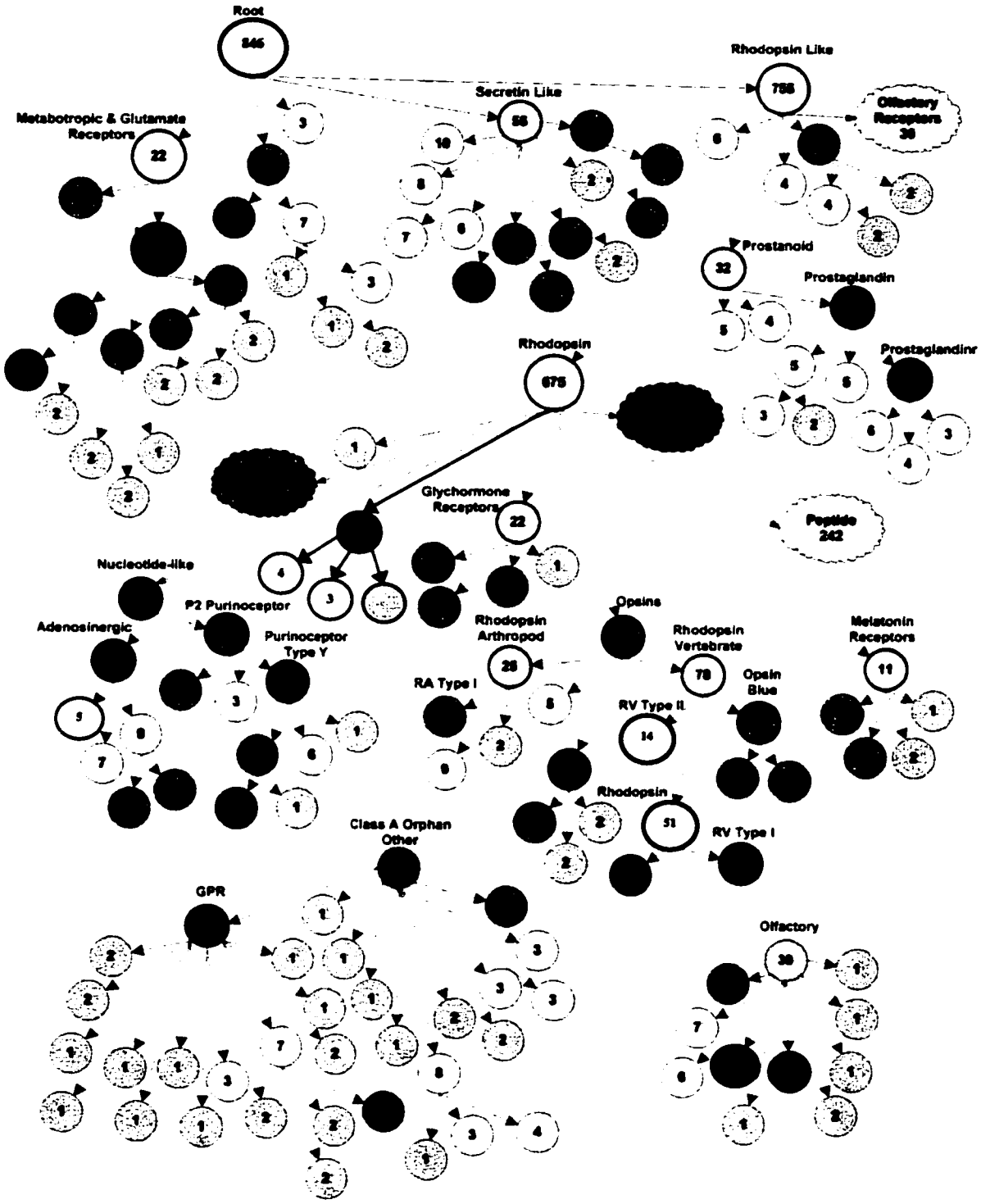


Figure 15: Color-Coded Reference Taxonomy (Part I). The reference class set is shown as a tree. Each node (circle or cloud) corresponds to a *b*-class and an arrow indicates that the node is contained by its parent. Some of the largest *b*-classes are first represented by clouds and subsequently by circles with thicker orange borders, which are properly expanded. Each node contains the size of the corresponding *b*-class. It is also labeled with the name of the corresponding *b*-class if the size of the *b*-class is greater than ten. This is continued in **Figure 15**.

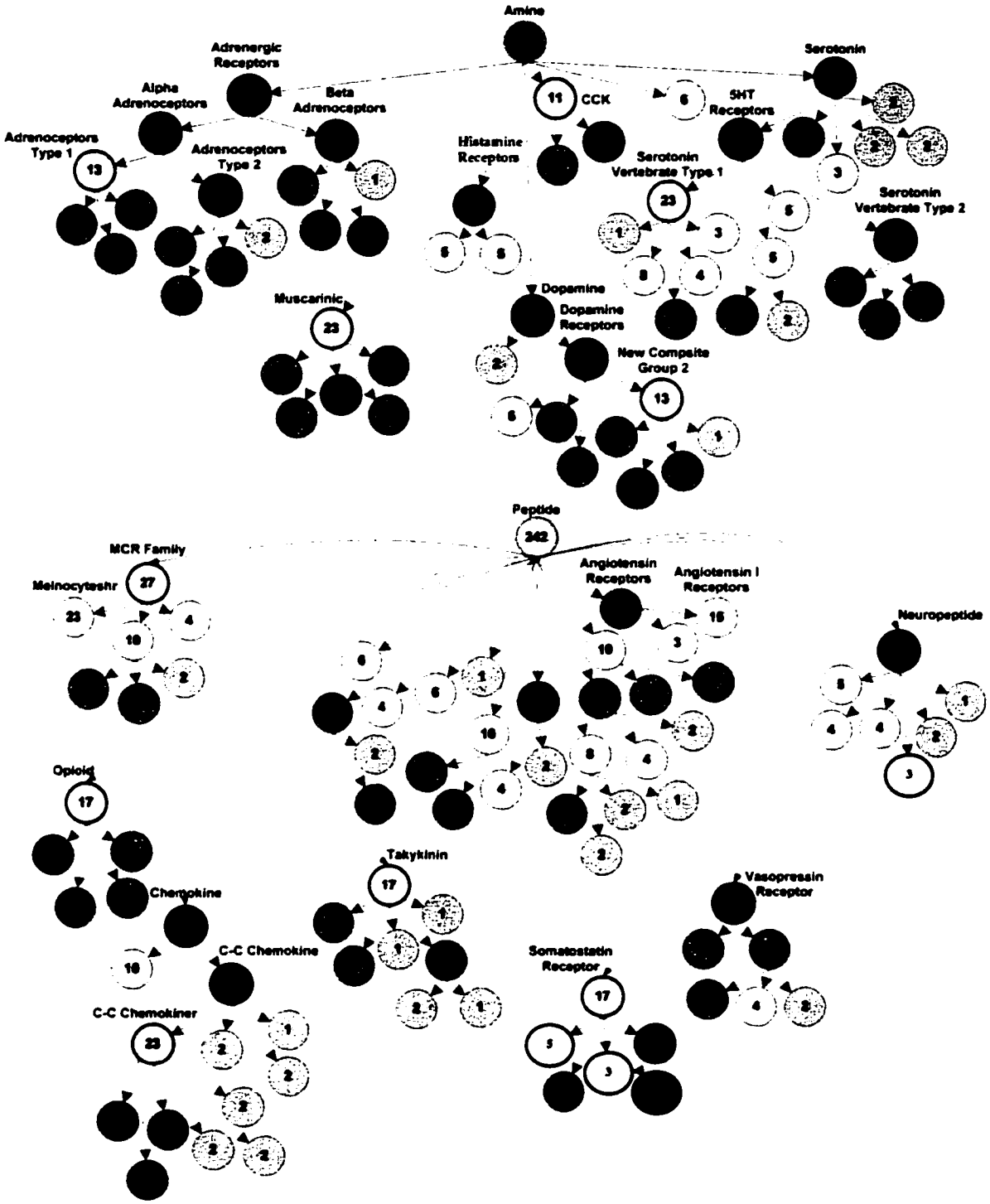


Figure 16: Color-Coded Reference Taxonomy (Part II). Each node has been painted with one of five different colors as follows:

a) A yellow node corresponds to a *b*-class that is matched both with a *c*-class and with a *p*-class.

b) A blue node corresponds to a *b*-class that could be matched neither with a *c*-class nor with a *p*-class.

c) A red node corresponds to a *b*-class that could be matched with a *c*-class but not with a *p*-class.

d) A green node corresponds to a *b*-class that could be matched with a *p*-class but not with a *c*-class.

e) A gray node corresponds to a *b*-class that has only one or two members.

Motif-Based Construction of a Functional Map for Mammalian Olfactory Receptors

1 Introduction

Mammals can smell thousands of environmental chemicals. The ability to detect a broad range of odorants can be attributed to thousands of olfactory receptors (ORs) residing on olfactory sensory neurons. The mammalian OR family is believed to be one of the largest protein families with one of the largest ligand repertoires. Although limited information about ligand recognition is available for selected ORs [2][36][44][51][61][79][88], ligand specificity and the specific functions of ORs remain mostly unknown. Nevertheless, the extremely large number of potential odorant-OR interactions [2][51] calls for many ligand binding sites and other protein regions that play functional roles in ORs, which inevitably leads to a complex organization of the mammalian OR family into subfamilies. By important functional role, we mean that the corresponding amino acids are critically involved in biochemical processes such as ligand binding, post-translational modification, G protein interaction, stabilizing structural conformation, axon guidance, etc.

Upon the recent release of the human genome and mouse genome, a comprehensive database of potential mammalian OR genes has become available [31][87][90]. This abundance of sequence data presents a significant opportunity to systematically explore this family to further understand its functional characteristics.

Even before the nearly complete OR repertoire was known, some efforts have been made to identify conserved regions in relatively few ORs. [52][58][90]. The conserved regions are typically identified by constructing a global multiple alignment of all the available ORs followed by identifying the conserved positions. Therefore, these regions are mostly present in all the available ORs, and it is certainly not straightforward to characterize these regions or compare them with some others. Some other efforts have been made to identify ligand binding sites in certain ORs based on the their primary structures using various computational techniques, such as correlated mutation analysis, Fourier analysis, and homology modeling [58][69]. In such an effort, typically individual residues rather than conserved regions are selected and studied. Now that a much bigger database of mammalian ORs is available, a systematic, unsupervised analysis would be most appropriate to explore the sequences for hints about important functional regions.

In this chapter, we present a systematic, unsupervised analysis of the nearly complete mammalian OR family to create a functional map of conserved regions, or motifs, that are likely to play important functional roles and to derive a corresponding functional taxonomy of classes that are likely to share common functional properties. Based on the evidence from the previous analysis of other protein families, including the GPCR superfamily, we expect a very significant majority of the discovered motifs to be true functional motifs, which should thus be further investigated particularly by experimental means. Furthermore, the same approach we used to generate and validate potential functional motifs and classes could be extended to other poorly characterized, large protein families.

We use CASTOR to generate functional motifs and classes for the mammalian OR family. Given a group of related protein sequences, CASTOR uses a top-down and recursive model to discover statistically significant motifs represented by regular expressions. These are further refined into profile HMMs and then used to infer a taxonomy of potentially overlapping classes on the given group of protein sequences. In [35], it has been shown that the pattern discovery algorithm [19] and the accompanying statistical framework [75] at the core of CASTOR are extremely likely to generate functional motifs. In a direct comparison with the PROSITE database, for instance, it has been shown that more than 75% of the discovered motifs are in fact associated with important biological functions. Additionally, by studying the serine protease family, it has been shown that a similar top-down and recursive model identifies motifs that are directly related to the active site. Out of the 11 discovered motifs, the three most statistically significant ones contain the three catalytic residues of serine proteases (serine, histidine, and aspartic acid), respectively, while the remaining seven motifs are critically related to the structure of the active site. Furthermore, in [48][49], it has been shown that CASTOR outperforms another automatic, sequence-based method in inferring a taxonomy on a given protein superfamily with respect to a manually constructed reference taxonomy.

Overall, it is especially appropriate to apply this approach to the mammalian OR family, which is very large, has little functional information available, potentially possesses many functional regions, and is potentially characterized by a complex organization. First, CASTOR is extremely efficient. For example, it takes about three hours to construct a complete taxonomy for the GPCR superfamily on a supercomputer of 64 nodes. Second, it proceeds in an unsupervised manner, requiring no prior functional information. Third, it works systematically, discovering statistically significant motifs with smaller and smaller support sets. Finally, it is capable of generating not

only a hierarchical taxonomy but also a non-hierarchical one, which indicate potential complex relationships among protein families.

A significant extension to the previous approach is that CASTOR is used not only to discover one-dimensional (1D) motifs from the amino acids in the primary structures but also three-dimensional (3D) motifs from the pseudo-sequences constructed to account for the proximity of the amino acids in 3D space. In this chapter, we focus on the 1D motifs in the construction of the functional map and the corresponding taxonomy but also examine the 3D motifs for illustration purposes. By combining the motifs with a variety of other clues, such as physio-chemical properties, analysis of local and global conservation, secondary and tertiary structural information, known functional motif databases, mutation data, and ligand binding profiles, we attempt, where possible, to infer and validate the functional or structural role of the motifs.

One of the key questions about the OR family is what is the underlying mechanism for ORs to recognize a large number of diverse odorants. We propose a combinatorial molecular mechanism, in which the ligand specificity lies in the simultaneous presence of multiple functional motifs in different regions. Specifically, each functional motif may have evolved independently and possess functional significance in its own right, but it is the ingenious interaction among multiple functional motifs that molds a mechanism that is as complex as ligand recognition for ORs.

This combinatorial molecular mechanism, which is reminiscent of the combinatorial signature generation mechanism in the immune system, is appealing for several reasons. First, it makes sense from an entropic perspective, as it is much more efficient to explain a large solution space by combining a moderate number of basic components in numerous ways than by having a huge number of elements, each accounting for one solution independently. Second, it is directly reflective of the observed combinatorial binding behaviors of ORs [51] where each OR recognizes a subset of all the ligands, and different such subsets could overlap. Conceivably, a region where functional motifs tend to occur invariantly is responsible for certain physio-chemical properties of the ligands. The composition of such regions in combination then determines the ligand specificity for the OR.

We explore the discovered motifs and find a few lines of evidence which strongly support the combinatorial molecular mechanism. Specifically, we observe extensive overlap among the motifs, which is consistent with the mechanism. We also identify a few examples where the

combination of a few motifs, but not any one of them, characterize small OR groups with similar ligand binding profile.

2 Results

This section contains an overview of the general properties of the mammalian OR family followed by a detailed account of the motifs discovered using CASTOR. For 1D motifs and 3D motifs, respectively, we discuss the general properties of the motifs, their relationships to known OR subfamilies, the functional roles of the individual motifs, and finally the relationships among multiple motifs.

We started by examining ORs globally, with respect to their primary-, secondary-, and tertiary-structures. We collected 1,332 potential full-length, intact amino acid sequences, which we will call the *complete OR database*. It covers about 90% of all the mammalian OR genes [87] and is the largest OR database ever analyzed. We also extrapolated the secondary structures for all the OR sequences. Under the assumption that for a relatively homogeneous protein family, a common functional activity would take place in all members at or near the same site, we constructed a global multiple alignment and a corresponding consensus sequence for the complete OR database (see Methods).

Each OR has seven transmembrane helices (TM1 – TM7) interconnected by extracellular (EC) and intracellular (IC) loops. From the global multiple alignment, we constructed a snake diagram (**Figure 17**) and a two-dimensional (2D) EC cross-section (**Figure 18**a, b) to show the general secondary and simplified tertiary structures. Since consecutive residues on a TM helix are theoretically located 100 degrees apart, all the residues could be mapped to one of eighteen possible faces, in a 20 degree angle increment (**Figure 18**a, b). Furthermore, since there are usually no more than 25 residues in a TM region, at most two positions would be mapped to the same face. We could see that there are two pockets formed by the helix bundles, the first pocket formed by TM1, TM2, TM3 and TM7 and the second one formed by TM3 through TM7.

2.1 General Properties of Olfactory Receptors

Since ORs can bind selectively to a large number of odorants, it is reasonable to assume that such specificity may be associated with those regions that are more variable globally. Therefore, we computed a variability index [58] for each position of the global multiple alignment (**Figure 17**, **Figure 18**a). We find that TM4, TM5, and the central region of TM3 are highly variable. The

last segments of the N- and C-terminals are also highly variable. IC loops are generally more conserved than EC loops, especially the parts immediately adjacent to the TM helices. For TM4 and TM5, the more variable faces do not necessarily face the inner core. For all the other TM regions except for TM7, which is highly conserved, the variable faces tend to face the lipid bilayer.

Since the majority of odorants are small hydrophobic molecules, it is reasonable to assume that ligand binding takes place in a pocket formed by the TM helices. Therefore, we examined the TM helices in more detail. We computed an average hydrophobicity index and side-chain volume for each transmembrane position in the global multiple alignment (see Methods) (**Figure 18b**). This is especially relevant since odorants come in both a hydrophobic (such as long-chain acids) and a hydrophilic (such as short-chain acids and alcohols) variety. We observe that, for several TM regions, the residues facing the inner core tend to be more hydrophilic and to have smaller side chains. Conversely, as should be expected, for TM1, TM3, TM4, and TM5, the residues facing the lipid bilayer tend to be more hydrophobic. Interestingly, the faces containing the more variable residues also tend to be more hydrophobic.

2.2 Overview of 1D Motifs

We ran CASTOR on the complete OR database to generate a set of 1D motifs and corresponding classes (see Methods). We obtained 86 motifs with 74 distinct support sets (**Table 3**). Each motif is represented by a profile HMM, whose consensus sequence is shown. The motifs are 10 to 29 residues long and they tend to occur in TM4, TM5, and EC3 (**Figure 19**). The support sets of the motifs include anywhere from 15 to 1,330 sequences. Notice that if the support set of a motif A is completely contained in that of another motif B, the set of sequences that match A is a subset of the set of sequences that match B. In the case of ORs, several support sets are completely contained within others, up to seven levels deep, hinting at a largely hierarchical taxonomy of classes (**Figure 20**).

2.3 Study of 1D Motifs with Respect to Known OR Subgroups

We first studied the generated motifs and classes by comparing the support sets of the OR motifs with certain groups of ORs that have been classified together based on their amino acid sequences and primary structures using techniques that suggest the sharing of protein function (see Methods). If there is any agreement, it will suggest that the subgroups of ORs are functional subsets, and the associated regions are functional regions. In that case, even if prior knowledge

about the functional subset was available, it was not used in the inference of the subset but only for validation and analysis purposes.

A few terms will be useful in discussing the results. We will say that a motif is conserved in a group if it is present in the vast majority of the members in the group. Consider now the group as part of a larger database. We will say that the motif is specific to the group if it is present in some members of the group but almost nowhere else in the database. Furthermore, we say that the motif characterizes the group if it is both conserved in and specific to the group. In other words, it is present in almost all the members of the group and nowhere else in the database.

2.3.1 Motifs Characterizing the Complete OR Database

Ten of the discovered motifs are conserved in the complete OR database (present in more than 90% of ORs) (**Table 3**, **Figure 21**a). We also constructed a database of 846 non-OR GPCR sequences and determined the presence of the discovered motifs in this database. Five of the ten OR conserved motifs, Motifs 1, 2, 3, 5, and 6, are not specific to the complete OR database (present in up to 40% of non-OR GPCRs). The other five, Motifs 4, 7, 8, 9, and 16, characterize the complete OR database (present in less than ~ 1% of non-OR GPCRs). Therefore, they might constitute sequence signatures for the mammalian OR family and thus participate in critical OR-specific functional activities. For example, they would be ideal candidates for degenerate PCR primers. As another example, all ORs are expected to bind to Golfs upon activation. Since IC3 is known to be a key determinant of G protein-coupling specificity [82], Motif 4, occurring partly in IC3, might contain some catalytic residues that contribute to the binding with Golfs.

2.3.2 Motifs Characterizing Class I and Class II Subgroups

ORs have been classified into either Class I (fish-like) or Class II (mammalian-like), with 165 and 1,167 members, respectively, in the complete OR database [31][87]. The split into Class I and Class II is likely to be the most ancient division of the mammalian OR family [27]. We computed a normalized ratio for each discovered motif to indicate its specificity to the Class I subgroup or Class II subgroup (see Methods).

45 motifs are specific to the Class I subgroup (normalized ratio > 7) and 10 motifs are specific to the Class II subgroup (normalized ratio < 0.10) (**Table 3**, **Figure 21**b,c). Among these, Motif 66 characterizes the Class I subgroup (present in more than 90% of Class I ORs), and Motifs 11 and 14 characterize Class II (present in more than 90% of Class II ORs). Interestingly, Motifs 66 and

14 both occur in the extracellular side of TM6. If the extracellular part of a TM region plays a critical role in the differentiation between Class I and II, the two classes might either select different kinds of odorants or employ different mechanisms for axon guidance

2.3.3 Motifs Characterizing Human and Mouse Subgroups

There are 322 human ORs and 1,010 mouse ORs in the complete OR database. We also computed a normalized ratio for each motif to indicate its specificity to the human subgroup or the mouse subgroup.

Motifs 49 and 56 are specific to the mouse subgroup (normalized ratio < 0.10) (**Table 3**). They have relatively small support sets (present in < 50 ORs) that contain almost exclusively Class II ORs. Conceivably, they may correspond to the unique regions that have survived in mice but disappeared in humans or have evolved in mice after the separation of humans and mice. The other motifs are specific to neither the human subgroup nor the mouse subgroup, suggesting that humans have retained most of the important functional motifs.

2.3.4 Phylogenetic Clusters

We constructed a consensus phylogenetic tree for the complete OR database (see Methods). We chose the 999 reliable clusters (good bootstrap supports > 50%) in the phylogenetic tree for further consideration and considered some of these as phylogeny-based families [87].

In total, 31 OR motifs characterize sixteen clusters. 25 of the 31 OR motifs characterize the clusters corresponding to the complete OR database, the Class I subset, or the Class II subset (**Figure 20**). The remaining six motifs all have relatively small support sets. For example, OR Motif 49 is present in 17 OR sequences, which all belong to one family, while OR Motif 56 is present in 33 out of 40 OR sequences, which belong to seven closely related families. The 55 OR motifs that do not characterize any clusters have support sets that are quite different from the clusters. In the most dramatic cases, the OR motifs have relatively small support sets and yet are specific to neither the Class I subset nor the Class II subset (normalized Class I ratio ~= 1.0). In some other cases, the OR motifs are specific to either the Class I subset or the Class II subset, but they would be present in the OR sequences that belong to distant families. The small number of matches between the support sets of OR motifs and the phylogenetic clusters combined with the large amount of discrepancy in the non-matches suggests that the evolutionary development of ORs substantially disagrees with the functional development of ORs.

We studied some of the motifs that do not characterize any clusters in the hope of gaining some functional insight from an evolutionary point of view. OR Motifs 39, 63, 71, and 78 are present in relatively few OR sequences that belong to several distant families. Each corresponding region could be an evolutionary relic in that it has lost its functional significance and thus disappeared in most ORs, or it could be the product of convergent evolution in that it has developed in evolutionarily distant ORs independently. We examined the distribution of matching score against the complete OR database for each of these motifs. For OR Motif 71, the distribution is relatively flat. The corresponding region might then be an evolutionary relic, since instances of the region should have all evolved and survived for a long time and thus become almost indistinguishable from one another and from the other parts of the OR sequences. For each of the other three OR motifs, the distribution has a clear cutoff that separates the support set of the OR motif and the rest of the complete OR database. Each corresponding region might then be a product of convergent evolution, since instances of the region should have been relatively young and retained the distinctive features.

2.4 Analysis of Individual 1D Motifs

We next studied the functional roles of the discovered motifs by combining a variety of clues, including the conservation of individual residues, and the matching with known motif that characterize other protein families and domains and known residues whose mutation alters protein functions (see Methods).

2.4.1 Sites with Structural Roles

We started by attempting to identify the discovered motifs that are likely to possess structural significance. Given that these would be unlikely to provide finer-grain functional specificity, we expected them to be conserved in the complete OR database (> 80% of ORs). Motif 7, occurring in EC2, has three conserved cysteines at Positions 5, 15, and 25. Motif 8, occurring in TM3, has an additional conserved cysteine at Position 7. The second conserved cysteine in Motif 7 and the conserved cystein in Motif 8 may form a disulfide bond [90], as they do in rhodopsins and amine receptors [8]. The first and the third conserved cysteines in motif 7 might form a second disulfide bond to further restrain the structure of EC2 in ORs. Motif 9, occurring in TM2, has a conserved aspartic acid at Position 1, while Motif 2, occurring in TM7, has a conserved asparagine at Position 4. The two residues, also conserved in many family A GPCRs, are close to each other

on the crystal structure of bovine rhodopsins [56] and might interact with each other to stabilize the overall structure in Family A GPCRs [89].

Motif 12, occurring in the C-terminal, is lysine and arginine rich. Despite the conventional secondary-structure predictions, we could argue that the corresponding region forms an additional alpha-helix. In the alignment of the consensus sequence for the complete OR database with the bovine rhodopsin sequence, this region matches the so-called "H8 helix" in bovine rhodopsins, which is a short alpha-helix positioned parallel to the membrane and right after TM7 [56]. In particular, the conserved lysines and arginines in Motif 12 are periodically distributed in this region. It is known that a lysine and arginine-rich end of a TM region could anchor the TM region to the membrane through ionic interactions with phospholipid head groups [7]. The alpha-helical periodicity (three or four positions apart) of these positively charged residues further implies that they may be in the same face on an alpha-helix and thus very likely to simultaneously engage in ionic interactions. Therefore, the presence of these conserved lysines and arginines may confirm the existence of an additional TM region. Motif 12 is absent in a small number of OR sequences, but the lysines and arginines are conserved at the corresponding alpha-helix periodical positions across the complete OR database.

2.4.2 Post-Translational Modification Sites and Other Functional Sites

We also compared the motifs with 1374 motifs in PROSITE [5] (see Methods). We obtained a few additional matches corresponding mostly to post-translational modification sites. Motif 10, occurring in the N-terminal of most OR sequences, contains a potential glycosylation site (Positions 1 ~ 4), which is known to be present and functional in other GPCRs [83]. Motif 42, occurring in the EC3 region of a relatively small number of ORs, may contain an additional N-glycosylation site (Positions 4 ~ 7), which is known to be present but not glycosylated in the V1a vasopressin receptors [83]. In GPCRs, glycosylation generally plays roles in expression, folding, and/or stability without affecting ligand binding or intracellular signaling [83]. In ORs, glycosylation might additionally be involved in axon guidance. It might be interesting to block the glycosylation sites and see what effects are produced.

Motif 2, occurring in the C-terminal of most OR sequences, appears to be involved in PKC phosphorylation (Positions 9 ~ 11). In light of the presence of a potential H8 helix, this motif would occur in the loop between TM7 and H8. Motif 45 occurs in the C-terminal as well (following OR Motif 2) in relatively few OR sequences, and part of it that is not very conserved

might correspond to a PKA phosphorylation site (Positions 6 ~ 9) that is present in a majority of the OR sequences (~ 3/5) in its support set.

On the other hand, GRK3s are the only GPCR-specific kinases that express in olfactory sensory neurons [65][21]. The motif that describes the corresponding phosphorylation site (if any) is not yet identified. However, the residue that is phosphorylated is expected to be preceded by some acidic residues in IC3, as is the case in alpha2-adrenergic receptors with respect to beta-GRKs [55]. This leads to Motif 4, which occurs in IC3 of most ORs, often preceded by two acidic residues, and contains a few conserved serines and threonines, which could be phosphorylated.

We further compared all the discovered motifs with 11853 motifs in BLOCKS+ [38] (see Methods). We obtained several matches (**Table 3**), where the discovered motifs occur mainly in the TM regions of ORs and the motifs in BLOCKS+ occur mainly in the TM regions of other transmembrane proteins, such as ion channels, exchangers, and transporters. These discovered motifs could be highly common to transmembrane proteins.

The fact that the majority of the motifs did not have a match in PROSITE or BLOCKS+, even though they have significant supports in ORs, is encouraging, as these may be related to the specific odorant binding characteristic of ORs.

2.4.3 Sites Matching Mutation Experiments in Other GPCRs

To identify potential functional residues in ORs, we analyzed the discovered motifs against 336 residues in GPCR sequences from tGRAP [10] and a separate database [80] (see Methods). These residues have been shown to be functionally active by single-substitution or single-deletion experiments, and we will refer to such residues as *mutation residues*. To increase the chance that these mutation residues possess the same functional significance in ORs as they do in GPCRs, we required that for a mutation residue, a discovered motif occur in the region of its source sequence where the mutation residue resides.

This leads to relatively few matches because, as mentioned before, most of the discovered OR motifs do not match non-OR GPCR sequences. Motif 1, occurring in TM3/IC2 of most ORs, contains the well-known DRY sub-motif (Positions 4 ~ 6) [29]. It also contains the MAY sub-motif (Positions 1 ~ 3), which is almost the only part of Motif 1 where no match with a mutation residue could be found. The MAY sub-motif actually characterizes the complete OR database and might thus constitute a sequence signature for the mammalian OR family as well as contain

functional residues that are not present in non-OR GPCR. Motif 5, occurring in TM5 of most ORs, matches two mutation residues (Positions 1 and 5), respectively. Both residues seem to be highly conserved in TM5, which is one of the more variable regions. Furthermore, both corresponding mutation experiments result in the decrease of agonist binding constants. Therefore, the positions in Motif 5 may play a role in ligand binding. Motif 6, occurring in TM1 of most ORs, matches one mutation residue (Position 8) in a similar way as Motif 5 matches the relevant mutation residues, and thus the position may also play a role in ligand binding.

We obtained some other matches as well. They usually have slightly worse matching scores, and they always reside in some regions that do not agree with the regions where the corresponding OR motifs occur. In addition, the mutation residues agree with the corresponding consensus residues in half of the cases and rarely or never show up at the corresponding positions in the motifs in the other cases. On the other hand, almost all the corresponding OR motifs occur in the TM regions or parts of the EC regions next to the TM regions. In addition, almost all the corresponding mutation experiments result in the changes of ligand binding constants. Furthermore, when multiple mutation residues are matched with different positions in the same OR motif, the positions usually are on the same face of an alpha-helix in the second pocket.

2.5 Analysis of Multiple 1D Motifs

We next studied the discovered motifs in groups to learn more about the potential functional roles they play.

2.5.1 Sites with Similar Functions

Some OR motifs, even if occurring in distinct regions, are so similar that they could be combined to form one motif that would still be highly specific. We identified a few cases (**Figure 22**). In each case, the OR motifs all occur in similar parts of the corresponding secondary-structure regions, such as all in the middle or all at the boundary of some TM region. The similarity in amino acid sequence and in secondary-structure location suggests that these OR motifs might correspond to regions with similar functions. In some cases, such similar motifs are present in identical ORs. For example, OR Motif 63 occurs in TM5 and OR Motif 64 occurs in TM7, and the former is present in most of the OR sequences where the latter is present. In the other cases, such similar motifs are present in different OR sequences.

2.5.2 Sites Involved in Ligand Binding

We assumed that a common functional activity would take place in all members of a protein family at or near the same site. It would also be reasonable to assume that differences between related functional classes within the family arise mostly from mutations at or near such sites. It has then been shown that under these assumptions, active sites, and functional interfaces may be identified for homologous proteins based on sequence conservation patterns and the mapping onto the appropriate protein surface [47]. In particular, such sites and interfaces correspond to those positions that mutate in concert with functional divergence, thereby conserved either within individual functional classes or even invariantly across the entire group of proteins, and cluster spatially. In our case, those extracellular positions that are conserved only within individual classes (locally conserved) and moreover highly variable across the complete OR database (globally variable) are especially likely to be the ligand binding sites due to the broad spectrum of ligands of ORs.

In looking for such globally variable and locally conserved (GVLC) positions, we focused on the more variable regions and relatively small classes, since larger, non-overlapping classes imply fewer functional motifs present in each region, which is unlikely to result in a large combination of functional motifs present in different regions. Therefore, we considered TM4, TM5, and EC3. For each of these regions, we considered those motifs that have relatively small support sets (< 200 ORs) that barely overlapped (< 1/6 of motif pairs with > 10% overlap of both support sets). In total, we identified ten such positions (see Methods) (**Figure 23**), which indeed tend to be on the extracellular side of the membrane. These GVLC positions could be the key residues involved in ligand binding.

We also observed extensive overlapping between the support sets of OR motifs. Specifically, we analyzed every possible pairwise combination of the discovered motifs with moderate support sets (< 400 ORs). For each pair of such motifs, we computed the overlap or intersection of their support sets, which would consist of all the ORs where both motifs are present. We then constructed the union of such intersections. The union covered a large portion of the complete database (992 ORs). Such extensive overlapping is rarely seen when we discovered motifs in other protein families (serine protease, kinase). This overlapping could be explained by the combinatorial molecular mechanism for ligand binding as proposed in the introduction. In such mechanism, support sets of motifs from different regions will need to overlap thus the two motifs

can work together, so to create more combinations. The majority of the ORs should be in one or more overlapping support sets, thus the union of such support sets should be large. Therefore, this observation favors the combinatorial molecular mechanism.

2.6 3D Sequences

All OR TM regions are alpha-helices. By rearranging the amino acids in the TM regions into pseudo-sequences that reflects their proximity in 3D space, we attempted to find conservation in 3D space (**Figure 24**). These 3D sequences may represent different and more subtle functional units than those found in the primary structures, since each of them corresponds to a very compact unit consisting of several resides that are likely to be jointly involved in a functional activity.

As mentioned before, there are theoretically eighteen faces in each TM helix arranged in a wheel manner, where Face 0 is adjacent to Face 17 (**Figure 24**). By combining the residues in every possible N adjacent faces, or mega-face, we can define 18 3D sequences for each OR sequence. We chose $N = 5$, so that each 3D sequence would span a 90 degree range (see Methods) and captures a significant portion of the helix. Each 3D sequence generally consists of five to seven residues, with consecutive residues spaced by either two or three positions in the corresponding primary structure.

For each mega-face in a given TM region, we selected the appropriate residues for each 3D sequence consistently over different ORs based on the global multiple alignment of OR sequences and the 2D EC cross-section (see Methods). We will refer to the group of such 3D sequences corresponding to a mega-face in a TM region as a *3D database*. Each 3D database contains approximately 1332 3D sequences, as the secondary-structural information is at times not available. There are a total of 7 x 18 = 126 3D databases.

2.7 Overview of 3D Motifs

There might be a higher level or different types of conservation in the 3D sequences, which is expected to correspond better to the development of protein function. Such conservation could again be described by motifs, which are not easily detected in 1D space. As mentioned previously, we assume that a functional activity happen in the same location in different ORs. Since the 3D sequences inherently correspond to distinct regions, we naturally would want to consider only those that correspond to the same region each time in performing motif discovery.

Therefore, We applied CASTOR to each 3D database independently to discover potential functional 3D motifs and infer the corresponding classes. We hypothesized that for each 3D sequence, all the residues in the corresponding mega-face could be involved in a functional activity together and thus be conserved. Therefore, we looked for 3D motifs that covered almost all the positions in each mega-face.

Due to the compactness of the 3D sequences, it is generally unnecessary to extend a 3D motif discovered by SPLASH and inappropriate to compute the support set of the 3D motif by threhsolding the *E*-values corresponding to matching the motif represented by the resulting profile HMM against the 3D sequences in a 3D database. Therefore, each 3D motif is represented by a regular expression consisting of amino acids, similarity classes of amino acids shown as multiple amino acids enclosed in square brackets, and don't-cares shown as dots, without being further refined into a profile HMM. While each mega-face consists of five adjacent faces, we say that a 3D motif occurs in the center face of the corresponding mega-face of the corresponding TM region.

Depending on the specific 3D-database, we obtained anywhere from 30 to 311 3D motifs. The compact nature of 3D sequences (5 to 7 residues) makes it impossible that distinct 3D motifs are present in identical 3D sequences. Accordingly, for each 3D database, the support sets of the discovered 3D motifs are all disjoint. Furthermore, many (> 50%) of the support sets have no more than five 3D sequences, and few (~10%) have more than ten in general (**Figure 25**a). There is a class for every discovered 3D motif, and we created an additional class for each 3D sequence that would not match any of the discovered motifs. The adjusted number of classes correlates generally well with the variability index for the corresponding mega-face, so that higher variability indices in the group of faces would correspond to larger number of classes (**Figure 18**b, **Figure 25**b).

We compared the 3D and the 1D motifs according to a number of criteria. While 1D motifs tend to be longer (up to 12 residues), 3D motifs are shorter (up to 7 residues) but tend to span a significantly larger region, typically an entire TM region. As such, they account for a large number (> 20) of consecutive amino acids in the primary structures. While 1D motifs tend to have relatively large (> 50) support sets, 3D motifs on average have very small (< 5) support sets. Furthermore, there are generally many more 3D motifs than 1D motifs in each TM region. Overall, the support set of a 1D motif is very rarely contained in that of a 3D motif, while the contrary is quite common. There are also a significant number of 3D motifs that do not overlap

with any 1D motif both in terms of support set and location, which are likely to be interesting features that would be completely missed in the OR sequences.

Some 3D motifs are special in that that the number of residues in the primary structure between two certain consecutive positions in such a 3D motif is not constant (either three or four) across its support set. This could happen when the starting position of such a 3D motif shifts (by one position) in its occurrences. Such 3D motifs account for a nontrivial portion (~ 17%) of the total population of 3D motifs and are concentrated in certain outward faces in TM1, TM4, and TM5. We imagined that they might still correspond to functional regions, as the irregularities appear insignificant. The shift of the starting position of such a 3D motif and thus the variable spacing between a certain pair of consecutive positions in the 3D motif implies some conflict in aligning the amino acids in the primary structure in the corresponding TM region, which is unique and interesting.

We shall call a 3D motif genuine if several of the remaining faces (not part of the corresponding mega-face) in the corresponding TM region are highly variable within its support set. We are generally especially interested in genuine 3D motifs (**Figure 26**a) because non-genuine ones would be likely to be embedded in 1D motifs as all the faces (and therefore all the positions in the primary structures) are conserved within their support sets. In other words, highly genuine 3D motifs are more likely to add significant information with respect to the 1D motifs. We computed a genuineness value for each 3D motif that is not special, as discussed in the previous paragraph, (see Methods) to indicate how variable these remaining faces are. We also estimated two baselines by Monte Carlo simulations of highly conserved and highly variable faces (**Figure 26**b). While almost none of the 3D motifs is completely genuine, many are associated with faces that are distinctly more conserved than the rest of the TM region (~ 35% with genuineness value > 2.7. These 3D motifs are mostly concentrated in TM4 and TM5.

2.8 Study of 3D Motifs with Respect to Known OR Subgroups

We studied the 3D motifs and classes with respect to specific subgroups of ORs that have been clustered together based on their ligand binding profiles. Any agreement between our results and such information would reinforce the notion that such subgroups of ORs constitute functional classes, and the associated motifs might then be functional motifs. It should be noted that no prior knowledge about protein functions was assumed in generating the motifs and classes but only for validation and analysis purposes. In that sense, our analysis falls in the category of

unsupervised learning. This is different and significantly more complex than the more standard supervised learning, where motifs would be discovered from protein sequence that had been previously classified according to some functional criteria, such as ligand-binding activity.

Two ORs, M71 and M72, have similar, although ultimately different ligand binding profiles. The intersection of the support sets of five 3D motifs, which are all very small (< 5 ORs), exactly characterizes this subgroup of two ORs. These 3D motifs are LVGGE in Face 2 of TM4, LF[ILMV]II in Face 15 of TM4, F[FW][IL][ILM]I in Face 11 of TM5, [ILMV]F[IV][ST]I in Face 14 of TM5, and FV[FY]AY in Face 1 of TM6. The other ORs that also match some of these 3D motifs might have similar ligand binding profiles as M71 and M72.

Three ORs, S50, S79, S85, all recognize the nonanedioic acid [51]. The intersection of the support sets of two 3D motifs, respectively containing 5 and 10 ORs, exactly characterizes the subgroup of three ORs. The two 3D motifs are L[HY][IL]A[ILM] in Face 8 of TM1 and SLPNY in Face 0 of TM7. Note that these 3D motifs are located in rather distant areas in the protein tertiary structure. However, the nonanedioic acid is a rather long molecule having seven CH_2 groups separating two carboxyl groups and could easily extend between these two sites. Alternatively, one of the two could be involved in access rather than binding activity. S85 also recognizes some other acids that S50 and S79 do not recognize. Correspondingly, one additional motif L[IV][FY][ILMV]S in Face 1of TM6, which has a support set of 11 ORs, is present in S50 and S79 but not in S85. It may be responsible for the additional selectivity of these two proteins. All of these 3D motifs are genuine (genuineness values > 2.7).

Two phylogenetic clusters, Family 258 and 259, may be involved in high-affinity binding with the isovaleric acid [87], while exactly which ORs in these families are the high-affinity receptors are currently unknown. Some 3D motifs, such as LNAPY in Face 3 of TM7, are highly specific to the union of these two families. Some additional 3D motifs are specific to only one of the two families. For example, ANSNI in Face 17 of TM7 is specific to Family 258, while [ST]NTN[IV] also in Face 17 of TM7 is specific to Family 259. When more specific ligand binding profiles for these two families becomes available, the latter 3D motifs may hold the key to the finer grain differentiation.

The OR 37 subfamily has unique expression patterns [76], which may be tied to specific functions. While no single 3D-motif characterizes this subfamily, a triplet of 3D motifs, Q.GE[ILM]M occurring in Face 2 of TM3 in 159 ORs, [ILMV].GCLA occurring in Face 4 of

TM3 in 60 ORs, and TAV[ILMV].[ILMV] occurring in Face 11 of TM6 in 38 ORs, collectively characterize this subfamily.

The analysis we have presented is far from exhaustive. However, notice that while we could find almost no instance of an individual 3D motif characterizing a subgroup of ORs with similar ligand binding profiles, we did find several examples where a combination of multiple 3D motifs is highly specific to, or even characterized the subgroup. This supports our hypothesis that the virtually limitless specificity of ligand binding would be implemented by a combinatorial molecular mechanism relying on the interaction of a few functional motifs.

2.9 Analysis of Individual 3D Motifs

We selected two especially interesting groups of 3D motifs (**Table 4**, **Table 5**). The first group consists of 101 3D motifs with large support sets (> 150), 70 of which are genuine. Such large-support 3D motifs are likely to correspond to TM sub-regions that play an important structural or other critical functional role. The second group consists of 115 3D motifs that are highly hydrophilic (hydrophobicity index < -1) (see Methods). Such 3D-motif may be involved in the binding of the more hydrophilic odorants, such as short-chain acids and alcohols.

The large-support 3D motifs generally occur in the inward faces except in TM4 and TM5, suggesting that these faces might possess important and thus common functional significance (**Figure 27**a). Furthermore, these 3D motifs are generally genuine except for those that occur in TM7, which is a generally very conserved helix (**Figure 18**a). As an example, the 3D motif LL[DE][ST][ILMV] occurring in Face 6 of TM2 in 419 ORs, is conspicuously genuine (genuineness value ~= 3.2). The acidic residue [DE] is generally highly conserved in Family A GPCRs and is thought to play important roles in receptor activation [29]. The fact that we found a genuine 3D motif containing this residue suggests that the few faces around this residue are also functionally important. Interestingly, the 3D motif QF.[DE][ILMV]M, occurring in Face 17 of TM3 in 711 ORs, also has a corresponding acidic residue, which is almost conserved in the complete OR database and is at about the same height in the helix as the one in TM2. Presumably, the residue in TM3 could prevent that in TM2 from adopting certain conformations, which could contribute to the activation or the inactivation of the specific ORs.

Hydrophilic 3D motifs are interesting, even if they are not genuine, since the TM regions are predominantly hydrophobic. Hydrophilic 3D motifs tend to occur in TM4 and TM7 (**Figure 27**b). A few of those in TM7 have large support sets, while all those in TM4 have small support

sets. Some occur in TM5 in faces opposite those in TM4. Genuine hydrophilic 3D motifs tend to occur in TM4, which is also the only region where there are no genuine large-support 3D-motifs, or otherwise in the inward faces. For instance, the genuine hydrophilic 3D-motif, FSGHQ occurs in Face 2 of TM4 in 9 ORs. The conserved serine, histidine, and glutamine in the 3D motif might serve as potential interaction sites for hydrogen bonds and salt bridges, while the small glycine could save space for the interactions to happen.

2.10 Analysis of Multiple 3D Motifs

As shown previously, the specificity of odorant-OR interactions seems to involve several TM regions at once. To confirm that the specificity is realized in a combinatorial way rather than a class-specific way, we looked for groups of 3D motifs that occur in different TM regions in the same pocket (I or II), with nearly identical support sets (see Methods). We obtained very few such groups, which further supports our hypothesis that a combinatorial molecular mechanism is at work, since if the solution to each ligand binding instance was identified independently, the relevant functional motifs would all be characterizing the subgroup of ORs under consideration. One notable exception consists of four 3D motifs all supported by roughly 20 ORs. These include [ILMV].GFLS in Face 5 of TM3, C[ILMV]WG[IV] in Face 5 of TM4, H[ILV][ST]C[ILMV] in Face 15 of TM6, and [ILMV][ST]PPT in Face 5 of TM7. These 3D motifs are all located in the second pocket, suggesting potential simultaneous interactions with a ligand. The 20 ORs belong to proteins that would be clustered by phylogenetic analysis and which are all located on a small region (111.73-112.41Mb) of Chromosome 10.

3 Discussion

In this chapter, we used CASTOR to generate a substantial list of potential functional regions (motifs) for the most up-to-date mammalian OR database available, containing more than 1000 members. We not only searched for motifs in 1D space but also used an unique approach to find conservation in 3D space. Extensive downstream analysis of all the motifs revealed potential interesting motifs that could be involved in specific functions. Therefore, we created a functional map of conserved regions that could be functional important for ORs and a corresponding functional taxonomy of classes that are likely to share common functional properties. In particular, combinations of these classes are likely to correspond to groups of ORs with similar ligand binding profiles.

3.1 Combinatorial Molecular Mechanism for Ligand Binding

In an attempt to explain the fundamental question of how the OR family is able to recognize a huge number of odorants, we propose a combinatorial molecular mechanism. In this mechanism, the ligand specificity lies in the simultaneous presence of multiple functional regions (motifs) in different regions.

This combinatorial molecular mechanism is reminiscent of the combinatorial signature generation mechanism in the immune system. One major difference is that by combining a set of variant gene cassettes, the billions of antibodies are generated via specific mutation mechanisms *de novo*, while the ~1000 OR genes are already there in the genome. Conceivably, the chemical sensory system cannot afford a long turnaround due to the potentially complex protein-synthesis and response-learning processes, and so a large number of genes with a corresponding pre-programmed neural circuitry are devoted for olfaction.

As mentioned in the introduction, this hypothesis fits well with both the entropic theory and the known ligand profiles of ORs. Our results from the motif analysis support the combinatorial molecular mechanism. First, we observed unusually extensive overlaps among the support sets of OR motifs, which could be easily explained by the combinatorial molecular mechanism which requires simultaneous presence of multiple motifs in different regions. Second, we were able to identify a few groups of motifs with distinct support sets, where each group characterizes a subset of ORs with similar ligand profiles. This is also as expected from the combinatorial molecular mechanism and rules out the model where ligand specificity is evolved independently for each group of ORs with similar ligand binding profiles. We expect future investigations and analysis to expand and refine our proposed combinatorial molecular mechanism by sorting out the details in the relationships between the functional motifs and the ligands, which could be fine-tuned in various ways, possibly dependent on many factors.

3.2 Conservation in 3D Space

The TM regions of ORs are where the ligand binding sites are expected to reside. They are alpha-helices, whose special conformations prompted us to systematically examine potential conservation in the 3D space. We constructed 3D sequences and looked for 3D motifs accordingly.

When we studied the TM regions, we specifically looked for ligand binding sites. As we did so, we hypothesized the following. When ligand binding takes place in a pocket formed by several TM regions, a group of consecutive faces rather than a single one in each of these TM regions would be involved. In addition, almost all the positions rather than a few scattered ones in these faces could be involved thus conserved, even if only a couple of them might be in physical contact with the ligand. Furthermore, such a group of faces could be distinctly more conserved than the other faces in the TM region for the ORs sharing the specific ligand binding behavior. Therefore, we looked for 3D motifs that cover almost all the positions in a group of faces in a TM region, and we specifically identified those such that the group of faces is distinctly more conserved with respect to their support sets. We indeed discovered a significant number of such 3D motifs, even if no group of five consecutive faces appears to be the only conserved part in any TM region across the complete OR database (**Figure 18**a). Therefore, our results support our assumptions.

We also discovered many other 3D motifs that cover almost all the positions in a group of faces in a TM region but are not genuine. Some of them indeed characterize certain subgroups of ORs with similar ligand binding profiles and thus could correspond to regions involved in ligand binding. It is not clear exactly which properties confer the potential functional significance to the corresponding regions. We specifically identified some of these 3D motifs as hydrophilic ones. They could correspond to ligand-binding sites, which might be less hydrophobic not having to face the very hydrophobic lipid bilayer frequently, especially for those more hydrophilic ligands. In general, it will be interesting to investigate what other properties might justify the functional significance of the corresponding regions.

We generally obtained many more 3D motifs for each group of faces and thus for each TM region than 1D motifs. We hypothesized that a higher level of conservation might prevail in the 3D space than in the 1D space. The large numbers of 3D motifs does not exactly attest to such an assumption. The deterministic, non-statistical representation of these motifs as well as the limited, face-based scope of the 3D sub-databases are some factors that potentially result in the large numbers of these motifs. Other factors also might be involved. For example, the type of conservation captured by these 3D motifs might not highly resonate with the kind of conservation that indicates the sharing of functions in the 3D space. We represented the 3D motifs by regular expressions made of amino acids, similarity classes of the amino acids, and don't-care, with the similarity classes defined based on a substitution matrix meant to characterize the evolutionary

properties of amino acids. It is possible that the evolutionary properties do not entirely dictate the development of functions. Therefore, we may consider other properties of the amino acids, such as biochemical ones, in various ways in the future.

Overall, we examined the 3D motifs at a relatively high level. Since the 3D databases overlap substantially, there is likely to be some redundancy in the resulting 3D motifs. We may consider how to perform reasonable filtering and selection to generate a condensed compilation of maximum information content with minimum redundancy in the future. Once we reach that point, we may continue to study the corresponding taxonomy of classes with respect to the biologically meaningful classes.

3.3 Structures and Post-Translational Modifications of ORs

Amine GPCRs are believed to adopt similar tertiary structures as rhodopsins, since the crystal structure of bovine rhodopsins can be used to explain convincingly many of the known experimental results on amine GPCRs. We hypothesized that ORs might also adopt similar tertiary structures as rhodopsins, and the following lines of evidence support this conjecture. First, there is a region in the C-terminal of most ORs that possesses certain features specific to an alpha-helix and also maps well to the H8 helix in bovine rhodopsins. Second, rhodopsins show kinks in certain TM regions, and some conserved residues in amine receptors that could adopt such conformations, such as the conserved glycine in TM1 and the conserved proline in TM2, are conserved for the appropriate positions in most ORs, [8] (Fig 1a). Therefore, the results support our assumption.

The desensitization of an OR is believed to be carried out through phosphorylation. GPCR-specific kinases (GRKs) and second messenger-dependent kinases (PKAs and PKCs) are known to play roles in the desensitization. We identified a potential GRK site in IC3 of some ORs and a potential PKC site at the C-terminal of most ORs without identifying a strong PKA site. PKAs and PKCs are known to modulate the desensitization of ORs following cAMP or IP3/DAG signals, respectively [11][12]. PKAs potentially phosphorylate phosducins, which in turn attract GRK3s to activated ORs [13]. The fact that we did not find a convincing potential PKA site further confirms that PKAs might not phosphorylate ORs directly. On the other hand, the IP3/DAG pathway in olfactory neurons may not be critical for the activation of olfactory neurons [17], while it is conceivable that DAGs activate PKCs, which in turn phosphorylate ORs. The fact that we identified certain potential PKC sites further confirms that the modulation of

desensitization, rather than the activation of transduction, might be the role of the IP3/DAG pathway in olfactory neurons.

Our results indicate that the N-glycosylation and PKC sites are present predominately in Class II ORs and tend to co-occur, while both tend to be absent in Class I ORs. This suggests that Class I is most likely a more ancient cluster and such sites might have evolved after the split into Class I and Class II. It also suggests that there might be significant differences in regulation between Class I and Class II.

3.4 Ligand Binding of ORs

The huge number of odorant-OR interactions calls for a large number of ligand binding sites in ORs. This suggests that the more variable regions in ORs, TM4 and TM5, are more likely to contain ligand binding sites (**Figure 18a**). Some other variable regions, the middle part of TM3, and small parts of TM6, EC2, and EC3, could also participate in ligand binding. The second half of the N-terminal, the middle of IC1, and the second half of the C-terminal, which are all rather variable, are probably not involved in ligand binding, especially because they are very far from TM4 and TM5.

In 3D space, experimentally determined ligand binding sites in rhodopsins and small-molecule Family A GPCRs are generally located in the area corresponding to the second pocket in ORs [29]. For ORs, only some of the variable positions are in the second pocket, while the others, especially those in TM4 and TM5 (**Figure 18a**, **Figure 25b**), tend to face the lipid bilayer. This raises some doubt about whether the second pocket indeed is a ligand binding pocket for ORs and whether the outward faces in TM4 and TM5 might also be involved in ligand binding, which could be justified as follows. First, the second pocket may be where the ligands bind, but certain residues in these variable outward faces could affect the shape of this ligand binding pocket and thus participate in ligand binding indirectly. Second, these variable outward faces may actually be part of a ligand binding pocket. Odorants, which are generally hydrophobic, may diffuse through the lipid bilayer and reach these outward faces [58]. Alternatively, ORs may form a ligand binding pocket with themselves or with other proteins by forming a dimer.

While it is not immediately clear what the exact role of the second pocket and that of the group of outward faces in TM4 and TM5 are in ligand binding, we employed various approaches to identify specific positions that might be involved in ligand binding, which turned out to reside mostly in these regions.

We first considered globally variable but locally conserved positions, which are quite likely to correspond to ligand binding sites, since the globally variability coupled with local conservation may directly reflect ligand specificity. We identified four positions in the second pocket, two in the outward faces in TM4, and a few in EC3 that appears sufficiently close to the others to interact with them. Next, we considered the positions that match the mutation residues, where the outcomes of point-mutation experiments could be affected by any relevant factor. We identified two in the second pocket and some others, all on the intracellular side of the membrane, pointing to peripheral roles in ligand binding. We also considered the positions in the hydrophilic 3D motifs. Those positions in the extracellular parts of such or any interesting 3D motifs are conceivably more likely to be directly involved in ligand binding. Many such 3D motifs, especially the genuine ones, occur in the second pocket, but some of them occur in the outward faces in TM4 and TM5 as well. In addition, we considered the positions in those 3D motifs characterizing or specific to subgroups of ORs with similar ligand binding profiles. Those for the group of M71 and M72 occur mostly in the outward faces in TM4 and TM5, but the others tend to occur in the second pocket. Finally, we considered the positions in the interesting group of 3D motifs that occur in possibly interacting faces of different TM regions and have almost identical support sets. These 3D motifs all occur in the second pocket.

Overall, our results suggest that ligand binding sites may be present in both the second pocket and the outward faces in TM4 and TM5, and the question would be how they are present in different ORs.

4 Methods

We describe the methods used to obtain and analyze all the results in this section.

4.1 Construction of Complete OR Database

The complete OR database is constructed as follows. 322 intact, full-length human OR sequences from HORDE [31] are included in the database, and 1,010 intact, full-length mouse OR sequences from [87] are also included in the database. The database thus contains a total of 1,332 sequences. For each OR, the putative transmembrane regions are identified via PHD [63], TMHMM [45], or based on sequence alignment.

4.2 Application of CASTOR to Complete OR Database

CASTOR is run on the complete OR database to generate a tree followed by a graph via the structure reorganization procedure. In this case, $N_B = 2$, $N_D = \infty$, $u_1 = 0.7$, $t_1 = 0.7$, $t_2 = 0.9$, $e_3 = 10$, $l = 5$, $w = 6$, $k = 4$, and $-n- = 3$.

Additional processing is performed as follows. In reorganizing the tree into a graph, some unusually long motifs, which have been discovered from relatively small subsets of sequences in most cases, are forced into motif groups of their own. For each motif group such that the consensus motif occurs in almost the entire OR family, the consensus motif is replaced by the motif with the largest support set in the motif group. Each of the consensus motifs is then assigned one or more non-overlapping sets of consecutive secondary-structure regions based on the structural distribution of its occurrences. It is then refined into one or more new motifs as follows. For each assigned set of consecutive secondary-structure regions, the set of occurrences in this set of regions is used to construct a new profile HMM using HMMER. The result is a set of new motifs, each associated with a set of consecutive secondary-structure regions and limited to occur in only the associated set of regions. The set of new motifs is then compared with itself in the same way that the set of original motifs was compared with itself to get a list of new motif groups. For each new motif group, one motif is manually selected as the consensus motif and iteratively refined twice using HMMER. The set of final consensus motifs in terms of their occurrence sets is considered as the final output.

4.3 Construction of Global Multiple Alignments and Global Consensus Sequences

The global multiple alignments for the Class I subset, the Class II subset, and the complete OR database were constructed using ClustalX (http://inn-prot.weizmann.ac.il/software/ClustalX.html). The values for the parameters were: 35 for gap opening penalty and 0.75 for gap extension penalty for pairwise alignment, 15 for gap opening penalty and 0.30 for gap extension penalty for multiple alignment, and default values for all the other parameters.

A profile HMM was constructed based on each global multiple alignment. A global consensus sequence was then constructed by running HMMEmit on the profile HMM.

4.4 Construction of Snake Diagrams and 2D EC Cross-Section

The snake diagrams for the Class I subset and the Class II subset were constructed as follows. The global consensus sequences corresponding to the Class I subset and the Class II subset were aligned to the bovine rhodopsin sequence by ClustalX. Based on the alignment and the known TM regions in a bovine rhodospsin, the corresponding TM regions in the consensus sequences were determined and plotted.

The 2D EC cross-section is constructed as follows. An extracellular seven-TM cross-section of the bovine rhodopsin 3D structure was taken from [8]. The global consensus sequences corresponding to the complete OR database was aligned to the bovine rhodopsin sequence by ClustalX. Based on the alignment, the face containing the most conserved residue of each TM region was determined. The other residues were assigned to faces by assuming an ideal helix with 100 degrees between each pair of adjacent residues.

4.5 Computation of Feature Values for Sequences and Motifs

For each position in the global multiple alignment, the variability index is calculated as described in [58]. For each position, the hydrophobicity index is calculated as follows. Each of the 20 amino acids is assigned a hydrophobicity scale as defined in [46]. The hydrophobicity index for the position is then computed as the average hydrophobicity scale over all the amino acids at the position. For each position, the side-chain volume is calculated as the average of the side-chain volume as described in (http://www.imb-jena.de/IMAGE_AA.html).

For each position in a motif, the variability index is also calculated as described in [58].

4.6 Construction of Motif Alignment

In **Figure 19,** the lengths of the TM regions and loops in the global multiple alignment for all the ORs were used. The position of each motif was computed based on the best match of the motif with respect to the secondary-structure information of the corresponding OR.

4.7 Construction of Phylogenetic Tree

A phylogenetic tree for all the sequences in the complete OR database is constructed as follows. The global multiple alignment is used as input to PAUP*4.0 beta (Sinauer Associates, Inc., Sunderland, MA.), and the majority-rule consensus Neighbor Joining (NJ) tree from 1000 bootstraps is obtained.

4.8 Computation of Specificity to a Group of ORs with Respect to a Bigger Database

Suppose that there is a database D and it could be partitioned into group A and group B. The specificity of a motif to A with respect to D is calculated as follows.

$(nA\_sup / n\_sup) / (nA\_D / nD)$,

where nA_sup is the number of ORs in the support set that belong to A, n_sup is the number of ORs in the support set, nA_D is the number of ORs in D that belong to A, and nD is the number of ORs in D. A large value indicates that the motif is specific to A, while a small value indicates that the motif is specific to B.

D is usually the complete OR database. In computing the specificity of a motif to the Class I subset or the Class II subset with respect to the complete OR database, A is the Class I subset and B is the Class II subset. In computing the specific of a motif to the subset of mouse OR sequences or the subset of human OR sequences, A is the subset of mouse OR sequences, and B is the subset of human sequences.

D could also the database of GPCR sequences. In computing the specificity of a motif to the complete OR database with respect to the database of GPCR sequences, A would be the complete OR database and B would be the subset of non-OR GPCR sequences. However, we show only the number of matches against the subset of non-OR GPCR sequences for each motif in this chapter. The subset of non-OR GPCR sequences is constructed by taking the list of GPCR sequences available from SWISS-PROT (Release November, 2001) and then removing the OR sequences and fragments. The result is a set of 846 full-length, non-OR GPCR sequences.

Each motif is matched against the set of 846 non-OR GPCR sequences using HMMSearch the way it is matched against the complete OR database.

4.9 Comparison between Different Groups of ORs

A group A of ORs and another group B of ORs can be compared in terms of the support identity constraint (see Chapter 2). A set identity is declared if they satisfy the support identity constraint. A and B can also be compared in terms of the support containment constraint. A set containment is declared if they satisfy the support containment constraint.

In identifying two motifs with identical support sets, A is the support set of one motif, and B is the support set of the other motif. In this case, t_2 in the support identity constraint is set to 0.9. A list of motifs with identical support sets is obtained as follows. All the motifs are considered one by one. Each motif is first associated with itself. For each motif that has not been marked, the support set of this motif is compared with that of every other motif that has not been marked. If they satisfy the support identify constraint, the second motif is marked and associated with the first motif instead. Eventually, all the motifs associated with one motif constitute a list of ORs with identical support sets.

In identifying two motifs such that the support set of one contains that of the other, A is the support set of one motif and B is the support set of the other motif. In this case, t_2 in the support containment constraint is set to 0.9.

In identifying identities between the support sets of motifs and the clusters in the phylogenetic tree, A is the support set of a motif and B is a phylogenetic cluster as long as both of them have at least three members. In this case, t_2 in the support identity constraint is set to 0.7.

In identifying two 3D motifs with identical support sets, A is the support set of a 3D motif, and B is the support set of the other 3D motif. In this case, t_2 in the support identity constraint is set to 0.7. A list of 3D motifs with identical support sets is obtained via the same procedure as with the 1D motifs.

In identifying two 3D motifs such that the support set of one contains that of the other, A is the support set of one 3D motif, and B is the support set of the other 3D motif. In this case, t_2 in the support containment constraint is set to 0.7.

A list of 3D motifs with "comparable" support sets is obtained as follows. Two groups A and B of 3D motifs have comparable support sets if the union of the support sets of the motifs in A and that of the support sets of the motifs in B satisfy the support identity constraint, and for every pair of 3D motifs such that one is in A and the other is in B, their support sets satisfy either the support identity constraint or the support containment constraint. Every 3D motif M such that the support set of M contains multiple support sets of other 3D motifs and is identical to the union of the contained support sets satisfy the support identity constraint is identified. In this case, t_2 in the support identity constraint is set to 0.7. All such cases are then analyzed together in a recursive way to determine the final list of 3D motifs with comparable support sets.

4.10 Comparison between Discovered Motifs and Known Motifs for Known Protein Families and Domains

Each motif is compared with each motif in Blocks+ [38], which consists of Blocks and some other database. Blocks [38] is a database of motifs represented by blocks, which are made via computational means from the most conserved regions in groups of proteins documented in Prosite [5]. Prosite is a database of protein families, each with motifs that are specific to the protein family and have been shown to be biologically significant. A motif specific to a protein family in Prosite is not used in any way to make the corresponding blocks, and the motif may or may not be contained in one of the corresponding blocks. Blocks is supplemented with other databases of motifs that are not represented in Blocks. These databases are Prints [3], where each entry corresponds to a protein family with multiple conserved domains, and Pfam (http://pfam.wustl.edu/), ProDom [72], and Domo [32][33], where each entry corresponds to a single protein domain. The result is Blocks+ (Version 13.0) containing 11,853 blocks from 2,608 families.

A block is constructed for each motif using Block Formatter (http://blocks.fhcrc.org/blocks/block_formatter.html). Ten blocks corresponding to the ten motifs are compared with all the blocks in Blocks+ each time using LAMA [57]. All the matches reported by LAMA are compiled and those such that the overlap is at least 70% of both the length of the motif and the length of the motif in Blocks+ are specifically studied.

Each motif is also compared with each motif in Prosite. As mentioned earlier, a motif in Prosite may or may not be represented in Blocks. Furthermore, a motif in Prosite may be excessively short, thus escaping the list of matches identified by LAMA. Prosite (Release 16 of July 1999 and updated up to October 2001) contains a total of 1,034 documentation entries that describe 1,374 motifs.

All the Prosite motifs are matched against the set of occurrences of all the motifs all at once using PPSEARCH (http://www2.ebi.ac.uk/ppsearch/). All the matches reported by PPSEARCH are compiled and those discovered-motif/motif-in-Prosite pairs such that the motif in Prosite matches at least 50% of the occurrences of the motif are specifically studied.

4.11 Comparison between Discovered Motifs and Known Mutation Residues in GPCRs

Each motif is checked against LMD, a local mutation database based on tGRAP [10] and GPCRMD [80]. Each entry in LMD corresponds to a single-deletion or single-substitution experiment on a residue in a GPCR sequence. There are 336 entries covering 99 GPCR sequences in LMD.

Each motif is matched against the set of 99 GPCR sequences using HMMSEARCH. Any part of a GPCR sequence is considered as an occurrence of the motif if it has an e-value below the motif-specific threshold. Each occurrence of the motif is then checked to see whether it contains any residue with an entry in LMD. If it does, the corresponding position of the motif is associated with the functional effect of the appropriate mutation experiment. It is possible that one position of a motif is associated with the functional effects of multiple mutation experiments. All the associations made this way are compiled, and those such that the occurrence of the motif has a relatively high e-value and an appropriate secondary-structure location and the residue with an entry in LMD agrees with the consensus residue for the position in the motif are specifically studied.

4.12 Identification of GVLC Positions

For a selected region, every position associated with the region in the OR multiple alignment is considered. If the variability index of the position is ranked among the top 35% with respect to the distribution over the OR multiple alignment and for some 2/3 of the corresponding positions in the appropriate OR motifs, the average variability is ranked among the lowest 35%, the position is considered as a GVLC position.

4.13 Construction of Complete 3D Database

The complete 3D database is constructed as follows. There are seven TM regions in an OR and eighteen faces in a TM region. All the faces in each TM region are first ordered, as shown in **Figure 25**b. The alpha-helices in different ORs corresponding to the same TM region are then oriented in a consistent way by having those residues mapped to the same position in the global multiple alignment reside in the same face and serve as anchors in positioning the other residues in the alpha-helices. The different faces in a TM region with respect to a specific alpha-helix are then referenced in a consistent way based on **Figure 25**b.

For each group of five consecutive faces in each TM region of every OR, a 3D sequence is constructed by enumerating all the residues in the faces in the natural linear order. Each 3D sequence generally has five to seven residues. All the 3D sequences corresponding to a specific group of faces in some TM region constitute a 3D sub-database. Each 3D sub-database contains approximately 1332 3D sequences (secondary-structure information not available sometimes). There are 126 (7 x 8) 3D sub-database in total, and they constitute the complete 3D database.

4.14 Application of CASTOR to 3D Sub-Database

CASTOR is run on each 3D sub-database to generate a tree with the following adjustments. After discovering a statistically significant motif by running SPLASH, CASTOR does not refine the representation of the motif. CASTOR determines the occurrence set of the motif by directly matching the regular expression against the current database. In this case, $l = 5$, $w = 5$, $k = 4$, and $-n- = 3$.

4.15 Computation of Feature Values for 3D Motifs

For each motif, the hydrophobicity index is calculated as follows. Each of the 20 amino acids is assigned a hydrophobicity scale as defined in [46]. A similarity class consisting of multiple amino acids is then assigned the average hydrophobicity scale over all the amino acids in the class. The hydrophobicity index is then computed as the average hydrophobicity scale over all the tokens in the motif.

For each motif, which corresponds to some TM region, the hydrophobicity difference value is calculated as follows. For each occurrence of the motif, a first hydrophobicity index is computed as the average hydrophobicity scale over all the residues in the TM region except for those in the occurrence of the motif, and a second hydrophobicity index is computed as the average hydrophobicity scale over all the residues in the occurrence of the motif. The hydrophobicity difference value of the motif is then computed as the difference between the first hydrophobicity index and the second hydrophobicity index.

For each motif, which corresponds to some TM region, such that the number of residues in the natural linear order between every two consecutive positions in the motif is constant over all the 3D sequences in the support set, the genuineness value is computed as follows. It is computed as the average variability index over all the positions in the TM region except for those token

positions (represented by tokens rather than don't-cares) in the motif for the support set of the motif.

The distribution of genuineness value for these motifs is compared with the distribution of variability index for highly conserved regions and that for the random background based on the complete OR database. A sample for the variability index of a highly conserved region is created for every motif by taking the average variability index over all the token positions in the motif for the support set of the motif. A sample for the variability index of the random background is created for every motif by taking the average variability index over all the positions in C random strings of length sixteen, which is approximately the number of positions in the TM region except for those token positions in the motif, where C is the support of the motif.

4.16 Comparison between 3D Motifs and 1D Motifs

Every 3D motif is compared with every motif in terms of the motif identity constraint. A motif identity is declared if they satisfy the motif identity constraint. In this case, u_1 is set to 0.7, and t_1 is set to 0.7.

Every 3D motif is also compared with every motif in terms of the *motif set-containment constraint*, which is modified from the motif identity constraint by requiring that the specified amount of overlap be for at least the support coefficient of the smaller of the support set. A motif set-containment is declared if they satisfy the motif set-containment constraint. In this case, u_1 is set to 0.7, and t_1 is set to 0.7.

Every 3D motif M such that the support set of M contains multiple support sets of motifs and is identical to the union of the contained support sets of motifs is identified, and similarly the other way around. In this case, t_2 in the support identity constraint is set to 0.7. Every 3D motif that is not identical to any other motif is also identified. In this case, t_2 in the support identity constraint is set to 0.5.

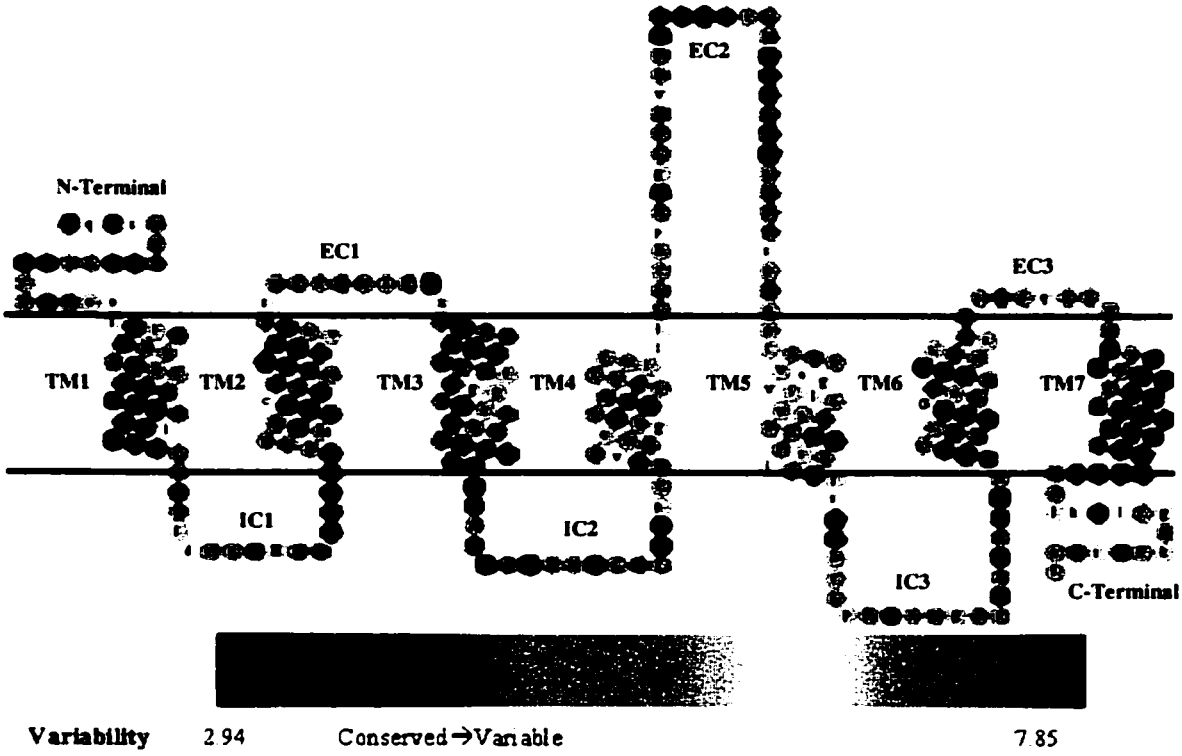


Figure 17: One-Dimensional Variability Plot. This figure shows the snake diagram of the consensus sequence for the complete OR database. The two black, horizontal lines delimit the region corredsponding to the cell membrane. The letter enclosed in each circle indicates the consensus residue for the corresponding position. The color of each circle indicates the variability index of the corresponding position.

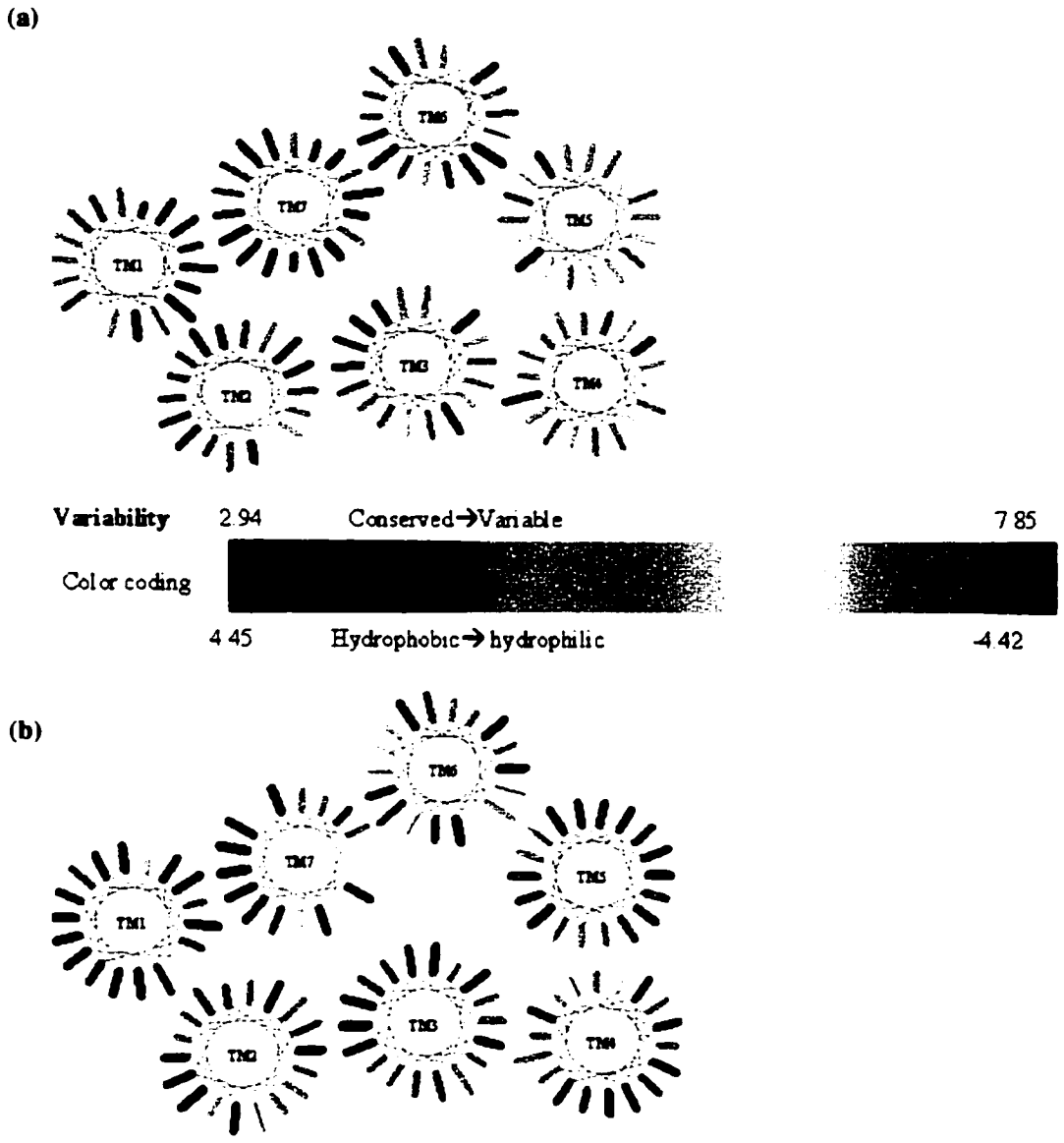


Figure 18: Two-Dimensional Variability and Hydrophobicity Plots. This figure shows an extracellular cross-section of the seven TM regions of the consensus sequence in the 3D space. In (a), each bar indicates the variability index of the extracellular position, and each line, if any, indicates the variability index of the intracellular position. The length of each bar or line indicates the average side-chain volume for the corresponding position. In (b), it is similar except that variability index is replaced by hydrophobicity index.

Table 3: One-Dimensional Motif Information. This table summarizes the relevant information about all the 1D motifs. It is split into two parts. The first part contains the first six columns (excluding the N# column), and the second part contains the last three columns. The N# column indicates the motif index. The Support column indicates the support. Those entries corresponding to the motifs characterizing the complete OR database are highlighted in yellow. The Consensus column indicates the consensus sequence for the corresponding profile HMM. The 2nd_Struct column shows the estimated secondary-structure location of occurrence. T, C, and E stand for transmembrane, cytoplasmic, and extracellular, respectively, and the number enclosed in parentheses indicates the starting position with respect to the corresponding region. The Class_I/Class_II column shows the normalized ratio that indicates the specificity to the Class I subgroup or Class II subgroup. Those entries corresponding to the motifs specific to the Class I subgroup are highlighted in orange, and those corresponding to the motifs specific to the Class II subgroup are highlighted in green. The Mouse/Human column indicates the normalized ratio that indicates the specificity to the human subgroup or the mouse subgroup. Those entries corresponding to the motifs specific to the mouse subgroup are highlighted in pink. The GPCR column indicates the number of matches in the database of non-OR GPCR sequences. Those entries corresponding to the motifs specific to the complete OR database with respect to the database of GPCR sequences are highlighted in blue. The PROSITE column shows the corresponding matches with PROSITE entries. The BLOCKS column shows the corresponding matches with BLOCKS entries. The TGRAP column shows the corresponding matches with TGRAP and GPCRDB entries.

N#	Support	Consensus	2nd_Struct	Class_I/Class_II	Mouse/Human	GPCR
1	1327	maydryvaickplry	T3(20)_T4(0)	0.959	0.998	348
2	1328	pllnpliyslrn	T7(10)_C4(1)	0.976	0.994	205
3	1327	sLhtPmYffLs	C1(7)_T2(6)	0.959	1.001	59
4	1330	kalstcashllvv	C3(16)_T6(8)	0.975	0.998	
5	1313	ilvSYvlllsti	T5(14)_C3(2)	0.987	0.98	116
6	1323	iYlvtllGNlllilli	T1(8)_C1(0)	0.962	0.994	282
7	1328	rLpFCgsnvlnHFFCdipplIkLaC	E3(2)_E3(26)	0.97	1	
8	1307	tlsfagCltQlf	E2(14)_T3(5)	0.962	0.997	
9	1314	DicyssvtvPkmL	T2(11)_E2(4)	0.921	1.008	
10	1145	NqtsvteFiLIGl	E1(4)_E1(16)	0.32	0.959	
11	1160	MsprvCvlLvagsW	C2(15)_T4(9)		0.96	105
12	1108	KdVKgAlkkllgrkks	C4(3)_C4(18)	0.234	0.945	74
13	947	dkvvsvfYtvv	E4(11)_T7(7)		0.967	15
14	1099	lfYgtaifmY	T6(13)_T6(22)		0.949	
15	472	lFvlfLv	T1(0)_T1(6)		1.009	15
16	460	kiksaeGrk	C3(5)_C3(13)		0.834	
17	1301	FfhffggtEcfLLav	T3(6)_T3(20)	0.876	0.995	14
18	263	PkSshSldq	E4(2)_E4(10)		0.674	
19	37	nlvsslv	T5(8)_T5(15)		0.557	
20	135	fvSLivTillF	T4(11)_E3(3)		0.947	29
21	1038	sDTsinElvifilagf	E3(29)_T5(7)	0.235	0.977	12
22	883	ElvifilaGfiivvtlli	E3(33)_T5(13)	0.142	0.915	59
23	1153	lqvlLFvlFLliYllTll	E1(22)_T1(14)	0.352	0.948	57
24	449	lnSlihTsltf	T4(13)_E3(0)		0.955	
25	336	sDthinelv	E3(29)_T5(0)		0.933	

Table 3: Continued.

26	122 PelqvP	E1(20)_E1(25)	0	0.473	5
27	51 niivPslt	T5(6)_T5(13)		0.323	
28	898 vgGfthslihTlltf	T4(10)_E3(1)	0.158	0.96	
29	559 iFvvgvlvlliPlll	T5(1)_T5(15)	0.253	0.989	48
30	283 ggeeilLlv	T3(10)_T3(18)		1.282	
31	662 lgateclllav	T3(9)_T3(19)		0.972	
32	164 vgGflnas	T4(7)_T4(14)		0.855	
33	14 SiNvlsmav	E3(31)_T5(3)		1.473	
34	197 SlvqtsLtl	T4(16)_E3(1)		1.11	
35	374 iNElvifivag	E3(33)_T5(6)		0.838	
36	110 dllsekK	E2(6)_E2(12)		1.275	
37	60 SshSmDt	E4(4)_E4(10)		0.481	
38	90 LiHtlfla	T4(16)_E3(2)	0.349	1.1	
39	97 GFLiilpPliL	T4(12)_T4(22)	1.134	0.723	30
40	190 GvlvilvPfll	T5(5)_T5(15)		1.042	15
41	146 nelvif	E3(34)_T5(2)		0.763	
42	55 Pssnnstek	T6(22)_E4(7)		0.375	
43	495 gtEcvLLav	T3(11)_T3(19)		1.008	
44	81 DllsekK	E2(0)_E2(6)		1.273	
45	80 rkllkKKisl	C4(11)_C4(20)		1.031	
46	13 sPetKK	E4(7)_E4(12)		1.269	
47	216 svfillvPtsL	T5(5)_T5(15)		1.088	10
48	329 SleqdKvvS	E4(7)_T7(1)		0.652	
49	17 mAFledg	E1(0)_E1(6)			
50	124 AsgFtlSfv	T4(10)_T4(18)		0.831	34
51	284 AliHTvltf	T4(16)_E3(1)		0.842	
52	961 fsdqpelqlfLFvlFL	E1(16)_T1(4)		0.955	11
53	246 ksaeGRk	C3(11)_C3(17)		0.637	
54	8 qqaddqDmidS	E4(5)_T7(1)		1.031	
55	120 iAvfTlitLv	T5(2)_T5(12)		0.687	46
56	34 ifPpliLgL	T4(16)_E3(1)	0.231		
57	71 lsiplCl	T1(0)_T1(6)		1.22	
58	503 mnqklCv	C2(16)_C2(22)		0.771	
59	35 lvgPLcl	T5(9)_T5(15)		1.061	15
60	338 ileLlviansGlisv	E3(30)_T5(5)	0.139	0.927	11
61	453 ggteillLv	T3(10)_T3(18)		1.056	
62	456 sekpelqllf	E1(17)_T1(0)		0.814	
63	139 lAvfTLivtLil	T5(2)_T5(13)	0.848	0.801	72
64	1015 vavfytvvTpvl	T7(1)_T7(12)	0.124	0.987	12
65	112 nlytlvP	T7(6)_T7(12)		1.178	
66	166 FYvPaiflsltHRFGkhvpplv	T6(9)_T7(1)		1.217	
67	79 iltesvvik	C2(14)_T4(0)		1.044	
68	149 hlWisiP	E1(25)_T1(3)		1.248	

Table 3: Continued.

69	71	vPkvvHi	E4(9)_E4(15)	1.22	3
70	104	vnsiyGI	E3(33)_T5(2)	1.229	13
71	245	lpskeaR	C3(5)_C3(11)	1.443	0.859
72	66	lqPvlFvl	E1(22)_T1(4)	0.595	1.25
73	67	vlplvFll	T4(13)_E3(1)		1.231
74	18	illDViL	T5(10)_T5(16)		1.375
75	786	MeggNqTsVtEF	E1(0)_E1(11)		0.967
76	97	vlPlpfLlk	T4(15)_E3(0)		1.233
77	7	Mlqpssmsevtnythdpfy	E1(0)_E1(18)	4.49	1.768
78	63	mllnksetEVTE	E1(8)_E1(19)	0.967	0.825 12
79	216	NsiYGLfvalltvg	E3(31)_T5(8)	5.566	1.012 16
80	495	siYGLfVilsvglDlll	E3(32)_T5(13)	2.588	1.066 100
81	351	lFlviyl	T1(3)_T1(9)		0.822
82	168	GLfvalstvglDsll	E3(35)_T5(6)	6.455	1.08 36
83	134	lglivvaNS	E3(31)_T5(0)		1.231
84	717	viFilavlillspll	T5(1)_T5(15)	0.449	0.903 37
85	118	akesvdlNKvVs	E4(0)_T7(2)		0.489
86	774	iGflhslihTllt	T4(11)_E3(0)		1.044

Table 3: Continued.

N#	PROSITE	BLOCKS	TGRAP (mutation database)
1		GPCR_Rhodpsn	maydryvaickplry, ACT, AG, G
2	PKC_PHOS(1102)	GPCR_Rhodpsn	
3			
4			
5			ilvSYvillsti, ACT, AG,
6		Integrin_B	iYlvtlGNIIIIIli, AG, desens.
7			
8			
9			
10	N-Glyc(1015)		
11	PKC_PHOS(360)		MsprvCvlLvagsW, AG
12	PKC_PHOS(199)	DnaJ_N, DNAJPROTEIN	
13			
14			
15			
16			
17		ExbD,CbiN,KCHANNEL, ACRIFLAVINRP	
18			
19			
20			fvSLivTillF, AG
21			
22			
23			lqvILFvlFLliYllTll, AG
24			
25			
26			
27			
28			
29		GPCR_Rhodpsn, STAS, CCMCBIOGNSIS , CbiN	
30		UL42	
31			
32			
33			
34			
35			
36			
37			
38			
39		NAHEXCHNGR	
40			
41			

Table 3: Continued.

42 N-Glyc (55)		
43		
44		
45 CAMP_PHOS(44)		
46		
47		
48		
49		
50		
51		
52		
53		
54		
55		iAvfTliitLv, AG
56		
57		
58 PKC_PHOS(186)		
59		
60		
61	UL42	
62		
63		lAvfTLivtLil, AG
64		
65		
66		
67		
68		
69		
70		vnsiyGl, AN
71		
72		
73		
74		
75 N-Glyc same as 9!		
76		
77 N-Glyc same as 9!		
78 N-Glyc same as 9!		
79		NsiYGLfvailtvg, AN
80	TCRTETB, NACHANNEL, SecY, Polysacc_synt	
81		
82		
83		
84	GPCR_Rhodpsn, STAS, CCMCBIOGNSIS , CbiN	viFilavlillspll, AG

Table 3: Continued.

85
86

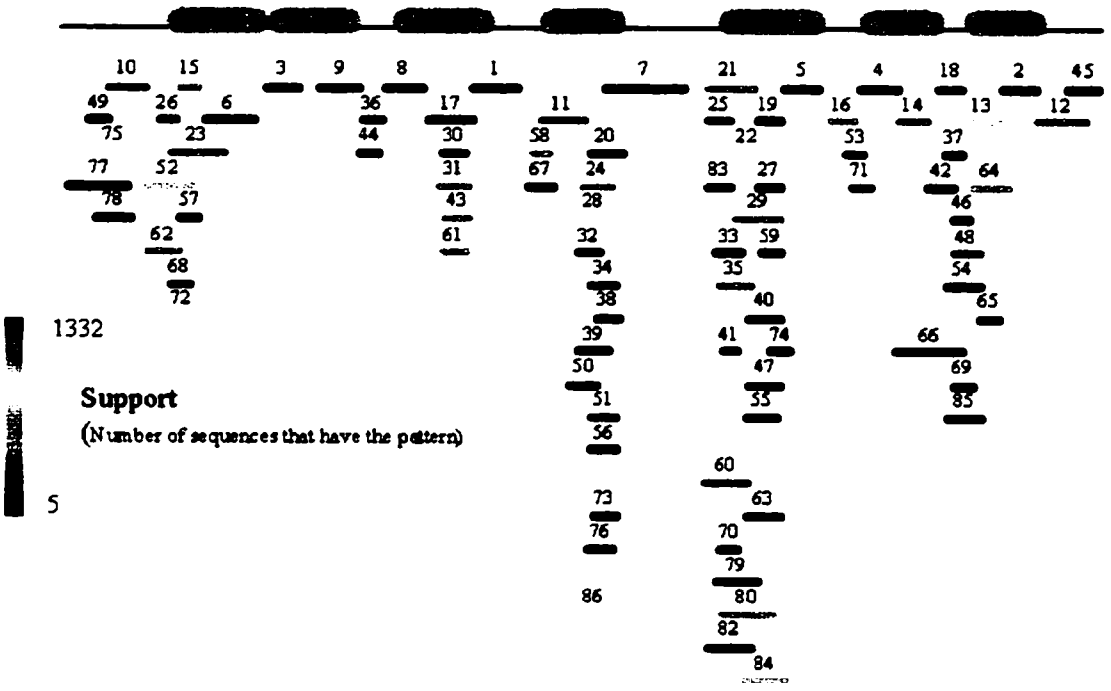


Figure 19: Distribution of Motif Occurrence. This figure shows the location of occurrence for all the 1D motifs graphically. Each colored bar corresponds to a motif whose index is indicated above the line. The color of the line indicates the support of the motif.

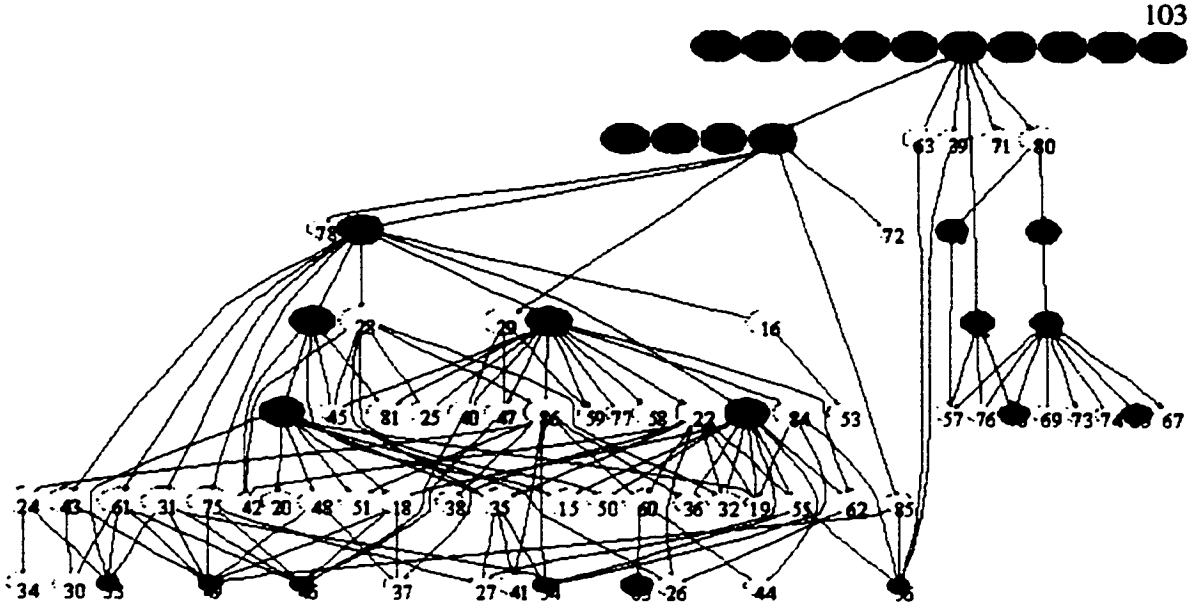


Figure 20: Organization of Classes. This figure shows the structure of the taxonomy as a graph in terms of how the support sets of the 1D motifs are related. Each node represents the support set of a 1D motif, where the index is indicated as the label of the node and the support is indicated through the size of the node. Each downward pointing arrow indicates a containment relationship, where the upper node contains the lower node. The overlap relationships are not shown. A node is colored red if the corresponding support set matches a phylogenetic cluster and pink otherwise.

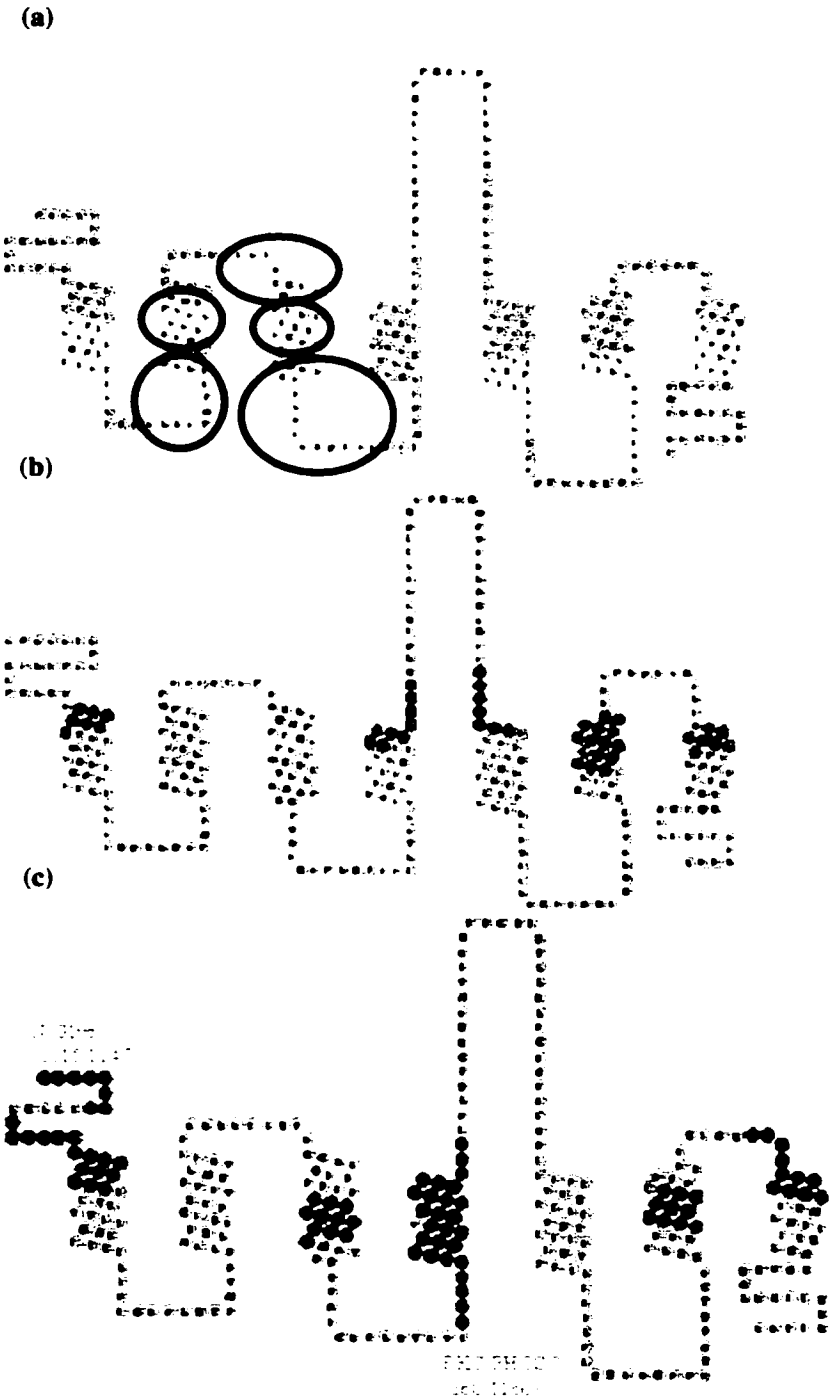


Figure 21: Interesting Sites. This figure shows the snake diagram of the consensus sequence for the complete OR database. In part (a), the locations of occurrence for those 1D motifs that are conserved in the complete OR database are shown in yellow. The black circles are shown to separate those locations that are very close to one another. In part (b), the locations of occurrence for those that are present in more than 50% of the Class I subgroup are shown in red. In part (c), the locations of occurrence for those that are present in more than 50% of the Class II subgroup are shown in green.

Index	Support	Pattern	Position
#38	97	GFLiilpPliL	T4(12)_T4(22)
#39	190	GvlvilvPfll	T5(5)_T5(15)
#56	71	lsiplCl	T1(0)_T1(6)
#57	503	mnqklCv	C2(16)_C2(22)
#58	35	lvgPLcl	T5(9)_T5(15)
#63	139	lAvfTLivtLil	T5(2)_T5(13)
#64	1015	vavfytvvTpvl	T7(1)_T7(12)
#73	66	lqPvlFvl	E1(22)_T1(4)
#74	67	vlplvFll	T4(13)_E3(1)

Figure 22: Similar Motifs. This figure shows a few groups of 1D motifs such that members of each group are similar in composition to one another. The Index column indicates the motif index, as in **Table 3**. The Support column indicates the support. The Pattern column indicates the consensus sequence for the corresponding profile HMM. The Position column indicates the secondary-structure location of occurrence, similar to Table 1.

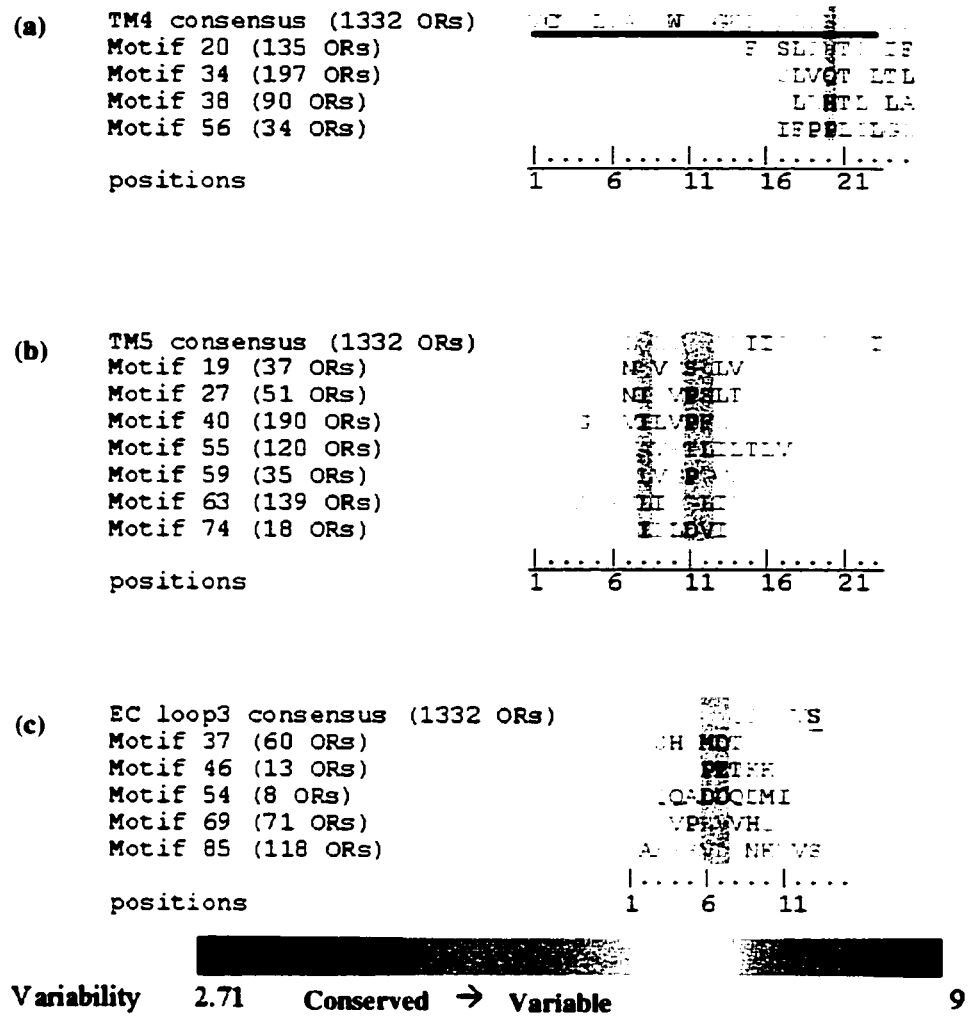


Figure 23: Globally Variable but Locally Conserved Positions. This figure shows the GVLC positions in TM4, TM5, and EC3 in part (a), part (b), and part (c), respectively. In each part, the top row corresponds to the consensus sequence corresponding to the global multiple alignment for a specific region. Each of the other rows corresponds to a 1D motif that occurs in this region and has a relatively small support. The support is indicated as the number enclosed in parentheses on the left, and the consensus sequence of the corresponding profile HMM is indicated aligned with the consensus sequence corresponding to the global multiple alignment on the right. The color of each consensus residue indicates the variability index of the corresponding position across the appropriate subset (complete OR database or the support set of a 1D motif.

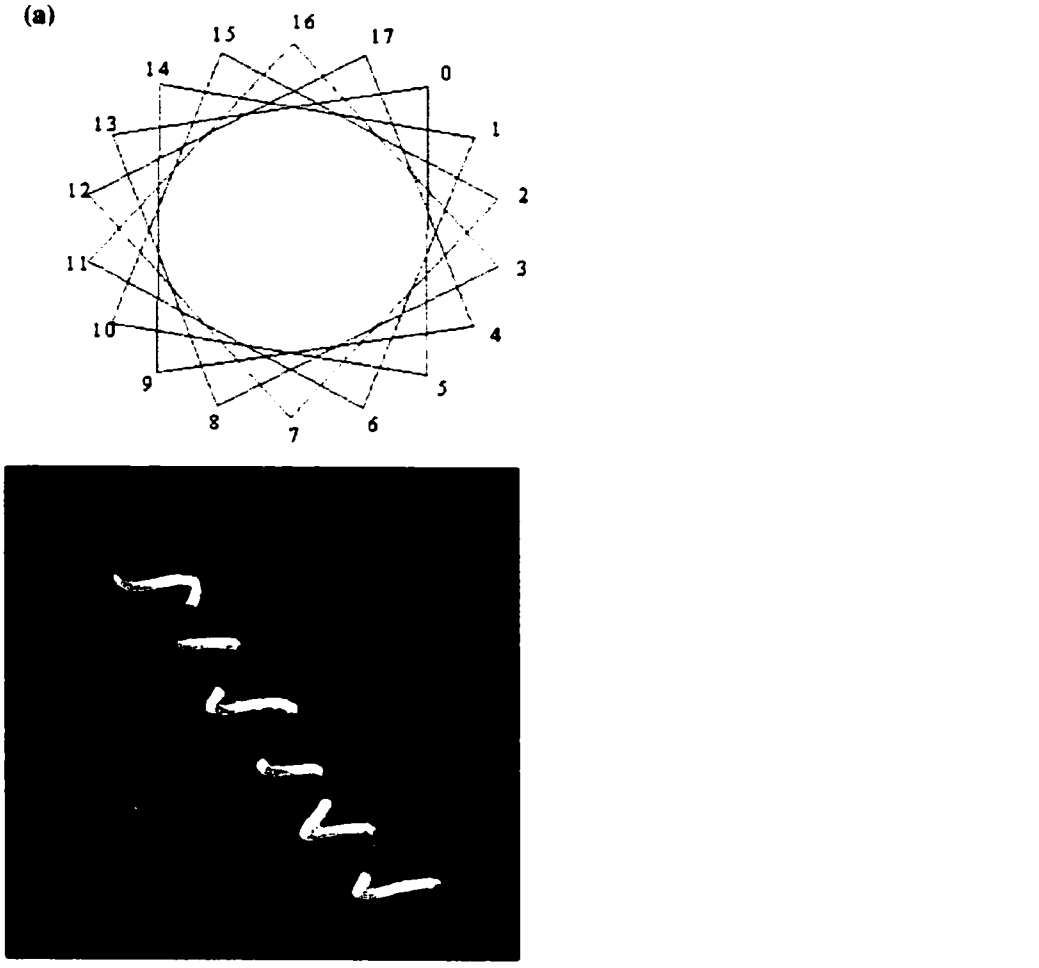
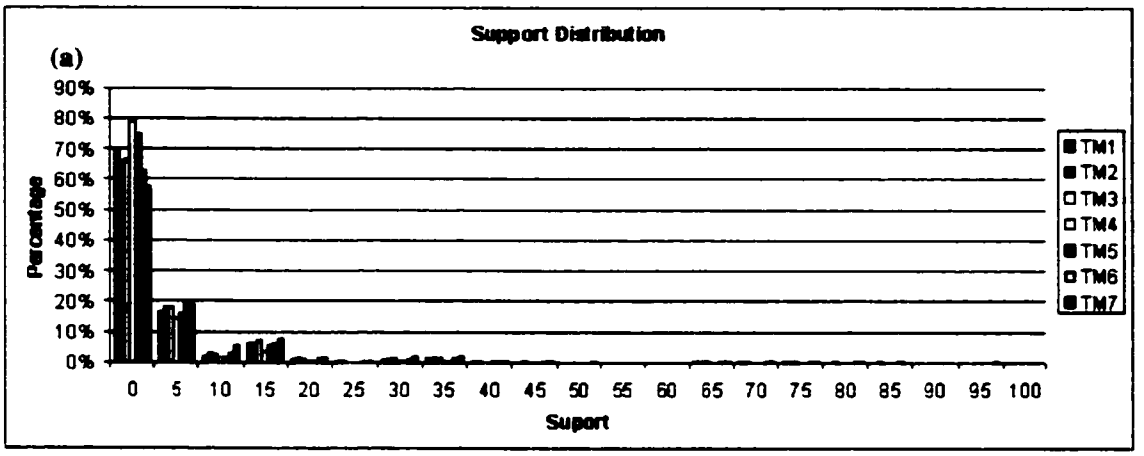


Figure 24: Three-Dimensional Sequence. This figure illustrates the formation of a 3D sequence. Part (a) shows the cross-section of a theoretical alpha-helix as a helical wheel with 18 faces labeled from 0 through 17. Part (b) shows an alpha-helix in the 3D space, where a group of consecutive faces is highlighted in red. Part (c) shows the corresponding protein sequence in the primary structure, where the residues that belong to the group of consecutive faces are highlighted in red as well. This group of residues corresponds to a 3D sequence.



(b)

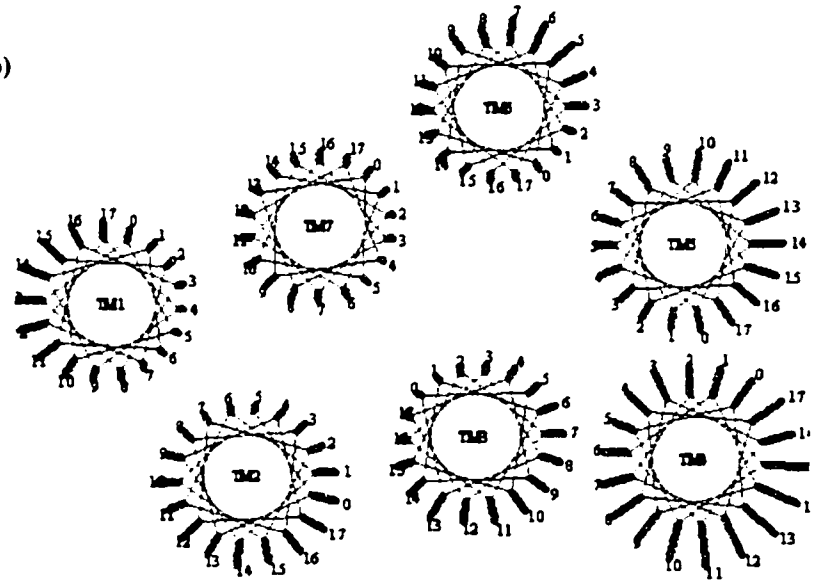


Figure 25: Three-Dimensional Motif Information. This figure shows some information about all the 3D motifs. Part (a) shows the distribution of support for each TM region. Part (b) shows the number of 3D motifs that occur in each face of each TM region, where the length of each bar is proportional to the number of 3D motifs.

(a)

```
Motif HVFGA Occur at TM6, face 15 of 12 ORs
ClrOR1133=========                    lp H lfv V sf F lst G ic A y
ClrOR1221=========                    lp H lfv V ll F lst G lf A y
ClrOR1223=========                    lp H lfv V sl F lst G if A y
ClrOR239*=========                    lp H lfv V sf F lst G ic A y
ClrOR508==========                    lp H lfv V sl F lst G sc A y
ClrOR578==========                    gs H lsv V cl F ygt G le A y
ClrOR661==========                    ss H lit V al F ygf G ll A h
ClrOR714==========                     p H ltv V tl F lss G fv A yl
ClrOR750==========                    ss H lat V ll F ygt G ss A y
ClrOR907==========                    lp H lfv V sl F lst G af A y
OR5AV1============                    ip H lvv V tl F mis G si A y
OR5AY1============                    vp H liv V tv F lvt G av A y
```

(b)

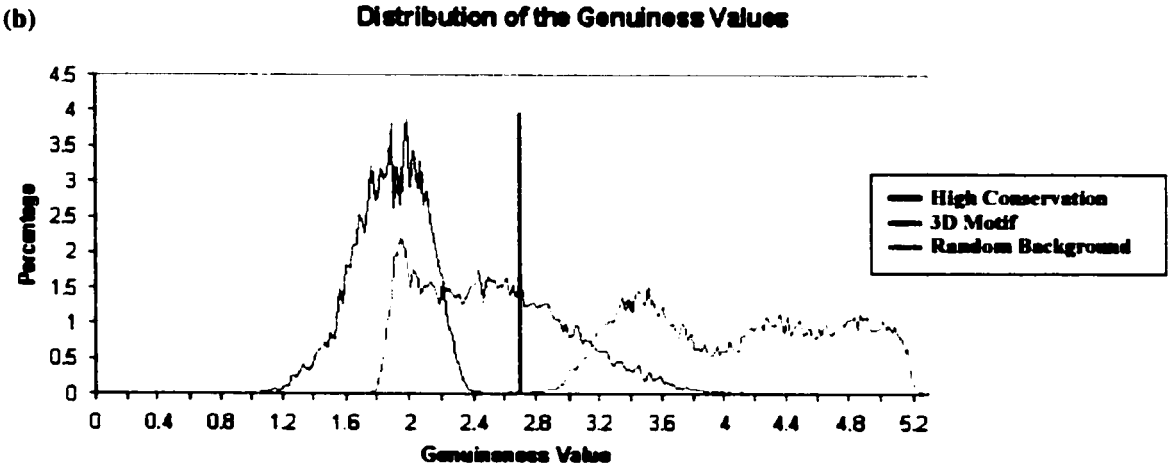


Figure 26: Genuine Three-Dimensional Motifs. This figure shows some information regarding the genuineness of 3D motifs. Part (a) gives an example of a relatively genuine 3D motif. Each row corresponds to an OR, where the name of the OR is indicated on the left and the protein sequence corresponding to some TM region is shown on the right. The residues that belong to a specific group of faces and thus form a 3D sequence are capitalized. Part (b) gives the distributions of genuineness value for three types of regions. The dark blue curve corresponds to highly conserved regions, the pink curve corresponds to the occurrences of the 3D motifs, and the light blue curve corresponds to totally unconserved or random background regions.

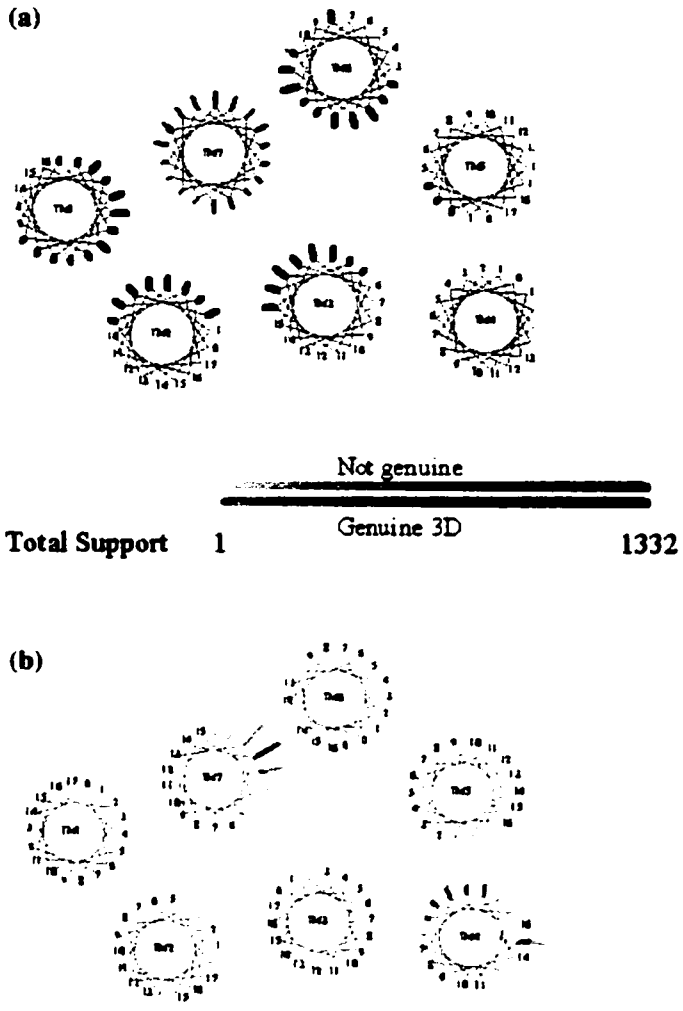


Figure 27: Other Interesting Three-Dimensional Motifs. This figure shows the location of occurrence for those 3D motifs that have large supports (> 150) and have small hydrophobicity (< -1) indices in part (a) and part (b), respectively. The red lines correspond to non-genuine (< 2.7) 3D motifs, while the black bars correspond to genuine ones. The length of each line/bar is proportional to the number of 3D motifs that occur in the corresponding face of the corresponding TM region. The tint of each line/bar is proportional to the sum of supports of all the 3D motifs that occur in the corresponding face of the corresponding TM region.

Table 4: Three-Dimensional Motifs with Large Supports. This table shows some information about those 3D motifs that have large supports. Each row corresponds to a 3D motif. The first column indicates a "!" if the 3D motif is special. The Motif column indicates the motif in its regular-expression representation. The TM and Face columns indicate the corresponding TM region and face. The Hydro column indicates the hydrophobicity index of the motif. The Support column indicates the support. The Genuineness column indicates the genuineness value of the motif.

	Motif	TM	Face	Hydro	Support	Genuineness
pattern_info	[ILMV].[ILMV]TG[ILM]	1	1	1.58	332	3.20
pattern_info	F[ILMV][ST]G[ILMV]	1	2	1.77	541	3.30
pattern_info	[ILMV]F[ILMV].N[ILM]	1	3	1.65	532	3.45
pattern_info	[HFY]Y[ST]N[ILMV]	1	4	-0.50	674	3.52
pattern_info	[FY].Y[ST]N[ILMV]	1	5	-0.20	688	3.40
pattern_info	[FY].Y[ILMV]N[ILMV]	1	6	0.53	797	3.44
pattern_info	[FY]LY[ILMV].[ILMV]	1	8	1.74	601	3.30
!pattern_info	FL[ILM][ILMV].[ILMV]	1	9	2.87	430	3.46
!pattern_info	[ILMV]L[ILMV][ILMV]L	1	10	3.68	291	4.00
pattern_info	[ILMV][ILMV][ILMV]GL	1	17	2.84	168	3.03
pattern_info	FL[ILMV][FY][ST]	2	2	2.04	187	2.85
pattern_info	[ILMV]FL[ILMV][HFWY]	2	3	2.63	242	2.87
pattern_info	FL[NDE][HFWY][ILMV]	2	4	1.21	334	3.12
pattern_info	[ILMV]LL[DE][HFWY]	2	5	1.41	344	2.93
pattern_info	LL[DE][ST][ILMV]	2	6	1.39	419	3.19
pattern_info	[ILMV][ILV]SD[ST]	2	7	0.54	276	2.86
pattern_info	[FY]LS[DE][ST]	2	8	-0.10	337	3.05
pattern_info	YLS[ILMV][ST]	2	9	0.91	243	2.91
pattern_info	QF.[DE][ILMV][MV]	3	0	0.41	535	3.53
pattern_info	QF.[DE][ILM][MV]	3	1	0.38	540	3.54
pattern_info	QF.[DE][ILM][MV]	3	2	0.38	540	3.54
pattern_info	[ILMV].G[DE]LA	3	3	0.88	351	3.47
pattern_info	[ILMV]FG.[ILM]A	3	4	1.87	154	3.19
pattern_info	[ILMV]H.[IMV]LA	3	5	1.59	159	3.09
pattern_info	QFF.[ILMV][MV]	3	16	1.46	402	3.58
pattern_info	QF.[DE][ILMV]M	3	17	0.22	711	3.67
pattern_info	[ILMV]P[ILV]S[IV]	5	2	1.94	211	3.71
pattern_info	[ILMV]P[ILV]Y[IV]	5	3	1.84	218	3.76
pattern_info	[ILMV][ILMV][ILMV]YI	5	4	2.80	261	3.82
pattern_info	C[ILMV]V[FY].[HY]	6	0	1.47	433	3.13
pattern_info	L[ILMV][HFY][ILMV]Y	6	0	1.83	218	2.93
pattern_info	[ILMV]V[FY][ILMV]Y	6	1	2.17	586	3.33
pattern_info	[ILMV][ST]Y[ILMV]Y	6	2	0.77	285	2.98
pattern_info	S[ST][ILM]G[HFWY]	6	8	0.16	231	2.88
pattern_info	SV[ILMV][ST][ILMV]	6	11	1.97	272	2.94
pattern_info	SV[HFWY][ST][ILMV]	6	12	1.12	214	2.92
pattern_info	TH[IV][HFWY].[ILMV]	6	13	0.57	214	3.15
pattern_info	THV[HFWY].[ILMV]	6	14	0.54	212	3.15
pattern_info	H[ILMV]F[ILMV][ILMV]	6	15	2.08	365	3.09
pattern_info	CH[IV][HFWY].[FY]	6	16	0.63	476	3.21
pattern_info	CHV[HFY].[HY]	6	17	0.11	468	3.20
pattern_info	[ILMV]Y[IV]L[IV]	7	13	2.96	213	2.73

Table 5: Three-Dimensional Motifs with Small Hydrophobicity Indices. This table shows some information about those 3D motifs that have small hydrophobicity indices. This is simliar to Table 4.

	Motif	TM	Face	Hydro	Support	Genuineness
pattern_info	FNS.T[RK]	2	0	-1.07	8	2.59
pattern_info	YLDDS	2	3	-1.06	2	2.13
pattern_info	FEYTN	2	4	-1.24	4	2.30
pattern_info	QQGAGT	2	14	-1.12	2	1.93
pattern_info	CQSGH	3	2	-1.08	2	2.04
pattern_info	SGNSS	4	0	-1.26	5	2.12
pattern_info	[LMV]SSA[DQ]	4	1	-1.78	2	3.04
pattern_info	M[ST][ST][HY]H	4	1	-1.01	2	3.07
pattern_info	MTSHH	4	2	-1.20	2	2.09
pattern_info	FSGHQ	4	2	-1.02	9	2.98
pattern_info	TSGDQ	4	3	-1.78	2	1.90
pattern_info	ASG[ND]Q	4	3	-1.28	17	3.17
pattern_info	S[ST]SSH	4	4	-1.27	4	2.77
pattern_info	[ST]SGS[DQE]	4	4	-1.25	2	2.89
pattern_info	[ST]WGSQ	4	5	-1.27	2	2.96
pattern_info	TWGGQ	4	5	-1.18	2	1.97
pattern_info	TANPP	4	6	-1.12	2	2.02
pattern_info	NNCPAS[HY]	4	12	-1.05	2	1.92
pattern_info	KARGP	4	15	-1.72	4	3.47
pattern_info	KSRLP	4	15	-1.40	5	2.57
pattern_info	GGNSS	4	17	-1.18	5	2.14
pattern_info	GQNFSS	4	17	-1.03	2	2.05
pattern_info	DNKTLS	5	0	-1.43	2	2.04
pattern_info	DNKTIS	5	1	-1.32	2	2.04
pattern_info	YCND.S	5	17	-1.10	2	2.15
pattern_info	KNSA.P	6	10	-1.33	4	2.16
pattern_info	HATPN	6	13	-1.44	2	1.91
pattern_info	RNHALP	6	13	-1.20	3	1.95
pattern_info	HV[HY]NY	6	17	-1.21	2	2.77
pattern_info	NNPNY	7	0	-2.68	4	2.21
pattern_info	[ST][ST][ST]NY	7	0	-1.41	79	2.52
pattern_info	QNPNY	7	1	-2.68	11	2.42
pattern_info	[ST][ST]PN[FY]	7	1	-1.17	605	2.59
pattern_info	TTPNY	7	2	-1.56	11	2.42
pattern_info	[NDE][ILMV]PNY	7	2	-1.26	101	2.42
pattern_info	TTPPY	7	3	-1.18	9	2.35
pattern_info	NFPPY	7	4	-1.04	5	2.20
pattern_info	NAPPG	7	5	-1.06	2	2.14
pattern_info	QN[ST]NI	7	16	-1.35	9	2.23
pattern_info	[ND][HY]PN[IV]	7	16	-1.30	13	2.51
pattern_info	NNPNV	7	17	-1.58	4	2.17
pattern_info	QN[ST]NI	7	17	-1.35	10	2.25

CASTOR+

1 Introduction

Given a large group of related protein sequences with abundant sequence data available but little functional information available, if we would like to gain some insight into the functions of the corresponding proteins, we could apply CASTOR to generate a taxonomy of potential functional subsets (see Chapter 2). Each of the subsets that are not negative sets (see Chapter 2) would be characterized by one or more statistically significant motifs, which correspond to potential functional regions (see Chapter 2). Now, given another smaller group of protein sequences that is very likely to be related to the large group, the question is how we could best classify the new group of protein sequences with respect to the old one so that we learn more about the functions of all the corresponding proteins globally. We have two answers as follows.

(1) We could generate a brand new taxonomy, where each class is associated with zero or more motifs (zero only if the class is a negative set and thus a terminal class), by applying CASTOR to the union of the two groups of protein sequences.

(2) We could reuse and extend the old taxonomy into a new one by applying a supervised system that classifies the new protein sequences based on the output of CASTOR.

With any of these options, the result will be an updated taxonomy, where each class is associated with zero or more motifs, for the union of the two groups of protein sequences. In other words, for each protein sequence in the union of the two groups, we will be able to indicate the potential functional subsets to which it belongs and the potential functional regions that it possesses.

In this chapter, we introduce CASTOR+, which could be used to classify new protein sequences based on the output of CASTOR. It is intuitively faster and more convenient to apply CASTOR+ to classify the new group of protein sequences, as described in (2), than to apply CASTOR to the union of the old group and the new group, as described in (1). However, if the protein sequences in the new group possess additional functional traits that are not captured by those in the old group, applying CASTOR+ alone may also lead to degradation of the classification quality. Therefore, we attempt to get an idea of how (1) and (2) might produce different results. Furthermore, for CASTOR+, we investigate two complementary ways to classify new protein sequences, which are expected to demonstrate the tradeoff between output quality and execution

speed. We also examine how CASTOR+ performs compared to a more traditional approach based on pairwise sequence similarities.

In this chapter, CASTOR+ works with the output of CASTOR. It classifies a new group of protein sequences with respect to a previous taxonomy constructed by CASTOR on a separate but related set of protein sequences. Even though a manually constructed and biologically motivated taxonomy (reference taxonomy) may exist, and a CASTOR-generated one may be in good agreement with it, CASTOR+ would not rely on the knowledge of the manually constructed taxonomy. However, it is possible to map the CASTOR-generated taxonomy to the manually constructed one to classify new protein sequences against the latter. This will be used as a criterion to compare the quality of the different classification approaches in our performance evaluation.

2 Methods

As previously discussed, given an initial group of protein sequences, the training set, CASTOR constructs a taxonomy, where each class is either a negative set or it is associated with a set of motifs, represented as profile HMMs. The taxonomy can be represented as either a binary tree or a graph. Given the output of CASTOR and an additional group of protein sequences, the test set, CASTOR+ infers the classes to which each test sequence belongs and which motifs each test sequence matches with respect to the output of CASTOR. We consider two implementations of CASTOR+.

2.1 Full-Taxonomy Implementation

The first implementation of CASTOR+ is the full-taxonomy implementation. It works by considering every class and its associated motifs in the given taxonomy as follows. Each test sequence is matched to each of the motifs in the given taxonomy. CASTOR+ determines whether the test sequence matches a motif the way CASTOR does, based on the corresponding E-value with respect to e_0 or to e_3 when secondary-structure information is available (see Chapter 2). It decides whether to assign the test sequence to a class based how the test sequence matches the motifs associated with a class, which will be discussed further in the next paragraph. In this way, the system infers the potential functional subsets to which the test sequence belongs and the potential functional regions that the test sequence possesses. Specifically, the potential functional

subsets are the classes in the given taxonomy extended by the appropriate test sequences, and the potential functional regions are the occurrences of the motifs in the given taxonomy.

There are a number of approaches discussed in the literature to determine if a sequence matches multiple motifs. Among the most established ones, MAST [4] considers the distribution of the product of *P*-values computed from matching all the motifs. This product has a chi-square distribution with *2n* degrees of freedom, where *n* is the number of motifs, and thus a one-sided fitness test may be used to determine whether the test sequence belongs to the class. Meta-MEME [34] uses a more complex and computationally intensive approach, based on dynamic programming, where individual HMMs become part of a meta-HMM model. Finally, FingerPRINTScan [68] requires that the motifs, if present in the test sequence, occur in the appropriate positional order as determined from the training set. This approach is not consistent with our approach, since when secondary-structure information is available, a match of a motif is allowed to be anywhere in a test sequence as long as the corresponding *E*-value is sufficiently small. Since improving the quality of multiple-motif matching is not one of the goals of this chapter, CASTOR+ uses the simple approach such that a test sequence is assigned to a class if and only if it matches at least one of the motifs associated with the class.

For each test sequence *X*, CASTOR+ infers a set of classes in the given taxonomy to which X should belong. These classes have different sizes and indicate the classification of the test sequence at different levels of the taxonomy. It is also desirable to specify a subset of training sequences, *near(X)*, that are the closest matches to *X*. In this chapter, such classification at the lowest level against a reference taxonomy is used for evaluation purposes.

In this implementation, the inferred classes are not necessarily related in any way. Therefore, CASTOR+ specifically constructs *near(X)* as follows. For each training sequence, a training vector is constructed such that each dimension of the vector corresponds to a distinct motif in the given taxonomy and the corresponding component is the *E*-value computed from matching the motif against the training sequence. For the test sequence *X*, a test vector is also constructed in the same way. CASTOR+ then identifies *near(X)* as the subset of *G* closest training sequences, as determined by the Euclidean distance between the corresponding test vector and each of the training vectors. In this case, *G* is set to one. In this way, CASTOR+ characterizes each test sequence based on the complete set of motifs in the CASTOR-generated taxonomy.

The vector construction may be adjusted or refined in several aspects. It may be appropriate to take the logarithm of each E-value or even convert it into a binary value indicating the matching result to scale down the fluctuations in E-values. It may also be appropriate to weight the components of a vector to account for the fact that the motifs occur in different and possibly overlapping regions, the fact that different motifs have different and possibly overlapping support sets. Exploration of these possibilities is left for future work.

2.2 Partial-Taxonomy Implementation

The second implementation of CASTOR+ is the partial-taxonomy implementation. It works by exploring only selected classes and the associated motifs in the given taxonomy. Which classes to explore depends on the structure of the given taxonomy and the composition of the test sequence. In fact, the system now takes the given taxonomy as a decision tree, even if the given taxonomy may be a graph, by exploiting the hierarchical sub-structure and traversing only the directed edges that correspond to strict class inclusion in the given taxonomy.

How the partial-taxonomy implementation works is presented in an algorithmic form in **Figure 28** and further illustrated in **Figure 29**. Specifically, for every test sequence, CASTOR+ starts with the class that corresponds to the entire training set as the current class. If the current class is a positive set (see Chapter 2) and thus is associated with one or more motifs, the test sequence is matched to each of the associated motifs, and it is assigned to the current class if and only if it matches at least one of them. If the current class is a negative set and thus is not associated with any motif, the test sequence is assigned to the current class. Overall, if the test sequence is assigned to the current class, the current class is considered as a matching class; otherwise, it is considered as a non-matching class.

If the current class is considered as a matching class, CASTOR+ proceeds as follows. If the current class is an internal class, the system advances to each of its children that is a positive set, in turn, which becomes the current class. One or more of these children may turn out to be considered as matching classes, in which case the system continues traversing fully or partially their respective inference structures (see Chapter 2), which may overlap. If the system has visited all of the children of a matching class that are positive sets and none of them is a matching class, the system advances to the child that is a negative set, which becomes the current class. On the other hand, if the current class is considered as a non-matching class, the system returns to the parent. In the special case where the current class corresponds to the training set, the system

declares that the test sequence is not related to the entire training set. In this way, the system infers the potential functional subsets to which the test sequence belongs and the potential functional regions that the test sequences possess by considering only selected motifs in the given taxonomy.

As discussed previously, for each test sequence X, it is desirable to specify *near(X)*. In this implementation, since the matching classes form several paths, where a class along such a path is a subset of the previous class along the path, it is reasonable to consider the terminal classes at the end of these paths. Therefore, CASTOR+ specifically constructs *near(X)* as follows. It computes a score W to each training sequence such that for every terminal class to which both the training sequence and the test sequence belong, it adds W_P to W if the terminal class is a positive set and W_n to W if it is a negative set. In this case, W_P is set to 500 and W_n is set to 1. The rationale behind this choice is that matching a motif simultaneously serves as much stronger evidence for sharing functional similarity than mismatching a motif simultaneously and thus the former is used as the primary selection criterion while the latter is used as a secondary one. Any pair of scores such that the score for a positive set exceeds the total scores of all the possible negative sets would also do. The system identifies *near(X)* as the subset of training sequences with the largest W. This is also illustrated in **Figure 29**.

3 Results

To evaluate the performance of CASTOR+, we use the GPCR superfamily as a test case. We take the set of 815 non-chemoreceptor GPCR sequences as the *database of GPCR sequences*. Chemoreceptors will be investigated separately in the future. We take the reference class set (see Chapter 2) as the *reference taxonomy*, which has 219 terminal classes. Roughly 80% of these terminal classes correspond to *types*, where a type is a set of different copies of one gene from different species, another 12% of them correspond to unions of multiple types. 63% of the terminal classes contain at least three members.

As in a three-fold cross-validation, we split the database of GPCR sequences into a training set and a test set for three distinct rounds. We then use CASTOR to construct a binary tree for the training set and convert it into a graph via structure reorganization. This is especially appropriate for the full-taxonomy implementation, since each component of a vector should correspond to a distinct motif, and a graph generally contains more detailed and complex information about the underlying functional regions and subsets as well. Specifically, we set the parameters to typical

values as follows. $N_B = 2$, $N_D = \infty$, $u_1 = 0.7$, $t_1 = 0.7$, $t_2 = 0.9$, $e_3 = 20$, $l = 8$, $w = 8$, $k = 4$, and $n = 3$. We also construct negative sets in addition as follows. For each positive set P in the graph that contains other positive sets, we construct a negative set as the difference between P and the union of its children. The negative set is designated only as the child of P, even if it may be contained in some other positive set that is not on the inference path of P or even contained in some other negative set. A negative set could turn out to be empty.

For a complete performance evaluation, it is necessary to examine the full classification results produced by either CASTOR+ or CASTOR, which consists of a set of classes at different levels of the given taxonomy, with respect to the reference taxonomy. However, a CASTOR-generated taxonomy in this chapter is a graph while the reference taxonomy is a tree, and it is generally tricky and potentially unfair to compare a non-hierarchical taxonomy with a hierarchical one. Nevertheless, we attempt to get some ideas about how the classification based on a CASTOR-generated graph is with respect to the reference taxonomy by considering the classification at the lowest level. Specifically, for each test sequence X, we consider *near(X)*, as discussed in Methods, against the terminal classes in the reference taxonomy as follows. We assign X to the terminal class in the reference taxonomy that has the most elements in *near(X)* and see whether X indeed belongs to the selected terminal class.

Overall, for each terminal class in the reference taxonomy that is represented both in the training set and in the test set, we compute two measures: the sensitivity (number of test sequences that belong to the class and are correctly assigned to it, normalized by the number of test sequences that belong to the class) and the specificity (number of test sequences that do not belong to the class and are not assigned to it normalized by the number of test sequences that do not belong to the class).

We first study the performance of the two implementations of CASTOR+ individually and with respect to each other. We then try to estimate the difference between the taxonomies obtained by running CASTOR on the entire database of GPCR sequences and those obtained by running CASTOR on a training set followed by running (the full-taxonomy implementation of) CASTOR+ on the corresponding test set. Finally, we consider how the performance of (the partial-taxonomy implementation of) CASTOR+ changes with certain adjustments.

3.1 Full-Taxonomy Implementation

Take the results in the first round of the three-fold cross validation as an example. 142 (out of 219) terminal classes in the reference taxonomy are represented in both the training set and the test set. Roughly, 77% of these 142 terminal classes correspond to types, another 17% of them correspond to unions of multiple types, and 84% of them contain at least three members. Therefore, a large percentage of these terminal classes correspond to types, each consisting of several copies of one gene from different species, which are expected to share much higher sequence similarity with one another than with copies of a different gene. As can be seen in Row 1 of **Table 6**, the full-taxonomy implementation makes very few mistakes in annotating a test sequence X using a specific *near(X)* with respect to the reference taxonomy. As can be seen in Row 2 of **Table 6**, if *near(X)* simply consists of the highest scoring training sequence as computed by BLAST, the result of annotating X using *near(X)* is as good as the result produced by the full-taxonomy implementation.

Even if the motifs are discovered from about 2/3 of the database of GPCR sequences, the fact that this implementation errs rarely suggests that the composition of the test sequences are still relatively close to that of the training sequences and CASTOR works in a relatively effective and robust way. On the other hand, it appears that one approach to classify a test sequence with respect to the output of CASTOR is to find the highest scoring training sequence as computed by BLAST and assign the test sequence to every class in the given taxonomy to which the training sequence belongs. Despite the convenience of using an existing method, this approach is of an approximate nature, since it does not ensure that the test sequence matches the motifs associated with every class to which it belongs in an appropriate way. More importantly, it might not perform well when a training sequence that shares extremely high sequence similarity with the test sequence is absent and sequence similarity does not exactly coincide with functional similarity. For example, a region conserved in two sequences does not necessarily correspond to a functional region. Therefore, the test sequence may be approximately equally close to several training sequences in terms of sequence similarity, while it is closer to some of them than the others in terms of functional similarity.

To verify this, we remove the sequences closest to some in the test set, in terms of pairwise sequence similarities, from the training set until a BLAST-based approach would yield incorrect results. We then determine if CASTOR+ would produce correct results under identical

circumstances. This approach is designed to make a BLAST-based approach fail and aimed at verifying that in some scenarios, CASTOR+ would produce better results than BLAST-based approaches. Such scenarios may exist, since CASTOR+ attempts to identify functional regions and uses them to perform classification, while BLAST considers full-length sequence similarity and may occasionally be misled by excessive sequence similarity. In practice, both pairwise sequence comparison and motif analysis should be performed, as they can produce largely complementary results.

Take the first round of the three-fold cross validation as an example. We consider the classes at the next-to-bottom level of the reference taxonomy instead of the ones at the bottom level as the terminal classes, since the former are larger with members sharing lower sequence similarity in general. Furthermore, we systematically select a subset of the old test set as the new test set and a corresponding subset of the old training set as the new training set such that the following holds. For each test sequence in the new test set, some training sequences *R1* that do not belong to the same terminal class in the reference taxonomy as the test sequence have higher scores as computed by BLAST than some other training sequences *R2* that do belong to the same terminal class. Furthermore, none of those training sequences that belong to the same terminal class as the test sequence and have higher scores than any *R1* is kept in the new training set, while at least one *R1* and at least one *R2* are kept in the new training set.

Specifically, we consider the test sequences in the old test set one by one with respect to the old training set. For a test sequence *X* that belongs to a terminal class *T*, suppose that there exists a training sequence *R1* that does not belong to *T*, has a lower score as computed by BLAST only to some training sequences that belong to *T*, has a higher score than another *R2* that belongs to *T*, and has not been marked irremovable. Suppose also that none of the training sequences that belong to *T* and have higher scores than any training sequences that do not belong to *T* has been marked irremovable. We then keep *X*, remove all the training sequences that belong to *T* and have higher scores than any training sequences that do not belong to *T*, and mark irremovable *R1* and every *R2* that has not been marked irremovable. Otherwise, we discard *X*. The result is that for every test sequence in the new test set, the training sequences that correspond to the same gene as the test sequence are usually absent in the new training set. Furthermore, the highest scoring training sequence as computed by BLAST in the new training set, where the score is still relatively high, would lead to incorrect classification at the lowest level.

We would like to see whether CASTOR+ would correctly classify these test sequences at the lowest level. We thus run CASTOR on the new training set of 129 training sequences, and run the full-taxonomy implementation of CASTOR+ on the new test set of 35 well-characterized (non-orphan, non-probable) test sequences. As can be seen in **Table 7**, the nearest neighbor computed by the full-taxonomy implementation leads to correct classification in about 20% of the cases. Interestingly, for MGR1_CAEEL, CASTOR+ considers MGR1_RAT as the nearest neighbor, which is very reasonable, while BLAST considers MGR3_HUMAN as the nearest neighbor. For OPSD_APIME, for example, CASTOR+ considers OPS4_DROVI as the nearest neighbor, which agrees with the reference taxonomy, while BLAST considers OPSD_SEPOF as the nearest neighbor, which would have been reasonable judging by their names.

3.2 Partial-Taxonomy Implementation

As can be seen in Row 3 of **Table 6**, the partial-taxonomy implementation makes slightly more mistakes than the full-taxonomy implementation in annotating a test sequence X using a specific *near(X)* with respect to the reference taxonomy. Take the results in the first round as an example. While the full-taxonomy implementation fails to achieve a sensitivity of one in about 8% of the cases, the partial-taxonomy implementation does so in about 25% of the cases. On the other hand, for every test sequence, the full-taxonomy implementation considers all the classes and thus all the motifs in the given taxonomy, while the partial-taxonomy implementation considers only selected classes and their associated motifs. In the latter case, as can be seen in **Figure 30**, the number of motifs to match varies from one test sequence to another, and the average is about 80% of the total number of motifs in the given taxonomy. Therefore, while the full-taxonomy implementation performs better in terms of the classification at the lowest level, the partial-taxonomy implementation certainly saves some time.

These results agree with our expectation. The full-taxonomy implementation compares protein sequences in a global and comprehensive manner by considering all the motifs in the given taxonomy. On the one hand, it takes into account not only the similarity in functional regions but also that in non-functional regions. On the other hand, it takes into consideration not only similarity with members that belong to the same class but also dissimilarity with members that do not belong to the same class. Therefore, it is intuitive that the full-taxonomy implementation performs better than the partial-taxonomy implementation by factoring in more information. Some other secondary factors may also contribute to the poor performance of the partial-

taxonomy implementation. For example, the thresholding of *E*-values, which implicitly converts each *E*-value into a binary value in determining whether a test sequence matches a motif, may result in a non-trivial loss of information. In addition, the way that *near(X)* is currently identified may also be further improved. Exploration of these possibilities is left for future work.

Nevertheless, we examine in some detail the cases where the partial-taxonomy implementation performs worse than the full-taxonomy implementation at least in terms of the classification at the lowest level. For about one-third of the cases where the partial-taxonomy implementation fails to achieve a sensitivity of one, it appears that the traversed paths in the given taxonomy for a test sequence *X* that belongs to a terminal class *T* in the reference taxonomy do not lead to a *near(X)* that is sufficiently fine-grain. In other words, *near(X)* may contain all the training sequences that belong to *T* plus some others that belong to some other terminal classes, and *X* may be eventually assigned to one of those other terminal classes. For the other two-thirds of the classes, it is possible that the test sequence does not fit sufficiently well to the structure of the given taxonomy, especially its hierarchical sub-structure.

3.3 Reconstructed Taxonomy vs. Extended Taxonomy

As discussed in Introduction, if we have an old group of protein sequences, a taxonomy on the old group constructed by CASTOR, and a new group of protein sequences, we could either get a brand new taxonomy by applying CASTOR to the union of the two groups or extend the given taxonomy by applying CASTOR+ to the new group. In either case, we obtain a taxonomy on the union of the two groups of protein sequences. In the former case, we call the resulting taxonomy a *reconstructed taxonomy*, and in the latter case, we call the resulting taxonomy an *extended taxonomy*.

In this section, the old group of protein sequences corresponds to a training set, the new group of protein sequences corresponds to a test set, and the union of the two groups corresponds to the database of GPCR sequences. We consider the resulting reconstructed taxonomy and extended taxonomy. The reconstructed taxonomy has 388 classes associated with 410 motifs. Take the results in the first round of the three-fold cross validation as an example. The taxonomy obtained by applying CASTOR to the training set and thus the extended taxonomy obtained by subsequently applying the full-taxonomy implementation of CASTOR+ to the test set has 224 classes associated with 249 motifs, which are a bit less than 2/3 of the classes and motifs in the reconstructed taxonomy. Some results of comparing the two taxonomies are shown in **Figure 31**.

A comparison between the distribution of class sizes in the reconstructed taxonomy and that in the extended taxonomy indicates that there are many more small classes in the reconstructed taxonomy. A comprehensive pairwise comparison between the classes in the reconstructed taxonomy and those in the extended taxonomy such that two classes are considered as a match if and only if their overlap is at least 70% of both indicates that the two taxonomies have about 69 classes in common, which is not a large proportion of either taxonomy. It also indicates that the classes at the lowest levels of the two taxonomies tend to disagree. This is as expected, since those motifs that are present in a majority of the sequences in the union of the training set and the test set still tend to be prevalent in the training set, which is still relatively large, and be discovered easily.

To gain more understanding about how the reconstructed taxonomy and the extended one differ, we compare the corresponding results of classification at the lowest level. Notice that the results based on the reconstructed taxonomy are obtained by applying CASTOR alone, where the training sequences and the test sequences are classified simultaneously. Therefore, we would actually be evaluating how well the test sequences are clustered with the training sequences rather than how they are classified against them. As can be seen in Row 4 and Row 5 of **Table 6**, if for each test sequence X, *near(X)* is identified in the same way as in the full-taxonomy implementation of CASTOR+ with the reconstructed taxonomy as the given taxonomy, the annotation of X using *near(X)* is rarely wrong with respect to the reference taxonomy. If *near(X)* is identified in the same way as in the partial-taxonomy implementation, the annotation is wrong more often. Overall, the results based on the reconstructed taxonomy are almost identical to those based on the extended taxonomy, even if the reconstructed taxonomy has many more classes and motifs. This could mean that while the reconstructed taxonomy offers more detailed and complex information about the underlying functional subsets and regions, it has the same capabilities for classifying the test sequences at the lowest level as the extended taxonomy.

It is possible to consider a hybrid of the reconstructed-taxonomy approach and the extended-taxonomy approach, where a fine-grain, selective reconstruction is carried out in those cases where it would otherwise result in a critical degradation of classification quality. Specifically, we could first apply CASTOR+ to the new group of protein sequences and extend the old taxonomy. However, if the size of any class in the extended taxonomy exceeds a predefined threshold, we could then apply CASTOR on the class, which is a subset of the union of the two groups of

protein sequences. The point is to select an appropriate threshold to strike the best balance between output quality and execution speed. We will explore this in the future.

3.4 Refinements of Partial-taxonomy Implementation

We explore some adjustments to the partial-taxonomy implementation, for which there appears to be more room for improvement.

We first consider how to traverse the given taxonomy in an alternative way that is reminiscent of the fuzzy clustering (see Chapter1). The idea is to introduce some fuzziness to the traversal of the given taxonomy by exploring both a matching branch and the corresponding non-matching branch in the graph when the E-value falls in the borderline range. Recall that the system considers the current class as a matching class as long as the test sequence matches one of the associated motifs. Originally, when secondary-structure information is available, if the system decides that the test sequence matches a motif based on e_3, it considers the current class as a matching class and advances to each of its children. With the introduction of fuzziness, if the system decides that the test sequence matches a motif while the corresponding E-value is greater than e_0, it considers the current class as a non-matching class but still advances to each of its children. Since the system considers the current class as a non-matching class, it is likely to eventually advance to the negative set corresponding to the parent of the current class. The approach turns out to perform less well than the original approach in terms of classification at the lowest level.

We then consider how to identify *near(X)* in alternative ways. One idea is to consider all the training sequences that are classified differently from the test sequence only at the lowest level, as it appears very likely to confuse two sequences that belong to the same super-families and families but to different sub-families. Originally, the system computes a score for each training sequence based on the terminal classes in the given taxonomy to which both the training sequence and the test sequence belong. With the introduction of some fuzziness to the bottom level of the given taxonomy, the system computes a score W for each training sequence based on the classes at the next-to-bottom level of the given taxonomy to which both the training sequence and the test sequence belong. For every such class, which is the parent of some terminal classes, if the training sequence and the test sequence belong to one common child, which could be a positive set or a negative set, the system adds W_i to W; otherwise, it adds W_d to W. In this case, W_i is set to 100 and W_d is set to 50. In this case, we intend to distinguish when the training sequence and the

test sequence belong to the same class and when they do not more than when the terminal class is a positive set and when it is not, and we set comparable values to W_i and W_d to ensure the fuzziness. This approach turns out to perform almost as well as the original approach in terms of classification at the lowest level.

Another idea is to exploit the combined statistical significance of the motifs. The system computes a score W for each training sequence based on how well the test sequence matches the motifs. For every terminal class to which both the training sequence and the test sequence belong, the system adds a number that is proportional to the combined statistical significance corresponding to matching the motifs associated with the paths that lead to the terminal class. Specifically, for each path that leads to the terminal class, the system computes the product of the appropriate P-values, which could be derived from the E-values. The system then computes the sum of the products over all the appropriate paths as the combined statistically significance. This approach turns out to perform almost as well as the original approach in terms of classification at the lowest level.

4 Related Work

Some other researchers have designed and implemented protein classification systems and applied them to the GPCR superfamily. These classification systems take a test sequence and a taxonomy, which is generally a manually constructed one, and assigns the test sequence to one of the classes in the given taxonomy.

Karchin et al. construct a one-to-rest support vector machine (SVM) for every class in a given one-level taxonomy and assign a test sequence to the class with the largest discriminant as reported by the SVMs. The fact that the input is a one-level taxonomy means that their method does not provide multi-level or any sort of complex membership information for each test sequence, while CASTOR offers classification results at different levels. In addition, their method takes a taxonomy and computes the features for each class in the given taxonomy, while CASTOR+ takes a taxonomy with the features of each class already computed. Furthermore, their method considers the features of a class as well as those of the other classes in deciding whether to assign a test sequence to the class. On the other hand, for each test sequence, the full-taxonomy implementation of CASTOR+ also considers all the motifs but in a global way rather than a one-to-rest way, and the partial-taxonomy implementation of CASTOR+ considers only those motifs that characterize a class in deciding whether to assign a test sequence to the class.

In their evaluation procedure, they take the set of GPCR sequences available at GPCRDB as the database of GPCR sequences, and they take the GPCR superfamily tree available at GPCRDB as the reference taxonomy (while we use a fusion of this tree and another tree available at PRINTS as the reference taxonomy), which is also their given taxonomy. This reference taxonomy (as well as the one we use) is hierarchical and classifies the superfamily into five major families, but they consider only the first-level partition and the second-level partition for Family A and Family C, which do not include any types. In fact, they perform a 2-fold cross-validation and consider only those classes in the reference taxonomy that have at least four members. They focus on Family A for its prevalence and Family C for its significance, and they focus on the first-level partition and the second-partition rather than the leaves of the tree due to the generally small sizes of the types and the occasional inconsistency in combining/mixing the types into individual classes. They compute two statistics, coverage and MEP (average errors per sequence at the minimum error point). The first one is the percent of true positives recognized before the first false positive and is supposed to measure the sensitivity of a classifier. The second one is the score threshold where a classifier makes the fewest errors of both kinds (false positives plus false negatives) and is supposed to provide a fair comparison of both the sensitivity and selectivity of different methods.

Scordis et al. use a simple algorithm to determine whether to assign a test sequence to a class in the given taxonomy [68]. This method takes a taxonomy with the features of each class already computed. In this case, the features are motifs represented by ungapped multiple alignments, each associated with a distinct location of occurrence. Their method assigns a test sequence to a class in the given taxonomy as long as it matches some of the motifs associated with a class in the right positional order. Therefore, they exploit the contextual information while allowing "partial" membership to a protein family.

As part of their evaluation procedure, they take selected queries and the GPCR family tree available at PRINTS as the reference taxonomy, which is also their given taxonomy. This reference taxonomy is again hierarchical. Their method does not combine the matching information regarding hierarchical protein families in any way in that a match to the superfamily of GPCRRHODOPSN and a match to the family of OPSIN are considered as two independent matches. It is simply that when a situation like this happens, they take it as indicating possible or confirming putative evolutionary relationships. They only discuss specific examples without reporting comprehensive evaluation results.

5 Conclusion

In this chapter, using the GPCR superfamily as a test case, we have shown the following. The full-taxonomy implementation of CASTOR+ performs almost as well as BLAST in terms of classification at the lowest level. Furthermore, it often still succeeds even as BLAST fails when no training sequence that shares extremely high sequence similarity with the test sequence is present and sequence similarity does not exactly coincide with functional similarity in general. This suggests that CASTOR+ may be good at identifying functional regions, which are conserved in the appropriate functional subsets and potentially more effective for classification purposes than mere pairwise sequence similarity. In addition, the partial-taxonomy implementation of CASTOR+ performs less well than the full-taxonomy implementation in terms of classification at the lowest level, but it also works more efficiently by considering only selected classes and their associated motifs in the given taxonomy. This demonstrates the tradeoff between output quality and execution efficiency, as expected. Finally, the reconstructed taxonomy is much larger than the extended taxonomy, but the classification results at the lowest level based on the reconstructed taxonomy are of approximately the same quality as those based on the extended taxonomy. This suggests that while the reconstructed taxonomy may offer more detailed and complex information about the underlying functional subsets and regions, it has the same capabilities for classification at the lowest level as the extended taxonomy. Therefore, it could be a good idea to reuse and extend the old taxonomy by applying CASTOR+ to the test set when classification at the lowest level is especially of interest but not necessarily when the size of the test set is sufficiently large and the entire classification at all levels are desired.

We have assumed that no functional information is available about the proteins under consideration in this chapter. However, CASTOR+ is a supervised system and does not have to work with the output of CASTOR. When some functional information, a biologically meaningful taxonomy in particular, is available about the proteins under consideration, the signature motifs represented by profile HMMs could be generated for each class in the taxonomy via appropriate motif-finding algorithms. The resulting taxonomy, where each class is associated with zero or more motifs, could then be used as input to CASTOR+. Alternatively, we could apply CASTOR to the union of the training set and the test set to generate another taxonomy and map the classes in the resulting taxonomy to the biologically meaningful one in an appropriate way. The CASTOR-generated taxonomy, where each class is associated with zero or more motifs, with the

appropriate mapping to the biologically meaningful one could then be used as input to CASTOR+.

```
Traverse(curr_class, test, seq) {

        If (curr_class is a positive set) {

                Match every motif associated with curr_class against test_seq.

                If (at least one associated motif matches test_seq) {
                        Assign test_seq to curr_class.
                        Declare curr_class as a matching class.
                }
                else {
                        Declare curr_class as a non-matching class.
                }

                If (curr_class is considered as a matching class) {

                        If (curr_class is an internal class) {

                                For each child that is a positive set, pos_child {
                                        Traverse(pos_child).
                                }

                                If (no child that is a positive set is considered as a matching class) {
                                        Traverse(neg_child).
                                }

                        }

                }

        }
        else {
                Assign test_seq to curr_class.
                Declare curr_class as a matching class.
        }

}

Traverse(entire_training_set, test_sequence).
```

Figure 28: Partial-Taxonomy Implementation. This is an algorithmic description of the partial-taxonomy implementation of CASTOR+.

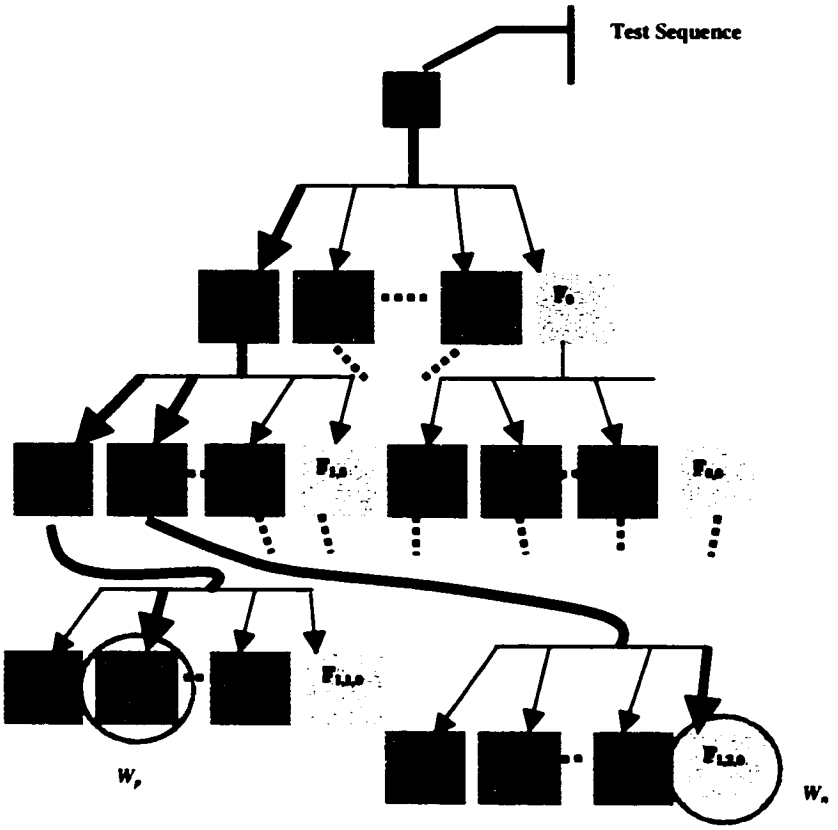


Figure 29: CASTOR-Generated Taxonomy and Partial-Taxonomy Implementation. This shows a typical CASTOR-generated taxonomy. The black node represents the class that corresponds to the entire database, a pink node represents a class that is a positive set, and a blue node represents a class that is a negative set. An arrow indicates that the source class contains the destination class. For a test sequence X, the partial-taxonomy implementation of CASTOR+ starts with the class corresponding to the entire database and decides which classes in the taxonomy to visit depending on how the test sequence matches the motifs associated with the classes that have been visited. In this illustration, the system traverses those paths indicated by arrows in bold and eventually reach the two class indicated by nodes in circles. It assigns a weight of W_p to every training sequence in $F_{1.1.2}$ and a weight of W_n to every training sequence in $F_{1.2.0}$, and it identifies *near(X)* accordingly.

Table 6: Summary of Classification Performance. Each entry shows the number of classes in the reference taxonomy with the corresponding sensitivity/specificity values (column header) for the corresponding method (row header). If the entry is a triplet, each component corresponds to one round in the three-fold cross-validation. If it is a single number, it corresponds to the first round of the cross-validation.

	Sensitivity 0.00	Sensitivity Between 0.00 & 1.00	Sensitivity 1.00	Specificity 1.00	Specificity < 1.00
Full-taxonomy for Extended Taxonomy	7, 7, 5	5, 4, 9	130, 141, 133	127, 141, 130	15, 11, 17
BLAST	6, 3, 4	3, 3, 3	133, 146, 140	132, 146, 140	10, 6, 7
Partial-taxonomy for Extended Taxonomy	29, 25, 20	6, 10, 10	107, 117, 117	111, 123, 126	31, 29, 21
Full-taxonomy for Reconstructed Taxonomy	8	3	131	130	12
Partial-taxonomy for Reconstructed Taxonomy	18	12	112	112	30

Table 7: Instances of Sole CASTOR Success. This table shows the results for those test sequences in the new test set where the full-taxonomy implementation of CASTOR+ succeeds in terms of fine-grain classification. Each row corresponds to a test sequence, where the number enclosed in parentheses indicates the number of training sequences that belong to the same terminal node as the test sequence. The second column shows the fine-grain classification result produced by the full-taxonomy implementation of CASTOR+, the third column shows the fine-grain classification result produced by BLAST, and the fourth column shows the members of the terminal node to which the test sequence belongs.

Sequence ID	CASTOR+	BLAST	Reference Taxonomy
5H1A_RAT (5)	5H1F_MOUSE	5HT_HELVI	5H1A_*, 5H1B_*, 5H1D_*, 5H1E_*, 5H1F_*
MGR1_CAEEL (3)	MGR1_RAT	MGR3_HUMAN	MGR1_*, MGR5_*
OPS1_CALVI (6)	OPSB_APIME	OPSD_SEPOF	OPS1_CALVI, OPS1_DROME, OPS1_DROPS, OPS1_HEMSA, OPS1_LIMPO, OPS1_SCHGR, OPS2_DROME, OPS2_DROPS, OPS2_HEMSA, OPS2_LIMPO, OPS2_SCHGR, OPS3_DROME, OPS3_DROPS, OPS4_DROME, OPS4_DROPS, OPS4_DROVI, OPS5_DROME, OPS6_DROME, OPSB_APIME, OPSD_APIME, OPSD_CAMAB, OPSD_CATBO, OPSD_PROCL, OPSD_SPHSP, OPSV_APIME
OPS1_HEMSA (6)	OPS2_SCHGR	OPSD_TOPDA	
OPS1_SCHGR (6)	OPS3_DROPS	OPSD_OCTDO	
OPS6_DROME (6)	OPS4_DROVI	OPSD_OCTDO	
OPSD_APIME (6)	OPS4_DROVI	OPSD_SEPOF	
PE21_RAT (7)	PE24_HUMAN	PF2R_RAT	PE21_*, PE23_*, PE24_*

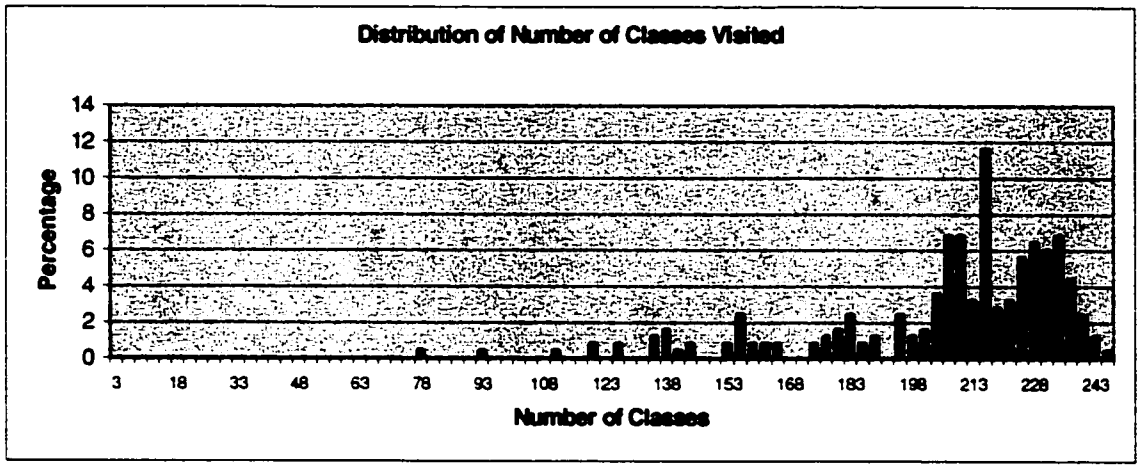


Figure 30: Distribution of Number of Visited Classes. The series of bars shows the distribution of numbers of visited classes in the first round of the three-fold cross validation with the partial-taxonomy implementation of CASTOR+. The total number of classes in the given taxonomy that is represented both in the training set and in the test set is 249.

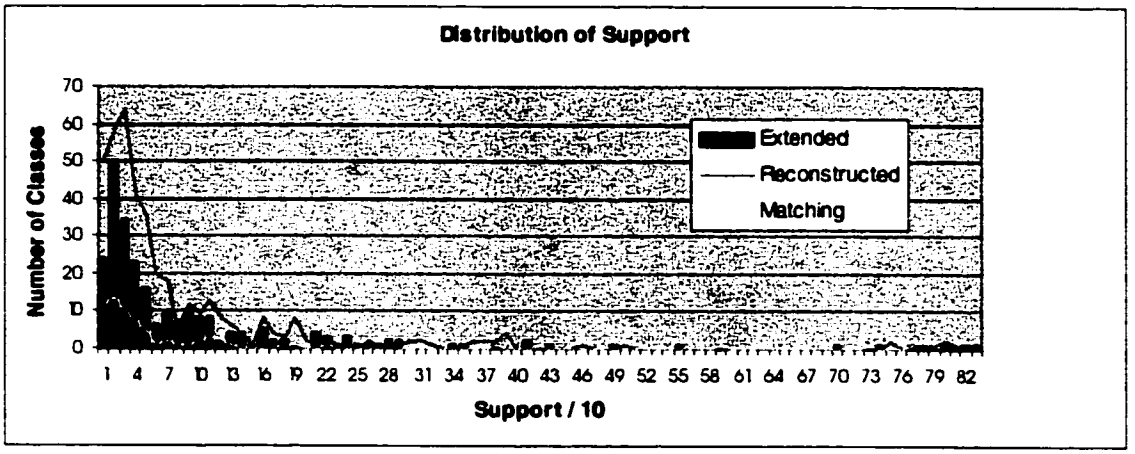


Figure 31: Distribution of Support. The Extended curve shows the distribution of supports over the classes in the extended taxonomy in the first round of the three-fold cross validation. The Reconstructed curve shows the same for the reconstructed taxonomy. The Matching curve shows the same thing over the matched classes.

Bibliography

[1] Antoniak, C. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Ann. Statist.* 2, 1152 – 1274.

[2] Araneda, R. C., Kini, A. D., and Firestein, S. 2000. The molecular receptive range of an odorant receptor. *Nat. Neurosci.* 3, 1248 – 1255.

[3] Attwood, T. K., Beck, M. E., Bleasy, A. J., and Parry-Smith, D. J. 1994. PRINTS – a database of protein motif fingerprints. *Nucleic Acids Res.* 22(17), 3590 – 6.

[4] Bailey, T. L. and Gribskov, M. 1998. Combining evidence using p-values: Application to sequence homology searches. *Bioinformatics* 14, 48 – 54.

[5] Bairoch, A. 1993. The PROSITE dictionary of sites and motifs in proteins, its current status. *Nucleic Acids Res.* 21(13), 3097 – 3103.

[6] Bairoch A., and Apweiler R. 1997. The SWISS-PROT protein sequence database: its relevance to human molecular medical research. J. *Mol. Med.* 75, 312 – 316.

[7] Ballesteros, J. A., and Weinstein, H. 1992. Analysis and refinement of criteria for predicting the structure and relative orientations of transmembranal helical domains. *Biophys. J.* 62, 107 – 109.

[8] Ballesteros, J. A., Shi, L., and Javitch, J. A. 2001. Structural mimicry in G protein-coupled receptors: implications of the high-resolution structure of rhodopsin for structure-function analysis of rhodopsin-like receptors. *Mol. Pharmacol.* 60, 1 – 19.

[9] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A., and Wheeler, D. L. 2000. *Nucleic Acid Res.* 28(1), 15 – 18.

[10] Beukers, M. W., Kristiansen, I., Ijzerman, A. P., and Edvardsen, I. 1999. TinyGRAP database: a bioinformatics tool to mine G protein-coupled receptor mutant data. *Trends Pharmaco.l Sci.* 20(12), 475 – 77.

[11] Boekhoff, I. And Breer, H. 1992. Termination of second messenger signaling in olfaction. *Proc. Natl. Acad. Sci. USA* 89, 471 – 474.

[12] Boekhoff, I., Schleicher, S., Strotmann, J., and Breer, H. 1992. Odor-induced phosphorylation of olfactory cilia proteins. *Proc. Natl. Acad. Sci. USA* 89, 11983 – 11987.

[13] Boekhoff, I., Touhara, K., Danner, S., Inglese, J., Lohse, M. J., Breer, H., and Lefkowitz, R. J. 1997. Phosducin, potential role in modulation of olfactory signaling. J. *Biol. Chem.* 272, 4606 – 4612.

[14] Bork, P., Sander, C., Valencia, A. And Bukau, B. 1992. A module of the DnaJ heat shock proteins found in malaria parasites. *Trends. Biochem. Sci.* 17, 129.

[15] Buck, L. And Axel, R. A 1991. Novel multigene family may encode odorant receptors: a molecular basis for odor recognition. *Cell* 65, 175 – 187.

[16] Brown, M., Hughey, R., Krogh, A., Mian, I. S., Sjolander, K., and Haussler, D. 1993. Using Dirichlet Mixture Priors to Derive Hidden Markov Models for Protein Families. *Proc. Int. Conf. Intell. Syst. Mol. Biol. (ISMB-93)*, 47 – 55.

[17] Brunet, L. J., Gold, G. H., and Ngai, J. 1996. General anosmia caused by a targeted disruption of the mouse olfactory cyclic nucleotide-gated cation channel. *Neuron* 17(4), 681 – 693.

[18] Bywater, R., Vriend, G., Oliveira, L., and van Aalten, D. 1995. Using Sequence information and Model Building to Explore Subtype Specificity in GPCRs. *Protein Folds: A Distance-Based Approach*. Eds. Bohr H. Et al. CRC Press. 174 – 185.

[19] Califano, Andrea. 1999. SPLASH: SPLASH: structural motif localization analysis by sequential histograms. *Bioinformatics* 16, 341 – 357.

[20] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. 1992. *Introduction to Algorithms.* MIT Press, Cambridge, MA.

[21] Dawson, T. M., Arriza, J. L., Jaworsky, D. E., Borisy, F. F., Attramadal, H., Lefkowitz, R. J., and Ronnett, G. V. 1993. Beta-adrenergic receptor kinase-2 and beta-arrestin-2 as mediators of odorant-induced desensitization. *Science* 259, 825 – 829.

[22] Dembo, A., and Karlin, S 1991. Strong Limit Theorems of Empirical Functionals for Large Exceedances of Partial Sums of I.I.D. Variables, *Ann. Probab.* 19(4), 1737 – 1955.

[23] Dodge, C., Schneider, R., and Sander, C. 1998. The HSSP database of protein structure-sequence alignments and family profiles. *Nucleic Acids Res.* 26(1), 313 – 315.

[24] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. 1998. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge, UK

[25] Firestein, S. 2001. How the olfactory system makes sense of scents. *Nature* 413, 211 – 218.

[26] Freitag, J., Krieger, J., Strotmann, J., and Breer, H. 1995. Two classes of olfactory receptors in Xenopus laevis. *Neuron* 15, 1383 – 1392.

[27] Freitag, J., Ludwig, G., Andreini, I., Rossler, P., and Breer, H. 1998. Olfactory receptors in aquatic and terrestrial vertebrates. J. *Comp. Physiol. [A]* 183, 635 – 650.

[28] Gasteiger, E., Jung, E., and Bairoch, A. 2001. SWISS-PROT: connecting biomolecular knowledge via a protein database. *Curr. Issues Mol. Biol.* 3, 47 – 55.

[29] Gether, U. 2000. Uncovering molecular mechanisms involved in activation of G protein-coupled receptors. *Endocr. Rev.* 21, 90 – 113.

[30] Glusman, G., Bahar, A., Sharon, D., Pilpel, Y., White, J., and Lancet, D. 2000. The olfactory receptor gene superfamily: data mining, classification, and nomenclature. *Mamm. Genome* 11, 1016 – 1023.

[31] Glusman, G., Yanai, I., Rubin, I., and Lancet, D. 2001. The complete human olfactory subgenome. *Genome Res.* 11, 685-702.

[32] Gracy, J., and Argos, P. 1998. Automated protein sequence database classification. I. Integration of compositional similarity search, local similarity search, and multiple sequence alignment. *Bioinformatics* 14(2), 164 – 173.

[33] Gracy, J., and Argos, P. 1998. Automated protein sequence database classification. II. Delineation of domain boundaries from sequence similarities. *Bioinformatics* 14(2), 174 – 187.

[34] Grundy, W., N., Bailey, T., L., Elkan, C., P., and Baker, M. 1997. Meta-MEME: Motif-based Hidden Markov Models of Biological Sequences. *Computer Applications in the Biosciences* 13(4), 397 – 406.

[35] Hart R., Royyuru, A. K., Stolovitzky, S., and Califano, A. 2000. Systematic and fully automated identification of protein sequence motifs. J. *Comput. Biol.* 7(3-4), 585 – 600.

[36] Hatt, H., Gisselmann, G., and Wetzel, C. H. 1999. Cloning, functional expression and characterization of a human olfactory receptor. *Cell Mol. Biol. (Noisy-le-grand)* 45, 285 – 291.

[37] Henikoff, S. And Henikoff, J. G. 1992. Amino acid substitution matrices from protein blocks. P. Natl. Acad. Sci. USA 89, 10915 – 10919.

[38] Henikoff, J. G., Greene, E. A., Pietrokovski, S., and Henikoff, S. 2000. Increased coverage of protein families with the blocks database servers. *Nucleic Acids Res.* 28, 228 – 230.

[39] Horn, F., Weare, J., Beukers, M. W., Horsch, S., Bairoch, A., Chen, W., Edvardsen, O., Campagne, F., Vriend, G. 1998. GPCRDB: an information system for G protein-coupled receptors. *Nucleic Acids Res.* 26(1), 277 – 281.

[40] Horn, F., van der Wenden, E. M. Oliveira, L., IJzerman, A. P., and Vriend, G. 2000. Receptors Coupling to G Proteins: Is There a Signal Behind the Sequence? *Proteins: Structure, Function, and Genetics* 41, 448 – 459.

[41] Kajiya, K., Inaki, K., Tanaka, M., Haga, T., Kataoka, H., and Touhara, K. 2001. Molecular Bases of Odor Discrimination: Reconstitution of Olfactory Receptors that Recognize Overlapping Sets of Odorants. J. *Neurosci.* 21, 6018 – 6025.

[42] Karchin, R., Karplus, K., and Haussler, D. 2002. Classifying G-Protein Coupled Receptors with Support Vector Machines. *Bioinformatics* 18(1), 147 – 159.

[43] Karlin, S., Dembo, A., and Kawabata, T. 1990. Statistical Composition of High-Scoring Segments from Molecular Sequences. *Ann. Statist.* 18(2), 571 – 581.

[44] Krautwurst, D., Yau, K. W., and Reed, R. R. 1998. Identification of ligands for olfactory receptors by functional expression of a receptor library. *Cell* 95, 917 – 926.

[45] Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. 1994. Hidden Markov models in computational biology: applications to protein modeling. J. *Comput. Biol.* 235, 1501 – 1531.

[46] Kyte, J. And Doolittle, R. F. 1982. A simple method for displaying the hydropathic character of a protein. J. *Mol. Biol.* 157, 105 – 132.

[47] Lichtarge, O., Bourne, H. R., and Cohen, F. E. 1996. An Evolutionary Trace Method Defines Binding Surfaces Common to Protein Families. J. *Mol. Biol.* 257, 342-358.

[48] Liu, A. H. And Califano, A. 2001. Functional classification of proteins by pattern discovery and top-down clustering of primary sequences. *IBM Systems Journal* 40 (2), 379-393.

[49] Liu, A. H. And Califano, A. 2002. CASTOR: Clustering Algorithm for Sequence Taxonomical Organization and Relationships. Submitted to *J. Comput. Biol.*

[50] Liu, A. H., Zhang, X., Califano, A., Stolovitzky, G., and Firestein, S. 2002. Motif-Based Construction of Functional Atlas for Mammalian Olfactory Receptors. In progress.

[51] Malnic, B., Hirono, J., Sato, T., and Buck, L. B. 1999. Combinatorial receptor codes for odors. *Cell* 96, 713 – 723.

[52] Mombaerts, P. 1999. Molecular biology of odorant receptors in vertebrates. *Annu. Rev. Neurosci.* 22, 487 – 509.

[53] Nagl, S. B., Freeman, J., and Smith, T. F. 1999. Evolutionary constraint networks in ligand-binding domains: an information-theoretic approach. *Pac. Symp. Biocomput.* 4, 90 – 101.

[54] Nevill-Manning, C. G., Wu, T. D., and Brutlag, D. L. 1998. Highly specific protein sequence motifs for genome analysis. *Pro. Natl. Acad. Sci. USA* 95, 5865 – 5871.

[55] Onorato, J. J., Palczewski, K., Regan, J. W., Caron, M. G., Lefkowitz, R. J. and Benovic, J. L. 1991. Role of acidic amino acids in peptide substrates of the beta-adrenergic receptor kinase and rhodopsin kinase. *Biochemistry* 30, 5118 – 5125.

[56] Palczewski, K., Kumasaka, T., Hori, T., Behnke, C. A., Motoshima, H., Fox, B. A., Le Trong, I., Teller, D. C., Okada, T., Stenkamp, R. E., Yamamoto, M., and Miyano, M. 2000. Crystal structure of rhodopsin: A G protein-coupled receptor. *Science* 289, 739 – 745.

[57] Pietrokovski, S. 1996. Searching databases of conserved sequence regions by aligning protein multiple-alignments. *Nucleic Acids Res.* 24, 3836 – 3845.

[58] Pilpel, Y. And Lancet, D. 1999. The variable and conserved interfaces of modeled olfactory receptor proteins. *Protein Sci.* 8, 969 – 977.

[59] Pongor, S., Skerl, V., Cserzo, M., Hatsagi, Z., Simon, G., and Bevilacqua, V. 1993. The SBASE domain library: a collection of annotated protein segments. *Protein Eng.* 6(4), 391 – 395.

[60] Pitcher, J. A., Freedman, N. J., and Lefkowitz, R. J. 1998. G protein-coupled receptor kinases. *Annu. Rev. Biochem.* 67, 653 – 692.

[61] Raming, K., Krieger, J., Strotmann, J., Boekhoff, I., Kubick, S., Baumstark, C., and Breer, H. 1993. Cloning and expression of odorant receptors. *Nature* 361, 353 – 356.

[62] Rigoutsos, I., Floratos, A., Ouzounis, C., Gao, Y., and Parida, L. 1999. Dictionary Building via Unsupervised Hierarchical Motif Discovery in the Sequence Space of Natural Proteins. *Proteins: Structure, Function, and Genetics* 37, 264 – 277.

[63] Rost, B. 1996. *PHD*: predicting one-dimensional protein structure by profile based neural networks. *Methods Enzymol* 266, 525 – 539.

[64] Saitou, N. And Nei, M. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol.* 4(4), 406-25.

[65] Schleicher, S., Boekhoff, I., Arriza, J., Lefkowitz, R. J., and Breer, H. 1993. A beta-adrenergic receptor kinase-like enzyme is involved in olfactory signal termination. *Proc. Natl. Acad. Sci. USA* 90, 1420 – 1424.

[66] Schultz, J., Copley, R. R., Doerks, T., Ponting, C. P., and Bork, P. 2000. SMART: a web-based tool for the study of genetically mobile domains. *Nucleic Acids Res.* 28(1), 231-234.

[67] Schwartz, R. M., and Dayhoff, M. O. 1978. Matrices for Detecting Distant Relationships. *Atlas of Protein Sequence and Structure*, ed. Dayhoff, M. O., 353 – 358.

[68] Scordis, P., Flower, D. R., and Attwood, T. K. 1999. FingerPRINTScan: intelligent searching of the PRINTS motif database. *Bioinformatics* 15(10), 799 – 806.

[69] Singer, M. S., Oliveira, L., Vriend, G. & Shepherd, G. M. 1995. Potential ligand-binding residues in rat olfactory receptors identified by correlated mutation analysis. *Receptors Channels* 3, 89 – 95.

[70] Skoufos, E., Healy, M. D., Singer, M. S., Nadkarni, P. M., Miller, P. L., and Shepherd, G. M. 1999. Olfactory Receptor Database: a database of the largest eukaryotic gene family. *Nucleic Acids Res.* 27, 343 – 345.

[71] Sneath, P. H. 1973. *Numerical Taxonomy*: The Principles and Practice of Numerical Classification. W. H. Freeman & Co., San Francisco, CA.

[72] Sonnhammer, E. L. L., and Kahn, D. 1994. Modular arrangement of proteins as inferred from analysis of homology. *Protein Sci.* 3, 482 – 492.

[73] Sonnhammer, E. L. L., von Heijne, G., Krogh, A. 1998. A hidden Markov model for predicting transmembrane helices in protein sequences. *Proc. Int. Conf. Intell. Syst. Mol. Biol. (ISMB98)*, 175-182.

[74] States, D. J., Harris, N. L., Hunter, L. 1993. Computationally Efficient Cluster Representation in Molecular Sequence Megaclassification. *Proc. Int. Conf. Intell. Syst. Mol. Biol. (ISMB-93)*, 387 – 394.

[75] Stolovitzky G. And Califano A., 1998. Statistical Significance of Motifs in Biosequences. *IBM RC.*

[76] Strotmann, J., Hoppe, R., Conzelmann, S., Feinstein, P., Mombaerts, P., and Breer, H. 1999. Small subfamily of olfactory receptor genes: structural features, expression pattern and genomic organization. *Gene* 236, 281 – 291.

[77] Tatusov, R. L., Koonin, E. V., Lipman, D. J. 1997. A Genomic Perspective on Protein Families. *Science* 278, 631 – 637.

[78] Thompson, J. D., Higgins, D. G., and Gibson, T. J. 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22, 4673 – 4680.

[79] Touhara, K., Sengoku, S., Inaki, K., Tsuboi, A., Hirono, J., Sato, T., Sakano, H., and Haga, T. 1999. Functional identification and reconstitution of an odorant receptor in single olfactory neurons. *Proc. Natl. Acad. Sci. USA* 96, 4040 – 4045.

[80] van Rhee, A. M. And Jacobson, K. 1996. A. Molecular Architecture of G Protein-Coupled Receptors. *Drug Develop. Res.* 37, 1 – 38.

[81] 1994. *The G-Protein Linked Receptor Factsbook*, ed. Watson, S. And Arkinstall, S. Academic Press, London, U K.

[82] Wess, J. 1998. Molecular basis of receptor/G-protein-coupling selectivity. *Pharmacol. Ther.* 80, 231 – 264.

[83] Wheatley, M. And Hawtin, S. R. 1999. Glycosylation of G-protein-coupled receptors for hormones central to normal reproductive functioning: its occurrence and role. *Hum. Reprod. Update* 5, 356 – 364.

[84] Wu, T. D. And Brutlag, D. L. 1996. Discovering Empirically Conserved Amino Acid Substitution Groups in Databases of Protein Families. *ISMB-96* 3, 230 – 240.

[85] Wu, T. D., Nevill-Manning, C. G., Brutlag, D. L. 1999. Minimal-risk scoring matrices for sequences analysis. *J. Comput. Biol.* 6(2), 219 – 235.

[86] Yona, G., Linial, N., Tishby, N., and Linial, M. 1998. A map of the protein space – An automatic hierarchical classification of all protein sequences. *Proc. Int. Conf. Intell. Syst. Mol. Biol. (ISMB-98)*, 212 – 221.

[87] Zhang, X. And Firestein, S. The olfactory receptor gene superfamily of the mouse. *Nat. Neurosci.* 5, 124 – 133.

[88] Zhao, H., Ivic, L., Otaki, J. M., Hashimoto, M., Mikoshiba, K., and Firestein, S. 1998. Functional expression of a mammalian odorant receptor. *Science* 279, 237 – 242.

[89] Zhou, W., Flanagan, C., Ballesteros, J. A., Konvicka, K., Davidson, J. S., Weinstein, H., Millar, R. P., and Sealfon, S. C. 1994. A reciprocal mutation supports helix 2 and helix 7 proximity in the gonadotropin-releasing hormone receptor. *Mol. Pharmacol.* 45, 165 – 170.

[90] Zozulya, S., Echeverri, F., and Nguyen, T. 2001. The human olfactory receptor repertoire. *Genome Biol.* 2, 0018.0011-0012

Vita

Agatha H. Liu earned a Bachelor of Science degree in Computer Science at Columbia University, a Master of Engineering degree in Computer Science at Cornell University, and a Doctor of Philosophy degree in Computer Science and Engineering at the University of Washington.